BAB I
PENDAHULUAN

1.1 Latar Belakang

Basis data (*database*) merupakan kumpulan data yang telah terbagi dan terhubung secara logikal serta deskripsi dari data yang dirancang untuk memenuhi keperluan informasi suatu organisasi Connolly dan Begg (2010). Pada saat ini terdapat berbagai macam jenis *database* yang didesain untuk berbagai tujuan tertentu seperti *relational database* yang digunakan pada data transaksi (Chickerur, dkk., 2015), dan *graph database* yang digunakan untuk data berbentuk *graph* dengan relasi yang rumit seperti *social network* (Kaliyar, 2015).

Graph database dan *relational database* merupakan *database* yang populer saat ini dan telah banyak pengguna untuk menggunakannya. Akan tetapi ada perdebatan yang sedang berlangsung bahwa *relational database* sudah usang dan orang-orang mencari *database* baru yang akan berfungsi memudahkan pekerjaan mereka.

Tesis ini bertujuan untuk menetapkan apakah *graph database* merupakan alternatif yang lebih baik kinerjanya daripada *relational database* atau setiap jenis *database* mempunyai kelebihan dan kekurangan masing-masng terutama dalam kemampuan performa *create, read, update, dan delete* (CRUD).

Banyak metode *databases* untuk mengelolah data, metode yang sering dipakai saat ini atau lebih familiar digunakan adalah metode *relation database. Relasiona database* telah ada sebagai standar selama lebih dari tiga puluh tahun. Mereka industri yang digunakan luas dan telah menjadi sukses besar hingga saat ini.

Relasional database telah menjadi penyimpanan data nomor satu selama beberapa dekade, menyediakan bisnis dengan standar yang fleksibel. Namun, kebutuhan penyimpanan berubah karena data menjadi lebih besar.

Sebenarnya kita harus mengetahui bahwa tidak harus menggunakan metode *ralation databas*e karena ada juga *database* yang lain seperti *graph database* yang menggunakan struktur data *graph* yg memiliki komponen *node*, *edge* dan *properties* untuk merepresentasikan penyimpanan data. *Graph databases* menyediakan index-free adjacency yang artinya setiap elemen berisi *direct pointer* ke *adjacent element* dan tidak membutuhkan lagi suatu *index lookups. Database* ini dapat diskala lebih ke himpunan data yang lebih besar karena umumnya tidak membutuhkan operasi "*join*" Karena mereka lebih cocok untuk dikelola secara *ad hoc* data yang berubah-ubah dengan skema yang terus diperbarui, Sebaliknya, *relational database* sebuah sistem manajemen *database management system* (RDBMS) adalah sebuah program komputer yang dirancang untuk mengatur/memanajemen sebuah *database* sebagai sekumpulan data yang disimpan secara terstruktur, dan melakukan operasi-operasi atas data permintaan penggunanya. Penggunaan model *database* ada banyak sekali dalam berbagai bidang kerja, misalnya akuntansi, manajemen sumber daya manusia, dan lain sebagainya. Tesis ini bertujuan untuk mengukur secara kuantitatif yang dimaksud adalah penelitan yang melibatkan proses pengumpulan dan analisis data numerik secara obyektif untuk menggambarkan, memprediksi, atau mengontrol variabel yang menarik. Penelitian kuantitatif diekspresikan dalam angka dan grafik. Ini digunakan untuk menguji atau mengkonfirmasi teori dan asumsi.

Dengan membandingkan *graph database* dan *relation databases,* menjadi pegangan untuk melakukan sesuatu rancangan agar cukup umum untuk memodelkan semua kemampuan dari kedua jenis dan informasi yang diminta harus bersifat realistis dan tidak bias.

Untuk mencapai hal ini, dataset yang digunakan adalah data wilayah indonesia yang terdiri dari provinsi beserta kabupaten, kecamatan, dan kelurahan/desa, dan dalam penelitian ini menggunakan data public. [1]

[1] Sumber : https://github.com/cahyadsn/wilayah/tree/70088ad499ff44ddb51f8e7bcf5150d8ca00c172

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka masalah-masalah diteliti dalam penelitian ini adalah Bagaimana menganalisa kinerja *graph databases* dan *relational Database* untuk pengelolahan data sehingga kita mengetahui metode mana yang baik dan paling efektif digunakan.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Bidang studi yang menjadi topik analisa adalah kinerja performa *create, read, update, dan delete* (CRUD). untuk melihat mana yang efektif digunakan dalam mengelolah data yang akan diimplementasikan pada *graph database* dan *relation databases.*
2. Tools Percobaan menggunakan Neo4j dan MySQL

1.4 Tujuan dan Manfaat Penelitian

1.4.1 Tujuan Penelitian

1. Menganalisis kinerja *graph databases*
   Menganalisis kinerja *Relational Database*
2. Membandingkan kinerja *grap databases* dan *Relational Database*, untuk mengetahui kekurangan dan kelebihan dari kedua *database* tersebut

1.4.2 Manfaat Penelitian

Manfaat yang dapat diambil dari pelaksanaan penelitian ini adalah

1. Dapat membantu kinerja *user* dalam pengelolaan data yang ada.
2. Dapat berguna bagi instansi untuk digunakan sebagai rujukan dalam membangun dan mengembangkan sebuah pengelolahan data dimasa yang akan datang

BAB II
LANDASAN TEORI

2.1 Analisa

Analisa adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi bagian – bagian komponen dengan tujuan mempelajari seberapa bagus bagian – bagian komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka (Whitten, 2004 : 176).

Tujuan utama analisis adalah untuk menentukan hal – hal detail tentang yang akan dikerjakan dan yang akan diusulkan dan bukan bagaimana caranya (Kadir, 2003,h.400).

2.2 *Database*

Menurut Fathansyah (2015:2) " *Database* terdiri dari 2 kata yaitu Data dan Base. Data adalah representasi fakta dunia nyata mewakili suatu objek seperti manusia (pegawai, siswa, pembelian pelanggan), barang hewan, peristiwa, konsep, keadaan, dan sebagian yang berwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya". Sedangkan Base kurang lebih dapat diartikan sebagai markas atau gudang, tempat berserang/berkumpul.

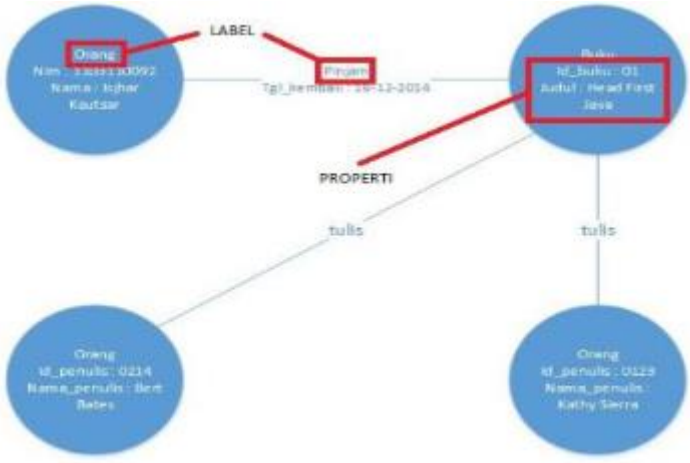
2.2.1 Definisi *Database*

Menurut Rosa dan Shalahuddin (2015:43) "*Database* merupakan salah satu bagian dalam rekayasa perangkat lunak yang terkomputerisasi dan bertujuan utama memelihara data yang sudah diolah atau media penyimpanan informasi agar dapat diakses dengan mudah dan cepat". Sedangkan menurut Yakub dan Hisbanarto (2015:25) menjelaskan, "*database* merupakan kumpulan data yang saling berhubungan atau punya relasi". Dapat disimpulkan bahwa *database* bagian dari rekayasa perangkat lunak yang terkomputerisasi sebagai media penyimpanan informasi yang saling berhubungan atau punya relasi untuk penyimpanan data informasi agar dapat diakses dengan mudah dan cepat.

2.3 *Graph databases*

Graph database adalah suatu model basis data yang menggunakan konsep teori *graph* yaitu dimana data disimpan dalam bentuk simpul/titik/verteks dan ruas/garis/edges dimana verteks dan edge tersebut mencerminkan entitas dunia nyata dan relasi atau hubungannya dengan entitas yang lain (Angles & Gutierrez, 2008; Robinson, dkk., 2015; Shimpi, 2013). Sebelum membahas berbagai jenis *graph database*, kita harus tahu *NOSQL*. *NOSQL* adalah keluarga dari *database*, dan *graph database* adalah bagian dari *NOSQL database*. *NOSQL database* yang dioptimalkan untuk menangani data yang sangat besar dimana unsur-unsur tidak terkait erat, Dalam *NOSQL* volume besar data disimpan secara efisien. Ada berbagai jenis *NOSQL database*. Setiap *graph database* adalah *database* yang menggunakan graph struktur dengan node, tepi dan properti untuk mewakili dan untuk menyimpan data. Model graph yang digunakan dalam *graph database* sangat memungkinkan menyimpan objek tertentu dengan hubungan diantara keduanya. *Graph* dapat dinyatakan sebagai G = (V, E) mana V adalah serangkaian simpul (*node*) dan E adalah serangkaian tepi (relasi/hubungan).

Dalam beberapa tahun terakhir, beberapa *graph database* seperti DEX, Neo4j, Orient DB dan Titan telah dikembangkan, yang saat ini digunakan dioperasi untuk menyimpan data atas sebuah model relasional tradisional. Sebuah sistem *database big table*.



Gambar 2.1 Contoh *Graph Database*

2.3.1 Neo4j

Neo4j saat ini adalah manajemen *graph database* paling populer di pasar (Van Bruggen,2014) Pertama kali dirilis pada 2007. Neo4j adalah *database graph open source* yang diimplementasikan dijava dan dikembangkan oleh teknologi Neo (Vicknair et al., 2010). Ini adalah mesin persistensi Java berbasis disk yang tertanam sepenuhnya, transaksional menyimpan data yang terstruktur dalam *graph* dan bukan dalam berbentuk *tabel*. Neo4j terdiri dari dua bagian, *klien* dan *server*. *Klien* bertanggung jawab untuk mengirim perintah ke *server*, dimana mereka berada dan diproses sebelum hasilnya dikembalikan ke *klien*. Ini diklaim sangat scalable oleh pengembangnya, dengan kemungkinan beberapa miliar node pada satu mesin. API-nya sangat ramah pengguna dan membantu memfasilitasi lintasan yang efisien. Neo4j dibangun menggunakan Lucene dari Apache untuk pengindekkan dan pencarian. Lucene adalah mesin pencari teks, ditulis dalam Java, diarahkan untuk kinerja tinggi (DeCandia et al., 2007).

Gambar 2.2 Neo4j

2.4 *Relational Database*

Relational database adalah koleksi data item yang diorganisasikan sebagai seperangkat tabel yang terdeskripsi secara formal dimana data dapat diakses atau disusun kembali dengan banyak cara tanpa harus mengorganisasikan kembali tabel - tabel basis data. (Bhugul,2015)

Dalam model *relational database*, sebuah *database* adalah kumpulan relasi yang saling terhubung dan tersimpan dalam tabel satu atau lebih satu sama lainnya.

Relasi adalah istilah dalam *relational database*, tapi kita lebih familiar jika menyebutnya sebagai tabel. Selayaknya tabel yang memiliki kolom dan baris, dalam *relational database*, kolom (*column*) disebut *attribute*, sedangkan baris (*row*) disebut *tuple*. Hal ini hanya sekedar penamaan, dan agar lebih gampang, kita hanya akan menggunakan istilah tabel, kolom dan baris dalam tutorial ini, namun jika anda menemui istilah relation, attribut dan tuple, itu hanya penamaan lain dari tabel, kolom, dan baris.

Relasional Database sebenarnya adalah suatu konsep penyimpanan data terstruktur, sebelum konsep *relasional database* muncul sudah ada uda model *database* yaitu *network database* dan *hierarchie database.* Teori relasional database dikemukakan pertamakali oleh Dr. E.F. Codd. Dalam relasional database, data disimpan dalam bentuk relasi atau tabel dua dimensi, dan antara tabel satu dengan tabel lainnya terdapat hubungan atau *relationship* sehingga dapat disimpulkan, *database* adalah kumpulan dari sejumlah tabel yang saling hubungan atau saling keterkaitan. Kumpulan dari data yang diorganisasikan sebagai tabel tadi disimpan dalam bentuk data elektronik didalam harddisk komputer dan dikelompokan secara logis berdasarkan *schema user*.

Untuk membuat struktur tabel, mengisi data ke tabel, memperbarui data dan menghapus data dari tabel diperlukan software. Perangkat lunak yang digunakan membuat tabel, isi data, ubah data, dan hapus data disebut *Relational Database Management System* atau yang biasa disingkat dengan RDBMS. Sedangkan perintah yang digunakan untuk membuat tabel, mengisi tabel, mengubah tabel, dan menghapus data disebut perintah SQL yang merupakan singkatan dari *Structure Query Language*. Jadi, setiap aplikasi perangkat lunak RDBMS pasti bisa dipakai untuk menjalankan perintah SQL.

Sebenarnya fungsi RDBMS bukan cuma untuk buat tabel, isi data, ubah data dan hapus data. Untuk manajemen data dalam skala yang besar dan agar bisa mendukung proses bisnis yang kontinyu atau terus menerus dan *real time* suatu *Relational Database Management System* dituntut untuk mempunyai kemampuan manajemen user dan keamanan data yang terjamin, mencadangkan data dan

mengembalikan data serta kemampuan lainnya yang berkaitan dengan kecepatan pemrosesan data. Sebuah aplikasi perangkat lunak RDBMS yang ada dipasaran saat ini dan paling sering digunakan adalah *Oracle Database* yang dikeluarkan oleh *Oracle Corporation*.

2.4.1 MySQL

MySQL adalah RDBMS open source dan multithreaded yang dibuat oleh Michael "Monty" Widenius pada 1995. Pada tahun 2000, MySQL dirilis dengan lisensi ganda yang mengijinkan publik untuk menggunakannya secara gratis di bawah lisensi GNU GPL (General Public License) yang menyebabkan popularitasnya melambung. Perusahaan yang memiliki dan mengembangkan MySQL adalah MySQL AB (AB = aktiebolag, istilah Swedia untuk perusahaan saham), yang sekarang menjadi anak perusahaan dari Sun Microsystems. Saat ini MySQL AB memperkirakan ada lebih dari 6 juta instalasi MySQL di seluruh dunia, dan melaporkan rata-rata jumlah unduhan instalasi MySQL dari situsnya dan situs mirror sebanyak 50.000 per hari. Keberhasilan MySQL sebagai basis data terkemuka adalah tidak hanya karena harganya, tetapi juga karena kehandalan, kinerja dan fitur-fiturnya.Banyaknya fitur MySQL membuat database ini tetap menjadi sistem basis data yang hebat. Kecepatan adalah salah satu fiturnya yang menonjol. Dalam perbandingan oleh eWeek pada beberapa basis data (MySQL, Oracle, MS SQL, IBM DB2, dan Sybase ASE),

2.5 SQL vs NoSQL

2.5.1 SQL

SQL (*Structure Query Language)* merupakan bahasa untuk mengelola database, Secara lisan sangat mudah dibaca karena merupakan bahasa generasi keempat, artinya sintak nya sudah menggunakan kata yang mudah dibaca oleh manusia, contohnya SELECT, FROM, WHERE dll. Didalam SQL terdapat dua tipe bahasa yaitu DDL (Data Definition Language) dan DML (Data Manipulation Language). Dua tipe bahasa tersebut mempunyai fungsi yang berbeda tentunya. DDL digunakan untuk membuat stuktur tabel, sedangkan DML digunakan untuk mengelola data didalam tabel yang telah dibuat. SQL biasa digunakan pada data

model relational, artinya setiap tabel saling berhubungan untuk menghasilkan informasi yang diinginkan.

2.5.2 NoSQL

NoSQL bukan merupakan bahasa. NoSQL adalah sebuah mekanisme untuk menyimpan data dan mengambil data yang dilakukan oleh database kita. NoQSL tidak membutuhkan data *model relational* dan bahasa SQL untuk melakukan hal tersebut. NoSQL menggunakan metadata pada *database* kita dan memanfaatkan index dari data tersebut. NoSQL mempunyai empat mekanisme:

- *Table-oriented.*
- *Graph-oriented.*
- *Document-oriented database.*
- *Key-value store*, contoh: Memcache dan Redis

Jika kita bandingkan SQL dengan NoSQL, masing – masing mempunyai keuntungan dan kerugian masing – masing.[1]

[1] Sumber: ***https://sis.binus.ac.id/2016/08/23/sql-vs-nosql/***

2.6 Penelitian Terdahulu

Tabel 2.1. Penelitian Terdahulu

No	Peneliti	Judul	Metode/ parameter	Keterangan
1	Surajit Medhi , Hemanta K. Baruah, 2017	Relational Database And *Graph* Database: A Comparative Analsis.	Paramaeternya menggunakan tool mysql dan neo4j	Pentingnya database relasional akan menurun karena pertumbuhan eksponensial data karena sulit untuk bekerja dengan jumlah besar dengan tabel. salah satu solusi terbaik adalah untuk menggunakan *graph* database untuk menyimpan data.
2	Panji Wisnu Wirawan, Djalal Er Riyanto, 2017	Kajian Implementasi *Graph* Database pada Rute Bus Rapid Transit.	GDM (*Graph* Data Model	Artikel ini mengusulkan sebuah model *graph* database untuk BRT dan implementasinya. Identifikasi kebutuhan data dilakukan, dilanjutkan dengan pemodelan menggunakan entity relationship (ER). Hasil ER tersebut kemudian dipetakan ke dalam property *graph* untuk kemudian diimplementasikan menggunakan produk *graph* database Neo4J.

3	Ognjen Orel, Slaven Zakošek, Mirta Baranovic, 2016.	Property Oriented Relational-To-Graph Database Conversion.	graph database model	Graph database mampu menjadi solusi berbagai permasalah tersebut. Namun, graph database sendiri masih memiliki beberapa kekurangan, yaitu pencarian dilakukan secara sekuensial pada saat proses retrieve data
4	Isjhar Kautsar, Kemas Rahmat Saleh W., S.T., M.Eng, Gia Septiana Wulandari, S.Si., M.Sc, 2015.	Analisis Performansi Metode Graph Decompositio n Index pada Graph Database.	graph decomposition index	banyak relasi mendorong para peneliti untuk menemukan model database yang baru. Salah satunya graph database. Graph database mampu menjadi solusi berbagai permasalah tersebut. Namun, graph database sendiri masih memiliki beberapa kekurangan, yaitu pencarian dilakukan secara sekuensial pada saat proses retrieve data. Oleh karena itu, suatu metode diperlukan untuk mengatasi masalah ini, yaitu dengan indexing. Metode indexing yang akan digunakan adalah graph decomposition index.

5	Pradana Setialana, Teguh Bharata Adji, Igi Ardiyanto, 2017	Perbandingan Performa Relational, Document-Oriented dan Graph Database Pada Struktur Data Directed Acyclic Graph	Directed Acyclic Graph(DAG) Metode yang dilakukan adalah mengimplentasi dataset	Tujuan penelitian ini adalah membandingkan performa dari relational database (PostgreSQL), document-oriented database (MongoDB) dan graph database (Neo4j) pada data DAG. Metode yang dilakukan adalah mengimplentasi dataset yang memiliki 3910 node dalam operasi single write synchronous (SWS) dan single read (SR) pada setiap database menggunakan Node.js
6	Mesri Silalahi, Didi Wahyudi, 2018.	Perbandingan Performansi Database Mongodb Dan Mysql Dalam Aplikasi File Multimedia Berbasis Web	engine yang biasa digunakan adalah MyISAM	Selain database relasional (SQL), yang menyimpan data terstruktur dalam tabel dengan skema yang ditetapkan, ada database non-relasional (NoSQL) dengan skema dinamis atau tidak terstruktur. Studi ini akan membandingkan kinerja antara database NoSQL (MongoDB) dan database SQL (MySQL) untuk aplikasi penyimpanan file multimedia berbasis web yang menyimpan file

				sebagai BLOB.
7	Hadyan Arif, Kemas Rahmat Saleh, Dr. Adiwijaya, 2015	Analisis dan Implementasi Graph Indexing Pada Graph Database Menggunakan Algoritma GIndex	GIndex	Dalam mempercepat pemrosesan query pada graph database dibutuhkan suatu metode yang dapat disebut graph indexing agar lebih cepat dan efisien. GIndex merupakan salah satu metode graph indexing yang mendukung pemrosesan query bertipe subgraph query.
8	ShefaliPatil, GauravVaswani, Anuradha Bhatia, 2014	Graph Databases- An Overview	Tidak menggunakan metode penelitian ini lebih terfokus ke pada tool yaitu neo4j	Saat ini, tren berubah karena basis data Graph dengan cepat mendapatkan popularitas. Bahkan, tidak salah jika menyebut mereka "masa depan DBMS". Representasi data dalam bentuk graph cocok untuk data terstruktur dengan skema dinamis. Makalah ini membahas aplikasi saat ini dan implementasi graph database, memberikan gambaran tentang berbagai jenis yang tersedia diaplikasi mereka. Karena penyebaran luas dari algoritma graph dan model,

				tidak ada sistem standar atau bahasa query telah ditolak untuk graph databases. Penelitian dan adopsi industri akan menentukan arah masa depan dari graph database.
9	Marcus Walldén, Aylin Özkan, 2016.	A graph database management system for a logistics-related service	Waterfall Model	Tesis ini menyajikan prototipe layanan terkait jaringan logistik menggunakan sistem manajemen basis data grafik yang disebut Neo4j, yang saat ini merupakan sistem manajemen basis data grafik paling populer yang digunakan. Jaringan logistik yang dicakup oleh layanan ini didasarkan pada data yang ada dari PostNord, penyedia solusi logistik terbesar diSwedia, dan terutama berfokus pada dukungan pelanggan dan bisnis ke bisnis.
10	Muntasir Rahman, Petrus Mursanto,	Perbandingan Kinerja Basis Data Relasionalde	Metrics of Object-Oriented Design	Penelitian ini bertujuan mengkaji standar penerapan model objek data dan metode perancangan skema

	2016	ngan Basis Data Berorientasio bjek Studi Kasus: Aplikasi Jpetstore	(MOOD)	data pada BDBO melalui pengukuran kinerja dan kualitas kode dari aplikasi. Penelitian ini mengkaji penerapan model data ODMG 3.0 dan notasi UML pada aplikasi JPetStore dengan menggunakan transformasi Muller untuk perancangan skema data.
11	Faizal Anugrah Bhaswara, Riyanarto Sarno,dan Dwi Sunaryono, 2017.	Perbandingan Kemampuan Database NoSQL dan SQL dalam Kasus ERP Retail	metode sharding	perbandingan database NoSQL dengan SQL dalam hal kinerja, fleksibilitas, dan skalabilitas. Setelah terbukti database mana yang tepat, maka akan diterapkan aplikasi ERP Retail dengan yang berorientasikan multitenancy dengan tujuan agar aplikasi memiliki performa bagus, fleksibel dalam penyimpanan data, juga mendukung penyimpanan data yang terus berkembang seiring berjalannya waktu.
12	Vatika Sharma, Meenu Dave, 2012.	SQL and NoSQL Databases		Dalam makalah penelitian ini, kami mensurvei tentang NoSQL, latar belakangnya, dasar-dasarnya seperti

				ACID, BASE, dan teorema CAP. Juga atas dasar teorema CAP, penelitian dilakukan tentang berbagai jenis penyimpanan data NoSQL dengan contoh, karakteristik, dan pro dan kontra dari NoSQL.

BAB III
Metodologi Penelitian

3.1. Metode Penelitian

Metode yang dipakai dalam penelitian ini adalah menggunakan metode eksperimen dan metode komparasi

a. Metode Eksperimen

Metode eksperimen dilakukan untuk dapat menghasilkan suatu data yang diperlukan dengan parameter yang sudah ditentukan. Metode eksperimen dilakukan dengan cara melakukan pengetesan lansung pada media yang diinginkan.

b. Metode Komparasi

Metode komparasi dilakukan untuk mencari perbandingan data yang dihasilkan dari metode eksperimen. Pada tahap komparasi, data yang sudah didapat kemudian dihitung untuk mendapatkan suatu hasil perbedaan.

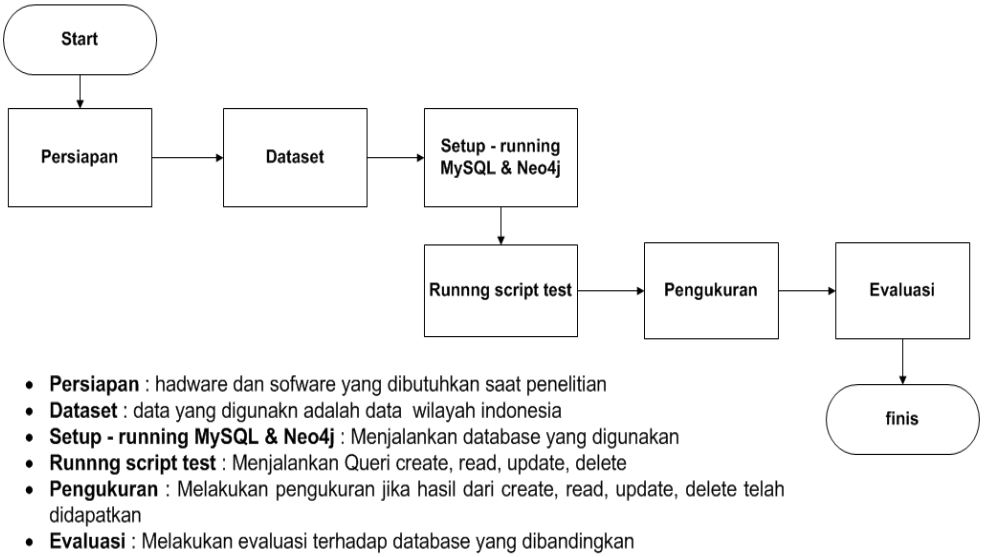
3.2 Metode Pengumpulan Data

Dalam mengumpulkan data penelitian ini menggunakan metode studi pustaka. Metode yang dilakukan adalah dengan cara mancari bahan yang mendukung dalam pendefinisian masalah melalui buku-buku, *internet*, yang erat kaitannya dengan objek permasalahan.

3.3 Kerangka Penelitian

Tahap bagian ini adalah untuk menetapkan kerangka kerja yang menjalankan percobaan pada aspek yang dijadikan acuan perbandingan dari kedua database adalah aspek DML (*Data Manipulation Language)* dan CQL *(Cypher Query Language)*, yaitu operasi CRUD (*Create, Read,Update, Delete*).

Untuk menentukan berbagai tujuan yang dibahas, dibagi menjadi tiga fase, persiapan, pengukuran, dan evaluasi, yang akan dibahas di bawah ini.



Gambar 3.1. Alur Penelitian Kerangka Kerja

3.4 Tahap persiapan

Sebelum melakukan percobaan, harus dimulai dengan mempersiapkan perangkat lunak dan perangkat keras yang di perlukan untuk memfasilitasi pada saat tahap pengujian, adapun perangkat keras dan perangkat lunak yang dipergunakan adalah:

Tabel 3.1 Hardware dan Sofware

Laptop *processor Intel Core i3*	PC *processor Intel Core i3*
RAM 8 Gb,	RAM 8 Gb,
HDD 500 Gb,	HDD 500 Gb,
Os windows 10,	Os windows 10,
Neo4j	mySQL

Semua DBMS tersebut terinstall dalam laptop dan komputer dalam spesifikasi yang sama yang akan digunakan pada tahap pengukuran graph databases dan relation databases

3.5 Perbandingan databases

Aspek – aspek yang dijadikan acuan perbandingan dari kedua database ini adalah aspek *Data Manipulation Language*, dan *Cypher Query Language* yaitu operasi CRUD (*Create, Read, Update, Delete*). Perbandingan dari *database – database* tersebut dipisahkan menjadi 2 bagian, yaitu:

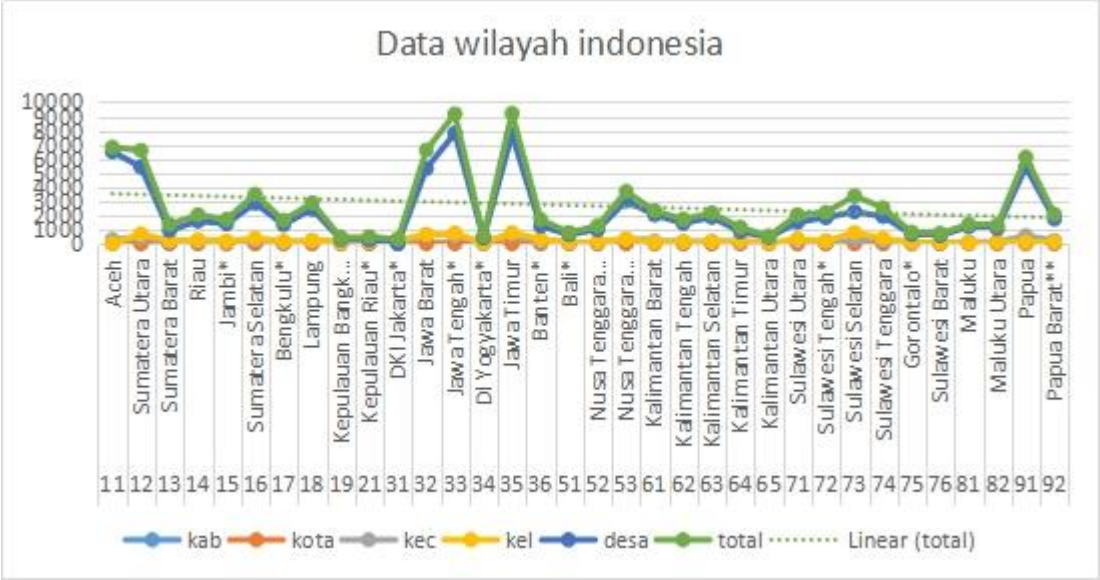
- Perbedaan syntax *Data Manipulation Language* dan *Cypher Query Language* untuk *graph database* dan *relational database*.
- Perbedaan kinerja CRUD antara *graph database* dan *relational database*.

3.6 *Dataset*

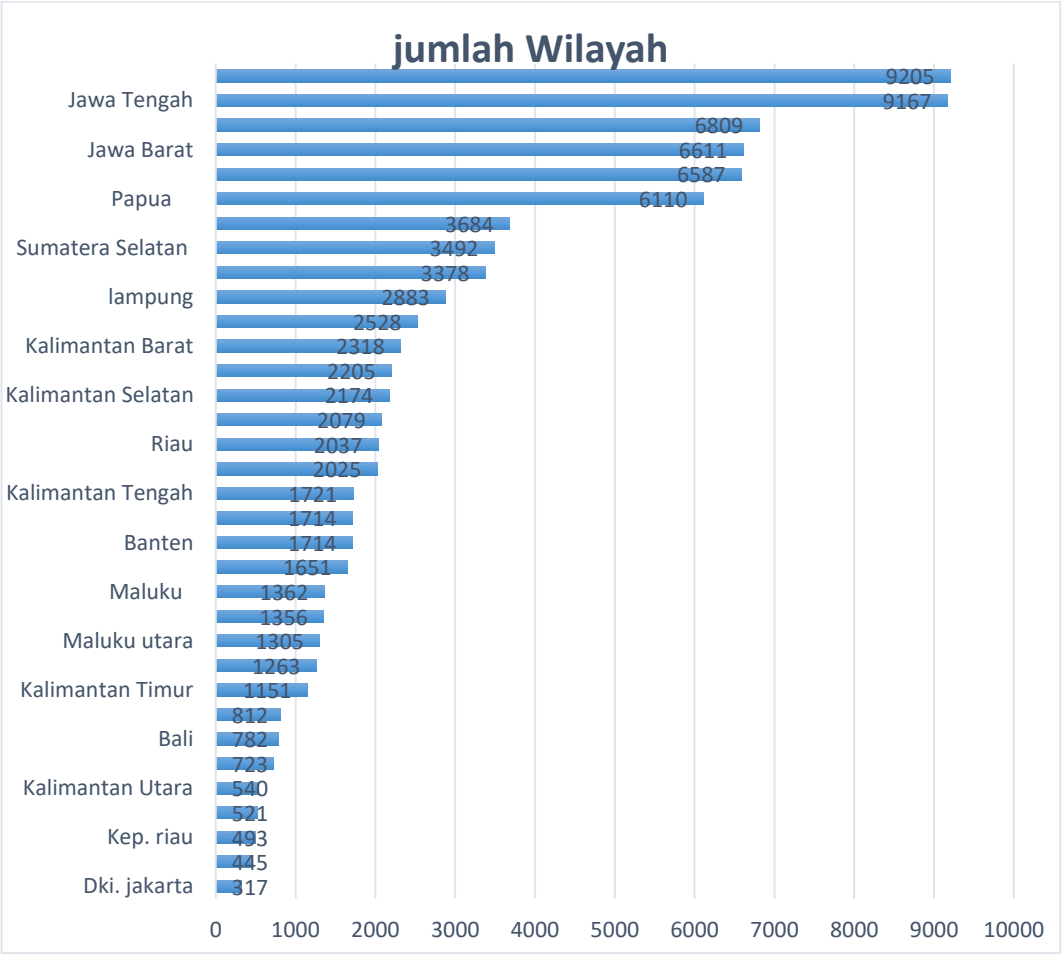
Dalam penelitian ini akan digunakan dua jenis Tools yaitu menggunakan Neo4j untuk *graph database* dan MySQL untuk *relational database*. *Dataset* yang digunakan dalam penelitian ini menggunakan data *public* wilayah Indonesia data provinsi, kabupaten, kecamatan dan kelurahan /desa *dataset* yang digunakan tersebut berisi 34 node yaitu 34 provinsi yang terdiri dari sekitar 91162 wilayah. Sudah termasuk data kabupaten, kecamatan dan kelurahan /desa sesuai dengan Permendagri No 137 tahun 2017.

Dataset akan dibagi secara bertahap. Untuk uji coba dilakukan dengan input *dataset* dengan jumlah yang sama dan dimasukkan secara berulang. Misalnya, inputan pertama menggunakan 1000 data yang meningkat secara bertahap dengan 2000, 3000, 4000, 5000, 8000. *Dataset* akan dijadikan percobaan adalah jumlah data wilayah yang paling kecil hingga sampai kedata yang peling paling besar, yang dapat kita liat di gambar 3.3 [1]

[1] Sumber : https://github.com/cahyadsn/wilayah/tree/70088ad499ff44ddb51f8e7bcf5150d8ca00c172

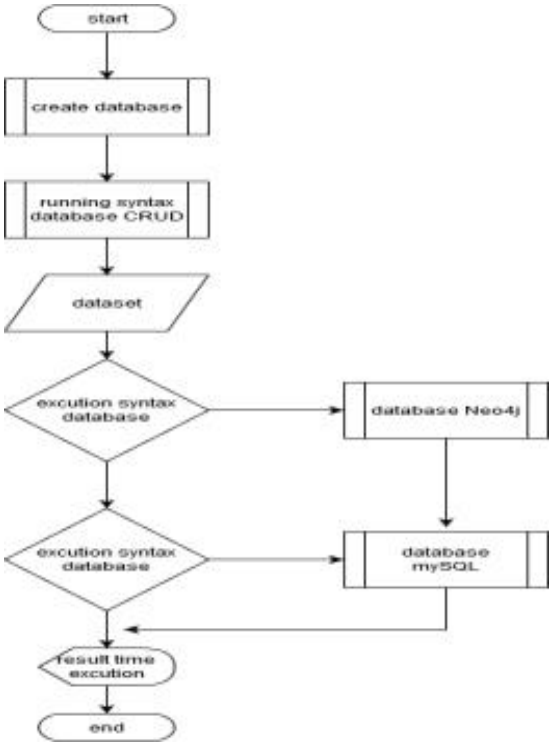


Gambar 3.2 Data Provinsi Indonesia



Gambar 3.3 Jumlah Wilayah

3.7 Tahap pengujian pengukuran



Gambar 3.4. flowchart Alur Kerja Tahap Pengukuran

Penelitian ini melakukan percobaan penyimpanan data dari *dataset* ke dalam *database* secara berurutan. Prosedur pengujian terhadap performa dilakukan sesuai dengan proses *create, read, update, delete*, (CRUD), data yang dilakukan perprovinsi dari jumlah wilayah terkecil sampai terbesar dan dilakukan sempai data habis. Setiap percobaan data ditulis ke dalam *database* secara berurutan (*synchronous*) sesuai dengan bahasa *query* setiap jenis *database*. Sebelum data dilakukan perulangan pada proses menyimpan data, sistem mengambil informasi waktu pada saat itu. Setelah proses perulangan berhenti, sistem kemudian mendapatkan waktu keseluruhan yang digunakan untuk menyimpan data.

3.8 Tahap evaluasi

Gambar 3.5 Flowchart Alur Evaluasi

Pada gambar 3.5 tersebut terlihat bahwa prinsip pengujian pada tahap evaluasi yaitu dengan menghitung waktu keseluruhan dibagi dengan jumlah data. Sehingga pengujian pada operasi CRUD dapat ditulis dalam bentuk Persamaan (1) dan (2). Persamaan (1) menunjukan perhitungan total waktu dalam dataset yang dibutuhkan untuk *create, read, update, delete* ke dalam *database*, terlihat bahwa terdapat fungsi "*Date.now()*" yang merupakan untuk mengambil waktu saat itu ketika fungsi tersebut dipanggil.

```
timeStart = Date.now(); // sebelum _ perulangan
timeEnd = Date.now(); // setelah _ perulangan
totalTime = timeEnd – timeStart;
```

Persamaan (2) menunjukan perhitungan waktu rata–rata yang digunakan untuk semua data dari basis data. Dalam perhitungan waktu rata-rata tersebut terdapat fungsi "*data.length*" yang merupakan fungsi untuk mengambil informasi jumlah data yang ada pada dataset. Sedangkan "*totalTime*" adalah nilai variabel

waktu keseluruhan yang didapat dari Persamaan (1). Hasil dari waktu rata–rata (*averageTime*) tersebut digunakan untuk mengambil kesimpulan mengenai performa penyimpanan data diambil dari waktu rata-rata yaitu total waktu dibagi dengan jumlah data. (setialana perdana, dkk., 2017).

$$averageTime = \frac{totalTime}{datalength}$$

BAB IV
HASIL DAN PEMBAHASAN

4.1. Pembahasan

Perbedaan *Syntax Data Manipulating Language* (DML) dan *syntax Cypher Query Language* (CQL) adalah seperangkat perintah yang dimiliki oleh *database* yang meliputi *Create, Read, Update, Delete*, Perbedaan *syntax* antara *Graph database* dengan Neo4j akan dianalisa dengan *syntax* pada *Relational database* pembanding yaitu mySQL

4.2. Tahaf Pengujian Pengukuran

4.2.1. *Syntax* Dml *Relational Database* Dan *Syntax* Cql Untuk *Graph Database*.

A. Berikut ini adalah *syntax* perintah DML dalam *Relational database*

1. Insert

```
INSERT INTO nama_tabel VALUES('nilai_masukan', 'nilai_masukan', 'dst');
```

2. Update

```
UPDATE nama_table SET nama_column = value where [condition]
```

3. Delete

```
DROP TABLE nama_tabel;
```

4. Select

```
SELECT * FROM nama_tabel
```

B. Berikut ini adalah *syntax* perintah CQL dalam *Graph database*

1. Insert

```
CREATE (n:nama_node {nama:"orang"})
```

2. Update

```
MATCH (nama_node) where nama_node.nama="orang"
SET nama_node.nama="orang2."
RETURN nama_node
```

3. Delete

```
Match (n:nama_node) DELETE n
```

4. Select

```
MATCH (n:nama_node)RETURN n
```

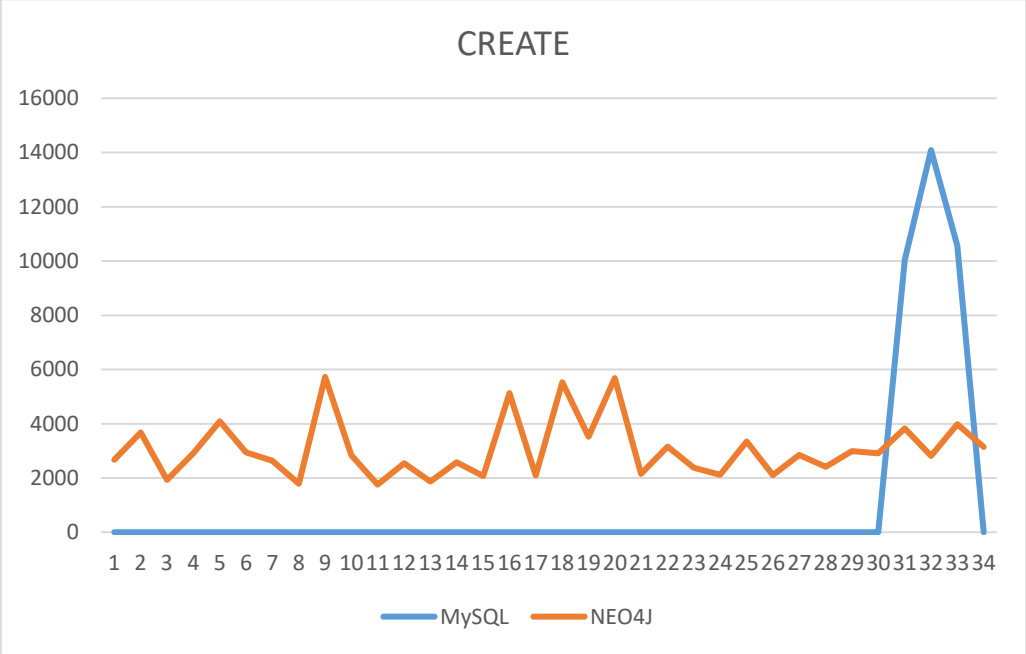
4.2.2. Pengujian kinerja CRUD antara *Relational Database* dan *Graph Database.*

Dengan menggunakan dataset wilayah indonesia , kinerja *Data Manipulation Language* dan *Cypher Query Language* dari kedua database dibandingkan, yaitu dalam aspek CRUD (Create, Read, Update, dan Delete) Dalam masing-masing percobaan, pertama-tama transaksi khusus dieksekusi memberikan titik awal untuk memperkirakan bagaimana sistem akan berperilaku ketika cukup data dalam sistem. dengan hasil dari masing – masing aspeknya adalah sebagai berikut.

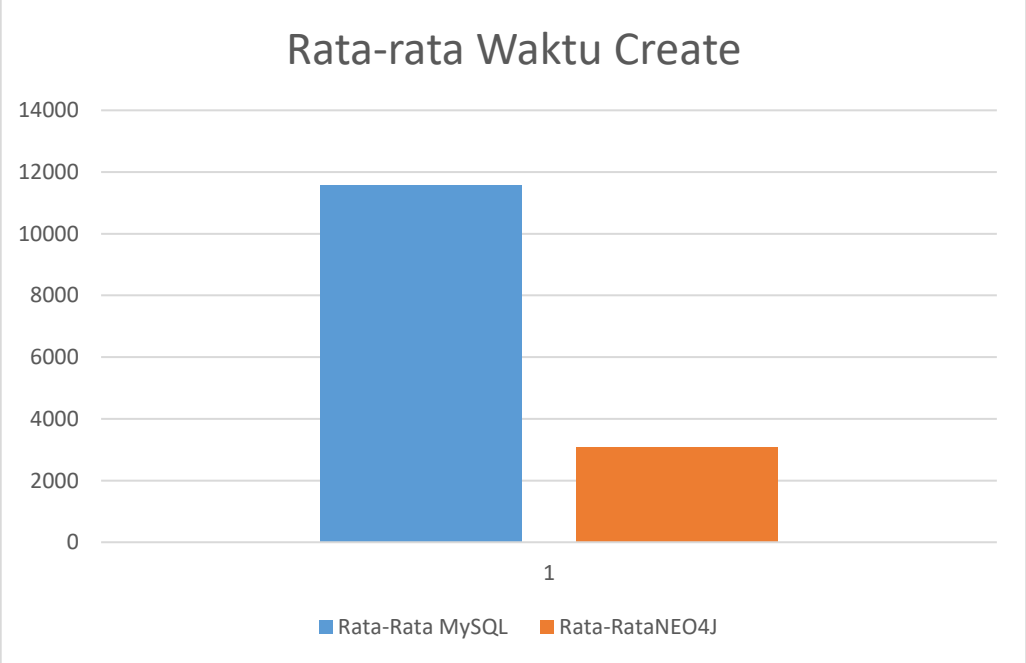
4.2.2.1 Perintah *Create*

Pada percobaan *create***,** telah diukur performanya yang dinilai dalam hal waktu pelaksanaan dengan memvariasikan beban kerja jumlah transaksi yang dilakukan dengan input dataset dengan jumlah yang sama, Runtime hasil uji coba dicatat setiap transaksi dari mulai *create* 317 data pertama, 445 data kedua, 493 data ketiga, dan seterusnya, Berdasarkan data yang terkumpul dibuat grafik *runtime* dan rata-rata perintah *create* dari MySQL dan Neo4j agar lebih mudah untuk dianalisa,

hasil ditampilkan pada gambar 4.1dan 4.2 sebagai berikut (garis lebih rendah menunjukkan lebih baik):



Gambar 4.1 Grafik *Runtime create* Neo4j vs MySQL



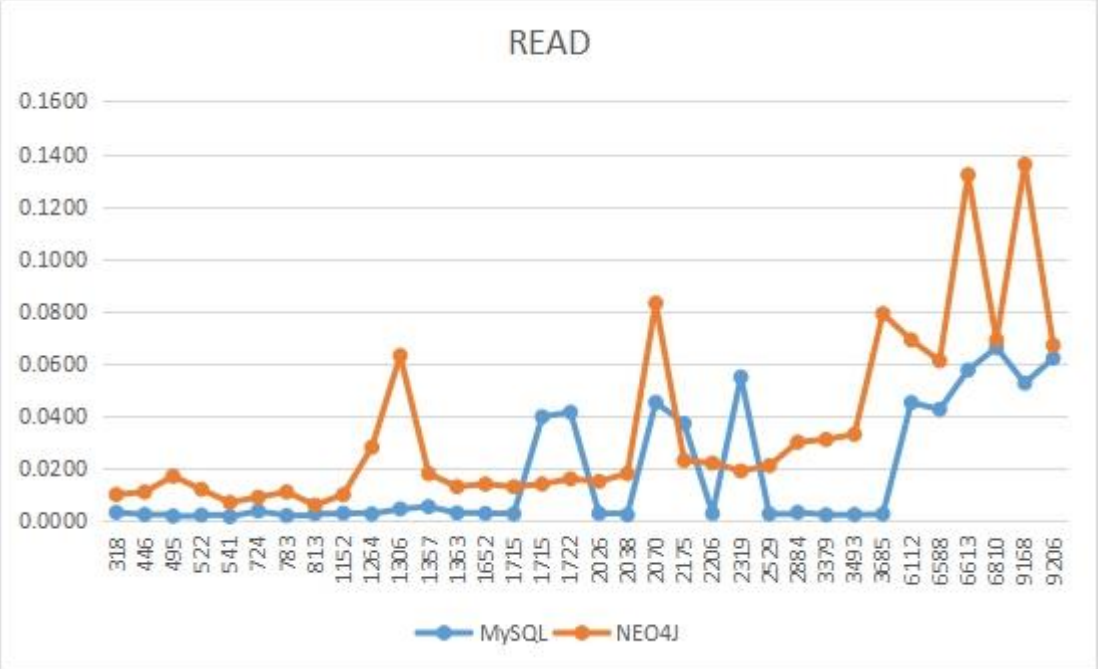
Gambar 4.2 Grafik Rata-Rata *Runtime create* Neo4j vs MySQL

Pada gambar 4.1 diatas menunjukkan bahwa pada *runtime create* dalam Neo4j lebih lambat dari pada perintah *create* dalam MySQL tetapi dari hasil gambar 4.2 rata-rata waktu Neo4j lebih renda dari Mysql dikarenakan pada pengujian create data 6611, 6809 dan 9167 *runtime create* dalam MySQL lebih lambat

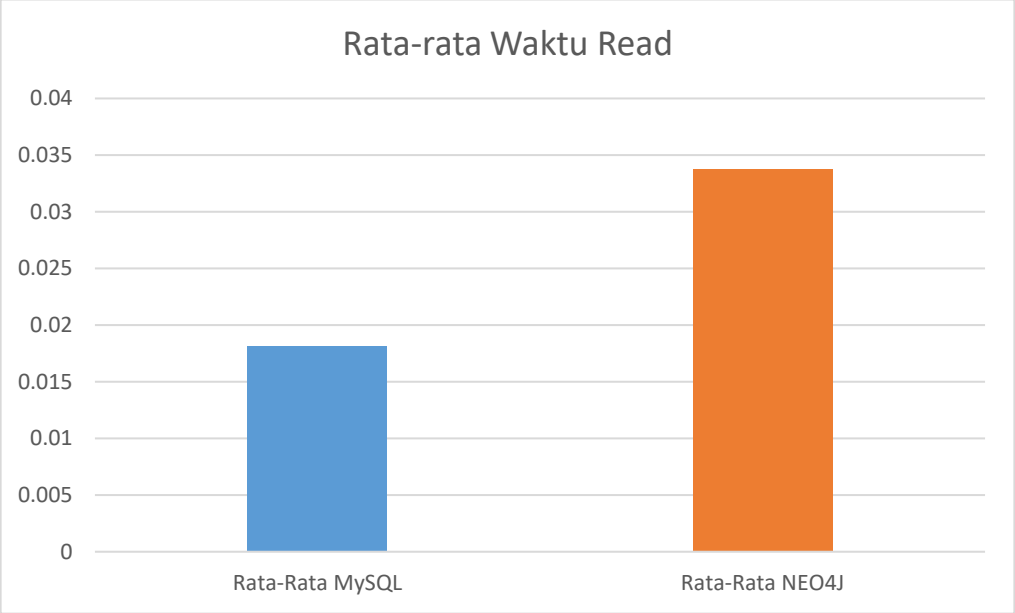
4.2.2.2 Perintah *Read*

Pada percobaan read, telah diukur performanya yang dinilai dalam hal waktu pelaksanaan dengan memvariasikan beban kerja jumlah transaksi yang dilakukan dengan input dataset dengan jumlah yang sama, Runtime hasil uji coba dicatat setiap transaksi per detik (tps) dimulai read 317 data pertama, 445 data kedua, 493 data ketiga, dan seterusnya.

Berdasarkan data yang terkumpul dibuat grapik runtime dan rata-rata perintah read dari MySQL dan Neo4j agar lebih mudah untuk dianalisa, hasil ditampilkan pada gambar 4.3 dan 4.4 (garis lebih rendah menunjukkan lebih baik):



Gambar 4.3 Grafik *Runtime read* Neo4j vs MySQL



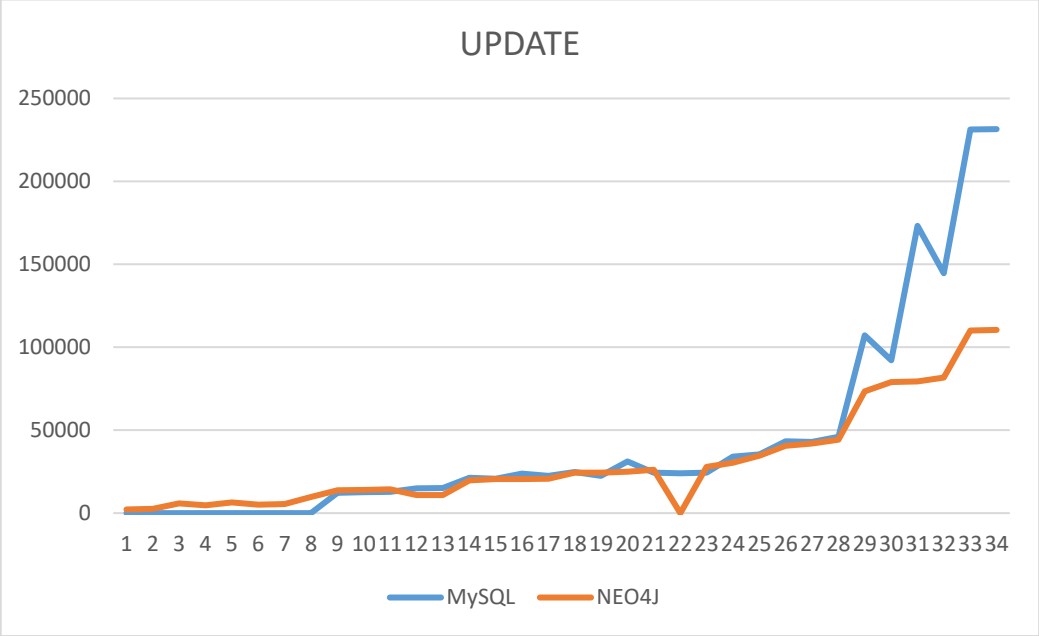
Gambar 4.4 Grafik Rata-Rata *Runtime Read* Neo4j vs MySQL

Gambar 4.3 diatas menunjukkan bahwa pada *runtime read* dalam Neo4j lebih lambat daripada perintah *read* dalam MySQL tetapi disaat melakukan *read* data 1715, 1722 dan 2319 Neo4j lebih cepat di bandingkan MySQL dan dari hasil gambar 4.4 rata-rata waktu Neo4j lebih tinggi dari Mysql dikarenakan pada pengujian read Mysql lebih unggul.

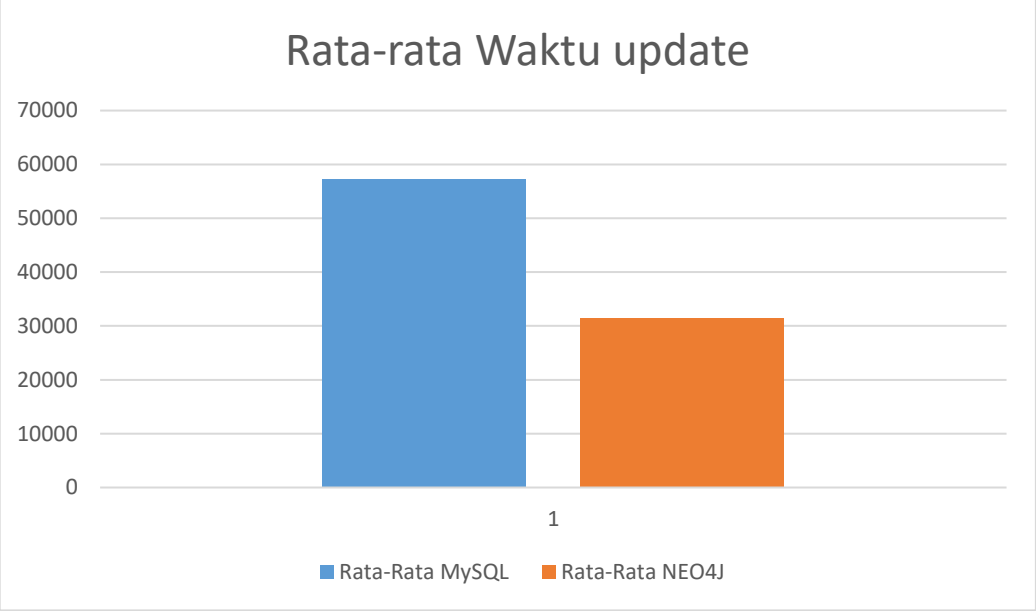
4.2.2.3 Perintah *Update*

Pada percobaan *update*, telah diukur performanya yang dinilai dalam hal waktu pelaksanaan dengan memvariasikan beban kerja jumlah transaksi yang dilakukan dengan input dataset dengan jumlah yang sama, Runtime hasil uji coba dicatat setiap transaksi per detik (tps) dimulai *update* 317 data pertama, 445 data kedua, 493 data ketiga, dan seterusnya.

Berdasarkan data yang terkumpul dibuat grapik runtime dan rata-rata perintah *update* dari MySQL dan Neo4j agar lebih mudah untuk dianalisa, hasil ditampilkan pada gambar 4.5 dan 4.6 (garis lebih rendah menunjukkan lebih baik)



Gambar 4.5 Grafik *Runtime update* Neo4j vs MySQL



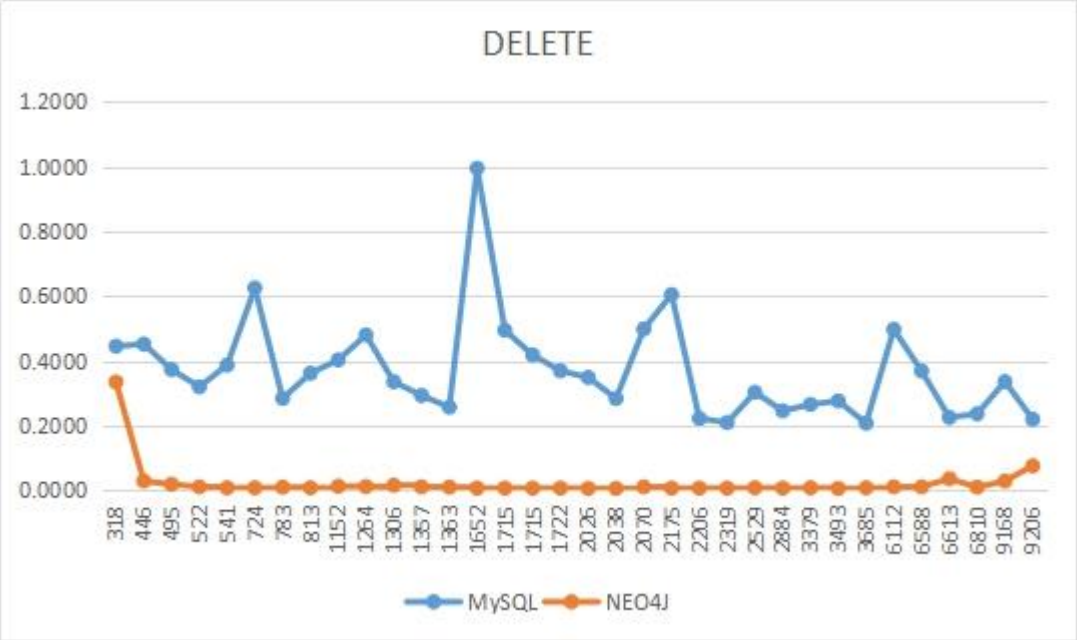
Gambar 4.6 Grafik Rata-Rata *Runtime update* Neo4j vs MySQL

Pada pengujian *syntax update* disini dilakukan pengujian dengan cara berbeda dikarenakan diMysql dapat melakukan update secara banyak, sedangkan di Neo4j tidak, sehingga memakan waktu untuk melakukan *update* di Neo4j

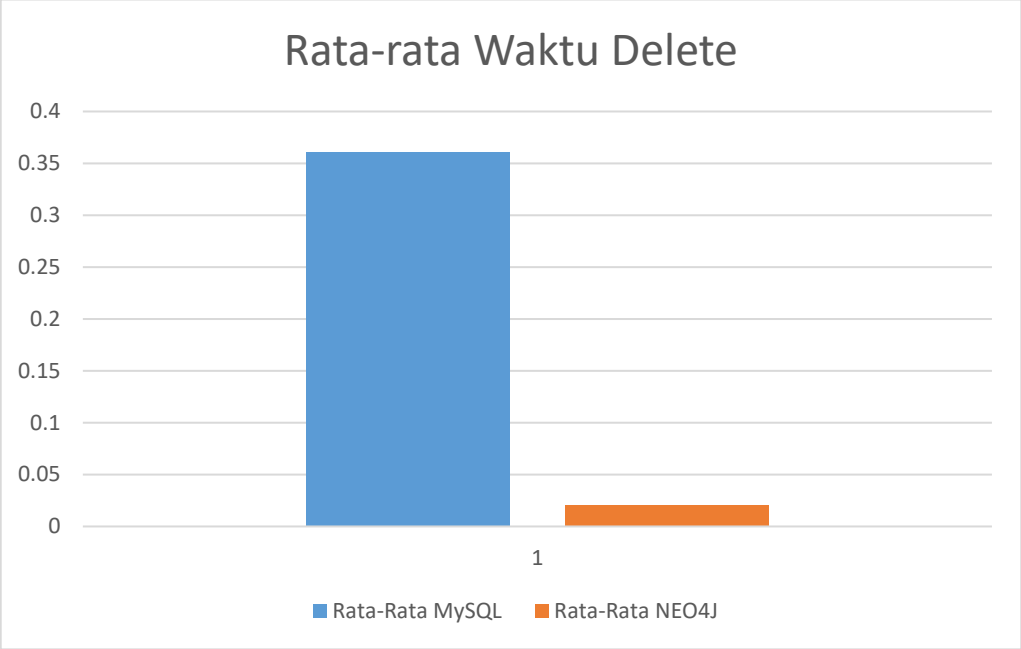
4.2.2.4 Perintah *Delete*

Pada percobaan *delete*, telah diukur performanya yang dinilai dalam hal waktu pelaksanaan dengan memvariasikan beban kerja jumlah transaksi yang dilakukan dengan menghapus dataset dengan jumlah yang sama, Runtime hasil uji coba dicatat setiap transaksi per detik (tps) dimulai *update* 317 data pertama, 445 data kedua, 493 data ketiga, dan seterusnya.

Berdasarkan data yang terkumpul dibuat grapik runtime dan rata-rata perintah *delete* dari MySQL dan Neo4j agar lebih mudah untuk dianalisa, hasil ditampilkan pada gambar 4.7 dan 4.8 (garis lebih rendah menunjukkan lebih baik)



Gambar 4.7 Grafik *Runtime Delete Neo4j vs MySQL*



Gambar 4.8 Grafik Rata-Rata *Runtime Delete* Neo4j vs MySQL

Gambar 4.7 dan 4.8 diatas menunjukkan bahwa pada runtime *delete* dalam Neo4j lebih cepat daripada perintah *delete* dalam MySQL

4.3 Evaluasi

Dengan menggunakan metode yang telah dijelaskan sebelumnya, terdapat empat macam evaluasi penelitian yaitu performa basis data pada operasi menyimpan *dataset* ke dalam *database*

Pada tahap pengukuran pengujian pada CRUD ditulis dalam bentuk Persamaan (1) dan (2). Persamaan (1) menunjukan perhitungan total waktu dalam dataset yang dibutuhkan terlihat bahwa terdapat fungsi "*Date.now()*" yang merupakan untuk mengambil waktu saat itu ketika fungsi tersebut dipanggil.

Persamaan (2) menunjukan perhitungan waktu rata–rata yang digunakan untuk semua data dari *database*. Dalam perhitungan waktu rata-rata tersebut terdapat fungsi "*data.length*" yang merupakan fungsi untuk mengambil informasi jumlah data yang ada pada *dataset* (setialana perdana, dkk., 2017).

Tabel 4.1 *data.length*

syntax	**MySQL**	**neo4j**
create	15.4694	104.173
read	0.6083	1.180
update	152.0995	1067.271
delete	12.6294	0.738

Sedangkan "*totalTime*" adalah nilai variabel waktu keseluruhan yang didapat dari Persamaan (1).

Tabel 4.2 *date.now*

syntax	time Start	time End	total Time
create	11:31	13:55	1.06
read	13:08	14:06	1.13
update Mysql	14:00	18:30	1.35
update Neo4j	12/1/2019 6:30	12/30/2019 14:45	87629.89
delete	20:30	22:15	1.78

Hasil dari waktu rata–rata (*averageTime*) tersebut digunakan untuk mengambil kesimpulan mengenai performa penyimpanan data diambil dari waktu rata-rata yaitu total waktu dibagi dengan jumlah data (*data length*)

Pemberitahhuan : untuk Pengerjaan update di Neo4j melakukan waktu 4 jam dalam 1 hari

4 jam x 30 hari = 120 jam'

120 jam di bulatkan menjadi hari

total 120 jam = 5 hari

total time di kali dengan total hari

87629.89 x 5 hari = 438149.45

Jadi *datenow* untuk percobaan di Neo4j adalah 438149.43

4.3.1 Evaluasi *create MySQL dan Neo4j*

Tabel 4.3 evaluasi Create MySQL dan Neo4j

create MySQL	
$timeStart = 11:31$ $timeEnd = 13:55$ $totalTime = 11:31 + 13:55$ $=$ 1.06	$averageTime = \frac{0.56}{15.4694}$ $=$ 14.598
Create Neo4j	
$timeStart = 11:31$ $timeEnd = 13:55$ $totalTime = 11:31 + 13:55$ $=$ 1.06	$averageTime = \frac{0.56}{104.173}$ $=$ 98.302

4.3.2 Evaluasi *Read MySQL dan Neo4j*

Tabel 4.4 *evaluasi Read MySQL* dan *Neo4j*

Read MySQL	
$timeStart = 13:08$ $timeEnd = 14:06$ $totalTime = 13:08 + 14:06$ $= 1.13$	$averageTime = \frac{1.13}{0.6083}$ $= 0.536$
Read Neo4j	

$timeStart = 13:08$ $timeEnd = 14:06$ $totalTime = 13:08 + 14:06$ $= 1.13$	$averageTime = \frac{1.13}{1.180}$ $= 1.040$

4.3.3 Evaluasi *Update MySQL dan Neo4j*

Tabel 4.5 *evaluasi Update MySQL* dan *Neo4j*

Update MySQL	
$timeStart = 14:00$ $timeEnd = 18:30$ $totalTime = 14:00 + 18:30$ $= 1.35$	$averageTime = \frac{1.35}{152.0995}$ $= 112.320$
Update Neo4j	
$timeStart = 12/1/2019\ 6:30$ $timeEnd = 12/30/2019\ 14:45$ $totalTime = 12/1/2019\ 6:30 + 12/30/2019\ 14:45$ $= 87629.89 \text{ X } 5$ $= 438149.45$	$averageTime = \frac{438149.45}{1067.271}$ $= 410.5332$

4.3.1 Evaluasi *Delete MySQL dan Neo4j*

Tabel 4.6 *evaluasi Delete MySQL* dan *Neo4j*

Delete MySQL	
$timeStart = 20:30$ $timeEnd = 22:15$ $totalTime = 20:30 + 22:15$ $= 1.78$	$averageTime = \frac{1.78}{12.6294}$ $= 7.090$
Delete Neo4j	

$$timeStart = 20:30$$
$$timeEnd = 22:15$$
$$totalTime = 20:30 + 22:15$$
$$= 1.78$$

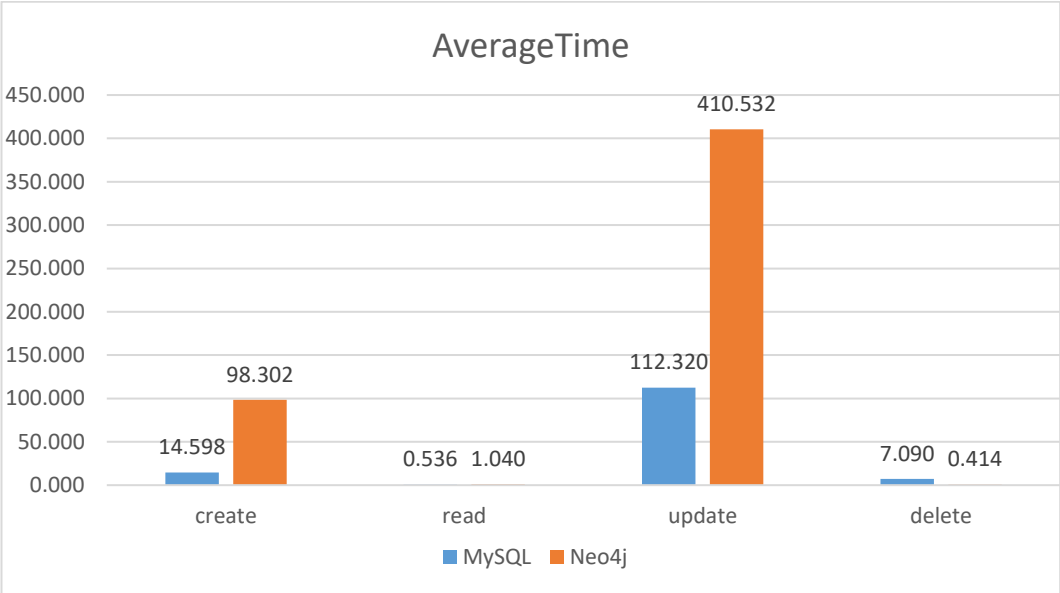
$$averageTime = \frac{1.78}{0.738}$$
$$= 0.414$$

4.4 Hasil performa

Berdasarkan hasil pengujian penyimpanan (*CRUD*) *dataset* ke dalam *database*, didapatkan rata-rata waktu sesuai dengan Tabel 4.1 dan Gambar 4.5

Tabel 4.7. Average.Time

Database	create	read	update	delete
MySQL	14.598	0.536	112.320	7.090
Neo4j	98.302	1.040	410.532	0.414



Gambar 4.9 Grafik Average.Time

Dari Tabel 4.7 dan Gambar 4.5 tersebut diperoleh hasil pengukuran bahwa:

1. Pengujian *Create*, Mysql memiliki performa yang lebih cepat dibandingkan dengan Neo4j. Neo4j memiliki performa yang paling rendah pada operasi *create* menyimpan data**.** Yaitu MySQL dengan waktu rata–rata 14.598 detik. Neo4j memiliki performa yaitu dengan waktu rata-rata 98.302 detik.
2. Pengujian *Read,* Mysql memiliki performa tidak jauh berbeda dibandingkan dengan Neo4j pada operasi *Read* menampilkan data**.** Yaitu MySQL dengan waktu rata–rata 0.536 detik. Neo4j memiliki performa dengan waktu rata-rata 1.040 detik.
3. Pengujian *Update*, Mysql memiliki performa yang cepat dibandingkan dengan Neo4j dikarenakan dalam pengujian diMysql melakukan *update* dengan data yang bayak sedangkan diNeo4j melakukan pengujian *update* dengan menggunakan data satu – satu sehingga memakan waktu.

   Pengerjaan untuk update di Neo4j melakukan waktu 4 jam dalam 1 hari 4 jam x 30 hari = 120 jam, 120 jam di bulatkan menjadi hari, yaitu 120 jam = 5 hari total time di kali dengan total hari, yaitu 87629.89 x 5 hari = 438149.43 lalu di bagi dengan *datalength* 1067.271 menjadi 410.532 pada operasi *create* menyimpan data MySQL dengan waktu rata–rata 112.320 detik. Dan Neo4j memiliki performa yaitu dengan waktu rata-rata 410.532 detik.

4. Pengujian *Delete*, Mysql memiliki performa yang lambat dibandingkan dengan Neo4j. Neo4j memiliki performa yang cepat pada operasi *Delete* dalam menghapus data**.** yaitu MySQL dengan waktu rata–rata 7.090 detik. Neo4j memiliki performa cepat yaitu dengan waktu rata-rata 0.414 detik.

BAB V
KESIMPULAN DAN SARAN

5.1.1 Kesimpulan

Berdasarkan Analisis Kinerja Pengujian performa terhadap *relational database* (MySQL) dan *graph database* (Neo4j) memiki keunggulan masing-masing. *Relational database* (MySQL) memiliki performa dan kecepatan yang lebih baik dalam *syntax create* yaitu dengan waktu rata–rata 14.598 detik dan *syntax read* 0.536 detik, *syntax update* yaitu dengan waktu rata-rata 112.320 detik sedangkan graph database (Neo4j) unggul di *syntax delete* 0.414 detik.

5.2 Saran

Untuk Penelitian selanjutnya sebaiknya dilakukan dengan melakukan pengujian performa lebih lanjut pada operasi lain seperti :

1. *Throughput*. Kemampuan keseluruhan komputer dalam memproses data, yang merupakan kombinasi kecepatan IO, kecepatan CPU, kemampuan paralel dan efisiensi sistem operasi dan perangkat lunak sistem.
2. Sumber Daya (*Resources*). Perangkat keras dan perangkat lunak, termasuk memori, kecepatan *disk, cache controller* dan sebagainya.
3. *Memory*. Jumlah total memori yang dibutuhkan untuk menyelesaikan eksekusi. *Create, read, update, delete* Nilai ini diambil setelah berakhirnya eksekusi.

DAFTAR PUSTAKA

A. M. Bhugul, 2015 . *Comparative Study of SQL & NOSQL Databases*, Int. J. Sci. Res. Dev., vol. 3, no. 2, pp. 1496–1498.

Dayne Hammes, Hiram Medero, Harrison Mitchell. 2014. *Comparison of NoSQL and SQL Databases in the Cloud*

Elena Baralis, Andrea Dalla Valle, Paolo Garza, Claudio Rossi, Francesco Scullino 2017. *SQL versus NoSQL Databases for Geospatial Application*

Faizal Anugrah Bhaswara, Riyanarto Sarno,dan Dwi Sunaryono, 2017. Perbandingan Kemampuan *Database* NoSQL dan SQL dalam Kasus ERP Retail

Fatima Yassine, Mamoun Adel Awad, 2018. *Migrating from SQL to NOSQL Database: Practices and Analysis*

Gansen Zhao, Qiaoying Lin, Libo Li, Zijing Li, 2014. *Schema Conversion Model of SQL Database to NoSQL*

Hadyan Arif, Kemas Rahmat Saleh, Dr. Adiwijaya, 2015. Analisis dan Implementasi Graph Indexing Pada Graph Database Menggunakan Algoritma GIndex.

Hadiwinata, Muhamad Rifki, 2019. "Konsep Basis Data Relasional.".

Imam, S. P, 2016. Perancangan Data Warehouse Untuk Mendukung Kebutuhan Informasi Penjualan Dalam Pengambilan Keputusan (Studi Kasus: Sesko Mart). Diss. Universitas Widyatama.

Isjhar Kautsar. 2015. Analisis Performansi Metode *Graph Decomposition Index* pada *Graph Database*.

José Guia , Valéria Gonçalves Soares dan Jorge Bernardino, 2017. *Graph Databases: Neo4j Analysis*

Marcus Walldén, Aylin Özkan, 2016. *A graph database management system for a logistics-related service*

Mesri Silalahi, Didi Wahyudi, 2018. Perbandingan Performansi *Database* Mongodb Dan Mysql Dalam Aplikasi File Multimedia Berbasis Web

Muntasir Rahman, Petrus Mursanto, 2016 Perbandingan Kinerja Basis Data Relasionaldengan Basis Data Berorientasiobjek Studi Kasus: Aplikasi Jpetstore

Ognjen Orel, Slaven Zakošek, Mirta Baranovic. 2016. *Property Oriented Relational-To-Graph Database Conversion*.

Panji Wisnu Wirawan, Djalal Er Riyanto, 2017. Kajian Implementasi *Graph Database* pada *Rute Bus Rapid Transit.*

Partel, T., and Tarik Eltaieb. "*Relational Database vs NoSQL." Journal of Multidisciplinary Engineering Science and Technology* (JMEST) 2.4 (2015): 691-695.

Saputra, Rachman, and Sang Aji, 2013. Analisis Pengaruh Penggunaan Sistem Informasi Pelayanan Cabang Terhadap Kinerja Operasional Karyawan pada PT. Taspen (Persero) Cabang Palembang.

Setialana, Pradana, Teguh Bharata Adji, and Igi Ardiyanto, 2017. "Perbandingan *Performa Relational, Document-Oriented* dan *Graph Database* Pada Struktur Data *Directed Acyclic Graph*." *Jurnal Buana Informatika* 8.2.

ShefaliPatil, GauravVaswani, Anuradha Bhatia, 2014. *Graph Databases- An Overview*

Silalahi, Mesri. "Perbandingan Performansi *Database* Mongodb Dan Mysql Dalam Aplikasi File Multimedia Berbasis WEB." *Computer Based Information System Journal* 6.1 (2018): 63-63.

Supriya S. Pore, Swalaya B. Pawar, 2015. Comparative Study of SQL & NoSQL Databases

Surajit Medhi , Hemanta K. Baruah, 2017. *Relational Database And Graph Database: A Comparative Analysis*.

Vatika Sharma, Meenu Dave, 2012. SQL and NoSQL Databases

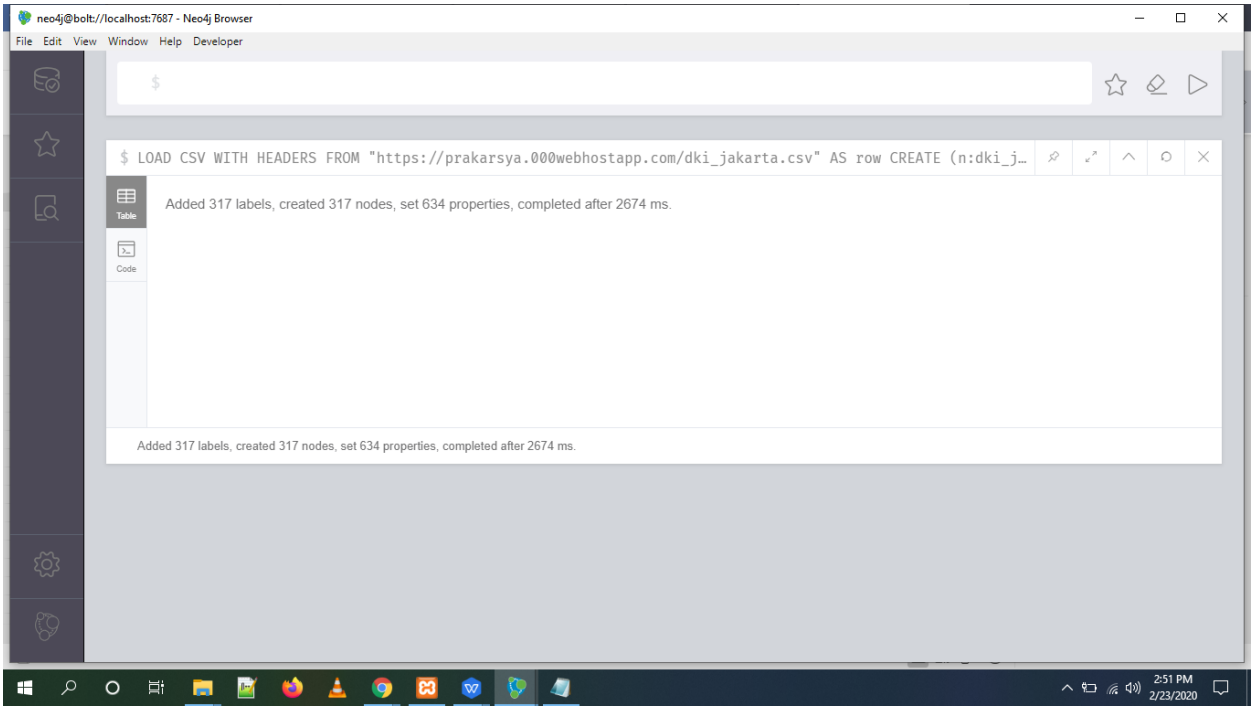
Yishan Li, Sathiamoorthy Manoharan, 2013. A performance comparison of SQL and NoSQL databases

Lampiran

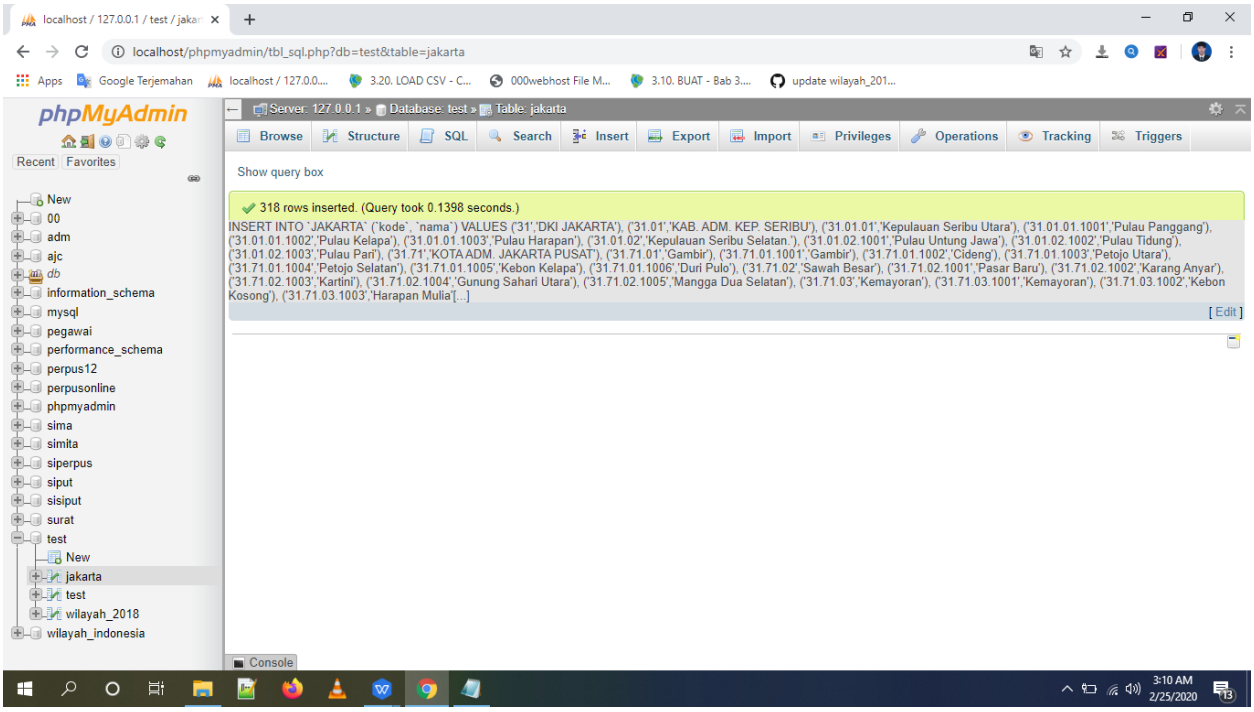
Creat MysQl vs Neo4j

Create dki Jakarta

Neo4j



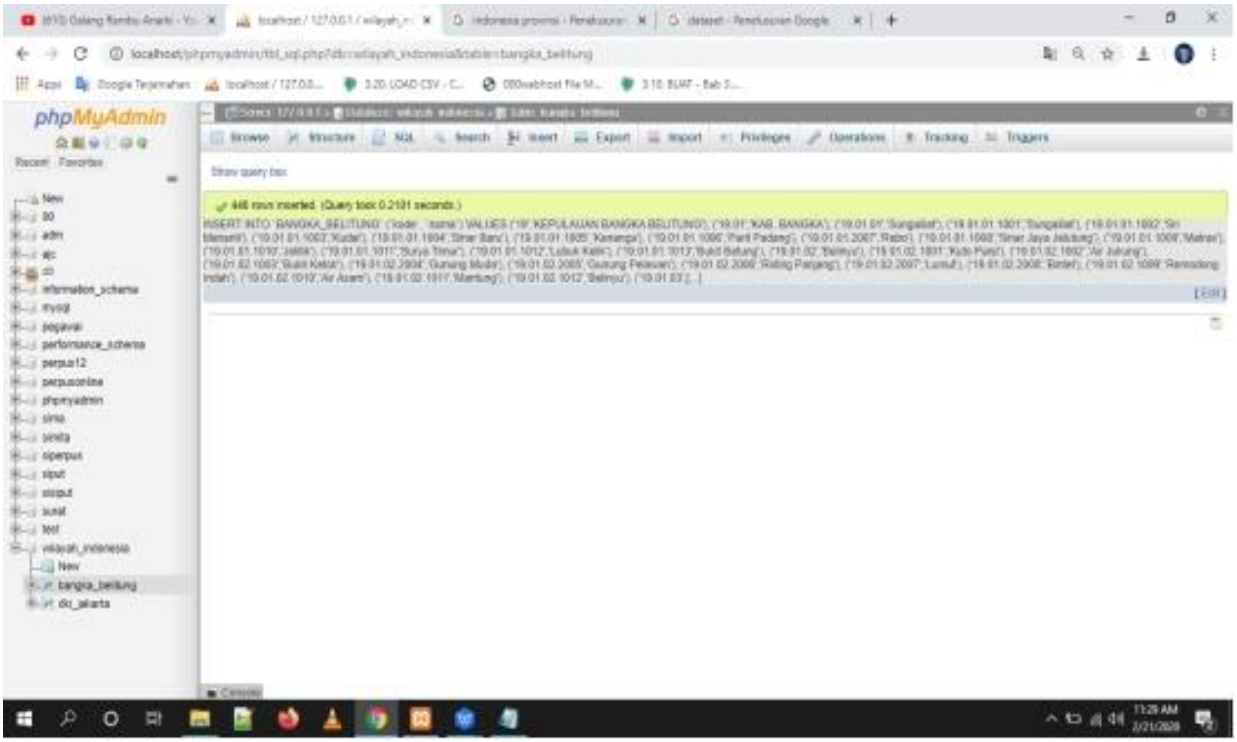
MySQL



Create bangka Belitung

Neo4j

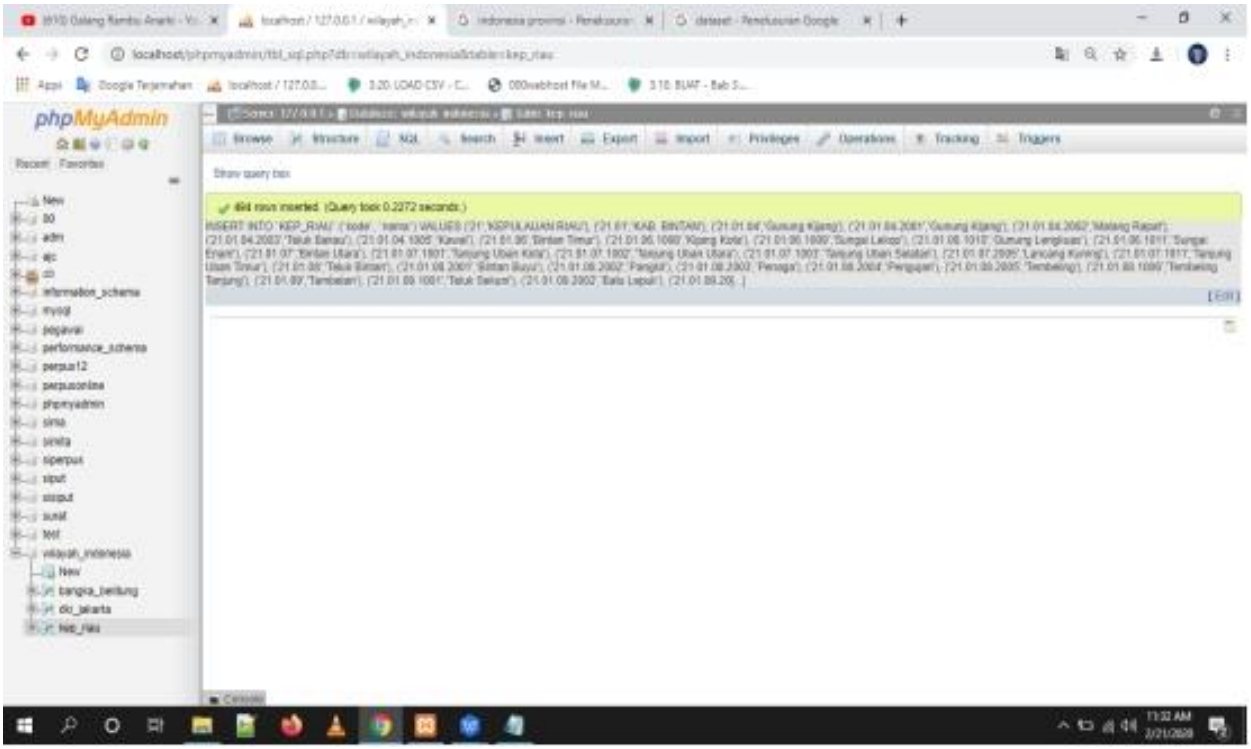
MySQL



Create riau

Neo4j

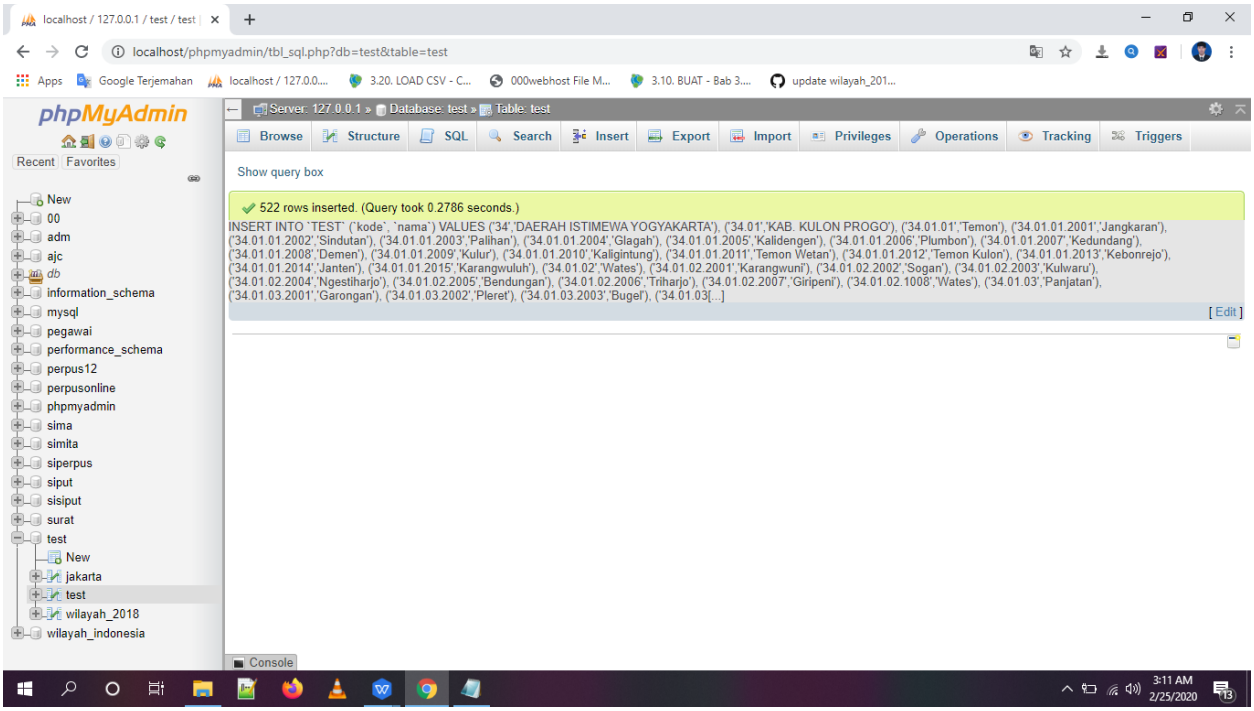
MySQL



Create Yogyakarta

Neo4j

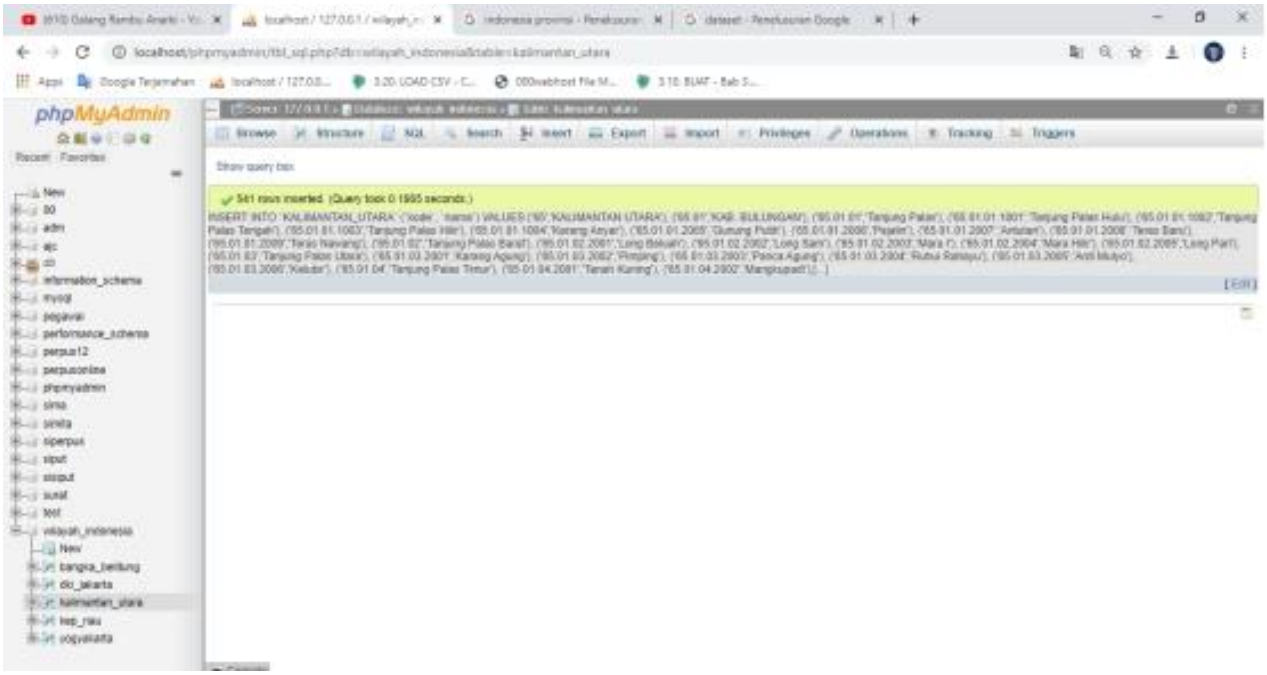
MySQL



Create Kalimantan utara

Neo4j

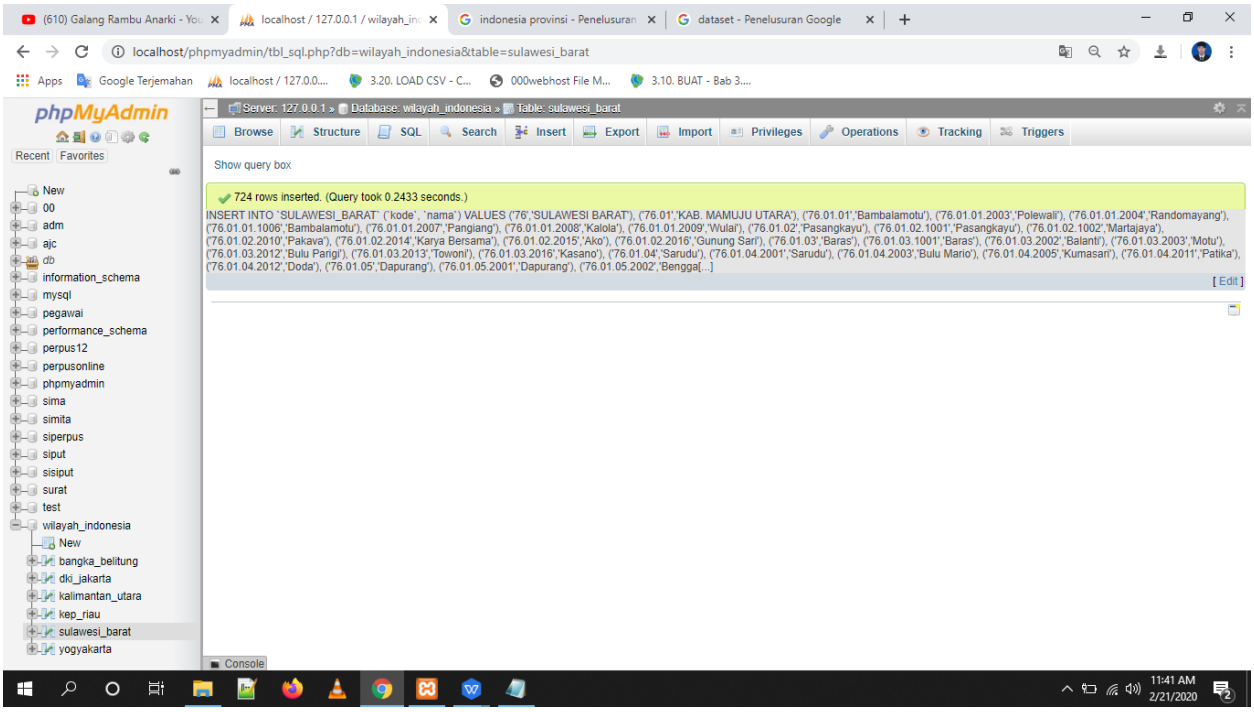
MySQL



Create Sulawesi barat

Neo4j

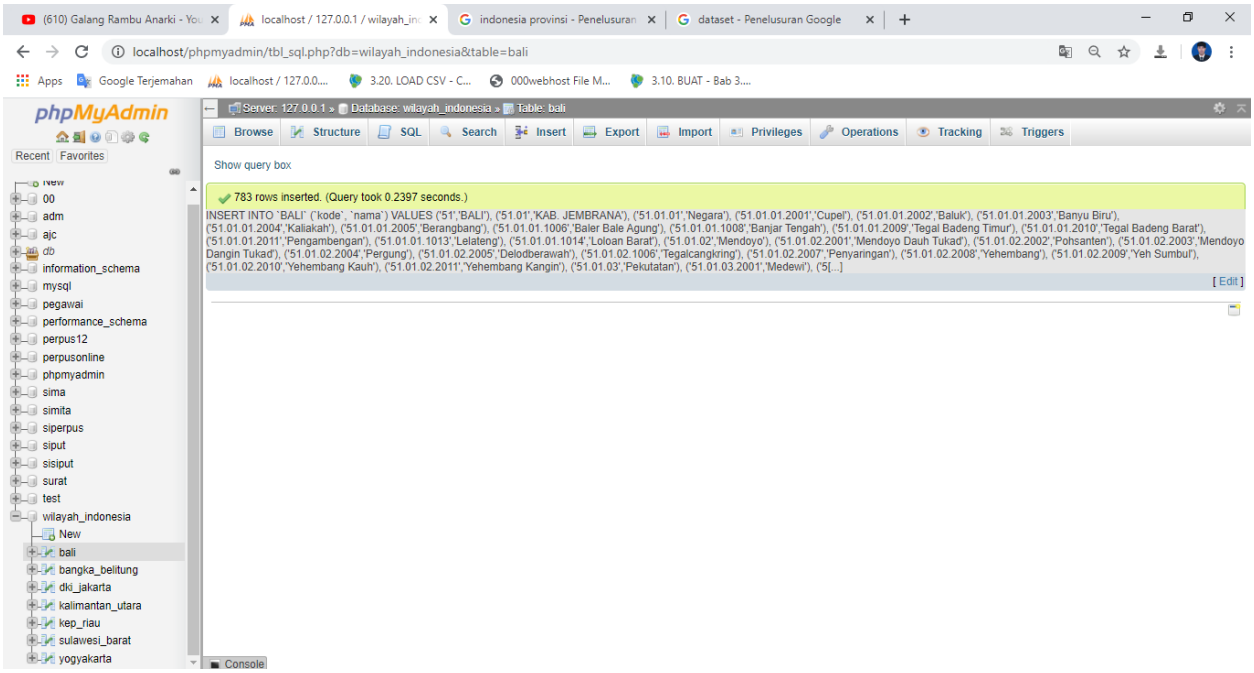
MySQL



Create bali

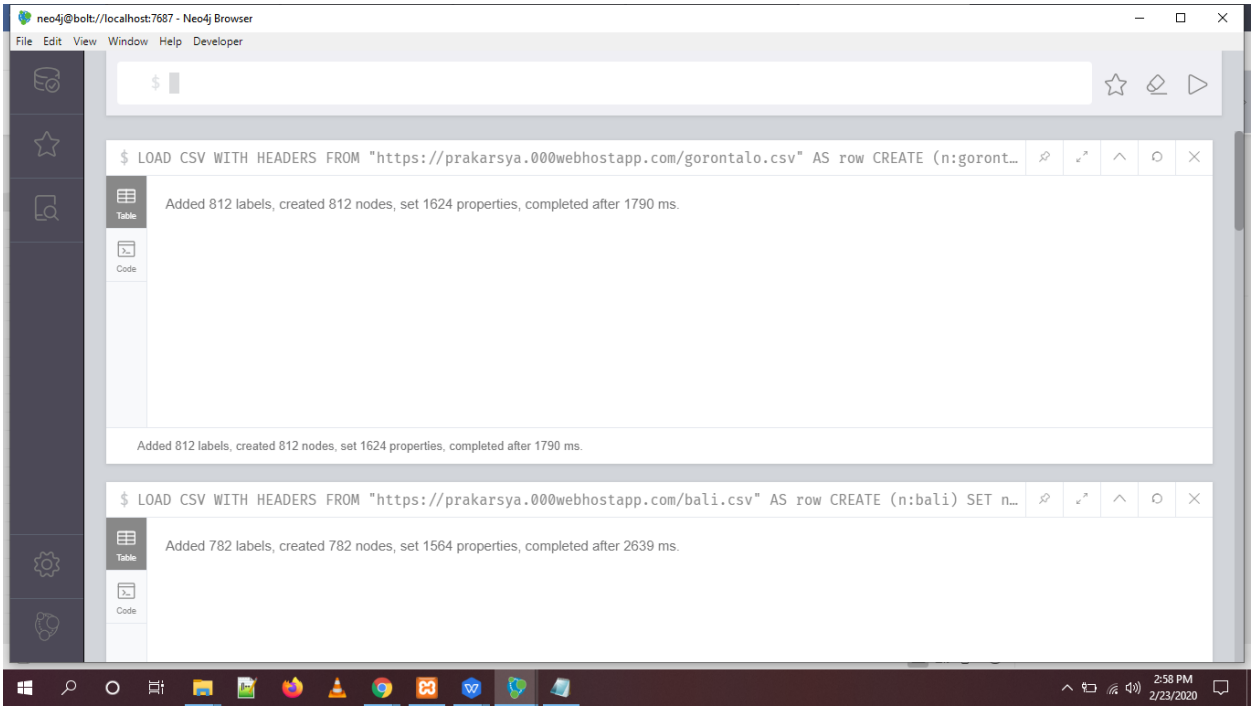
Neo4j

MySQL

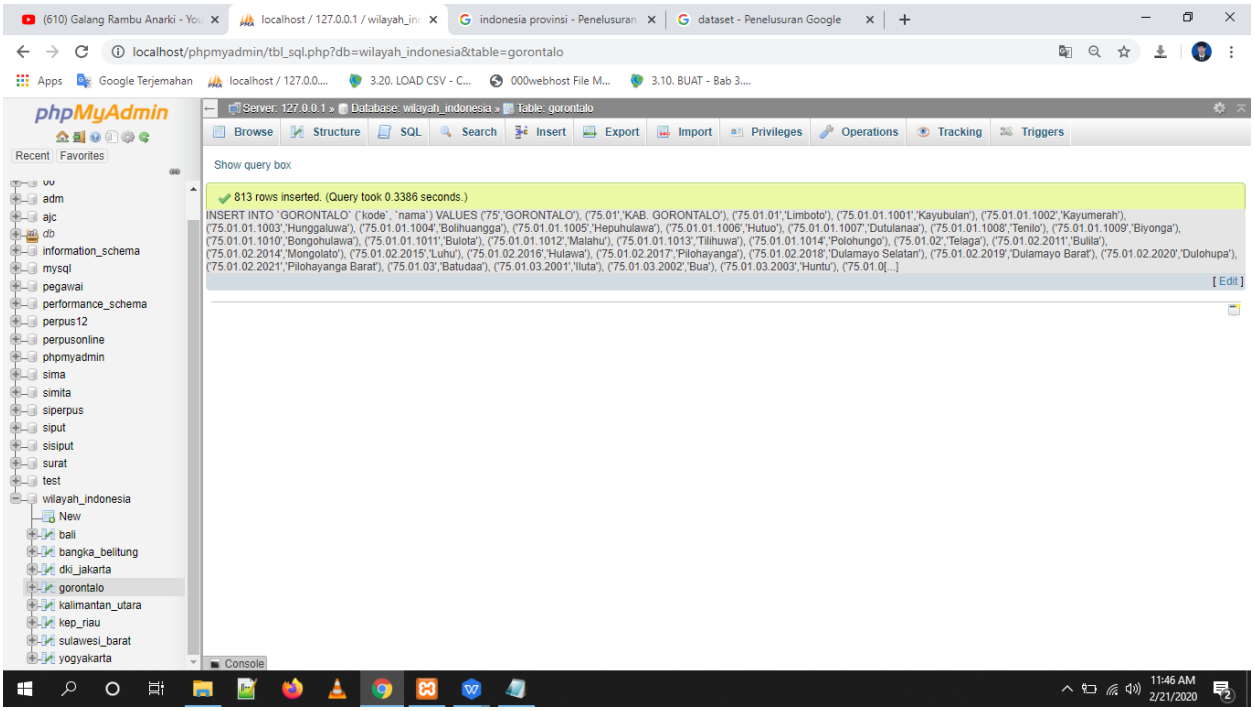


Create gorontalo

Neo4j



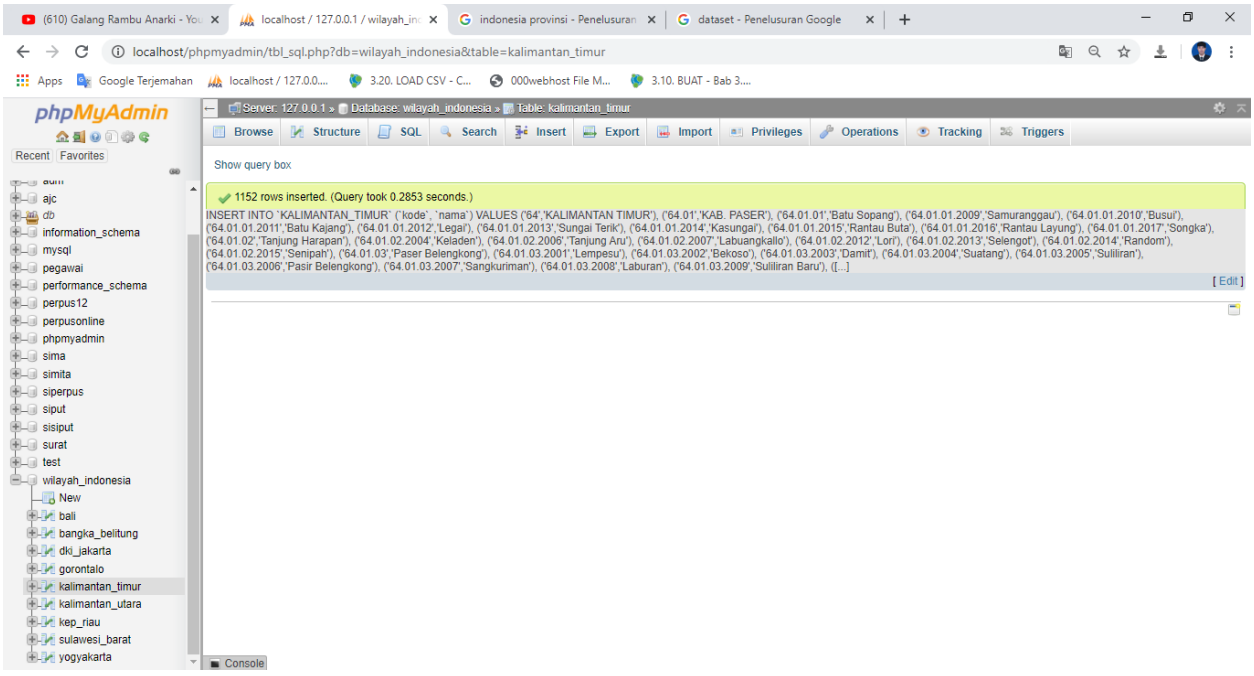
MySQL



Create Kalimantan timur

Neo4j

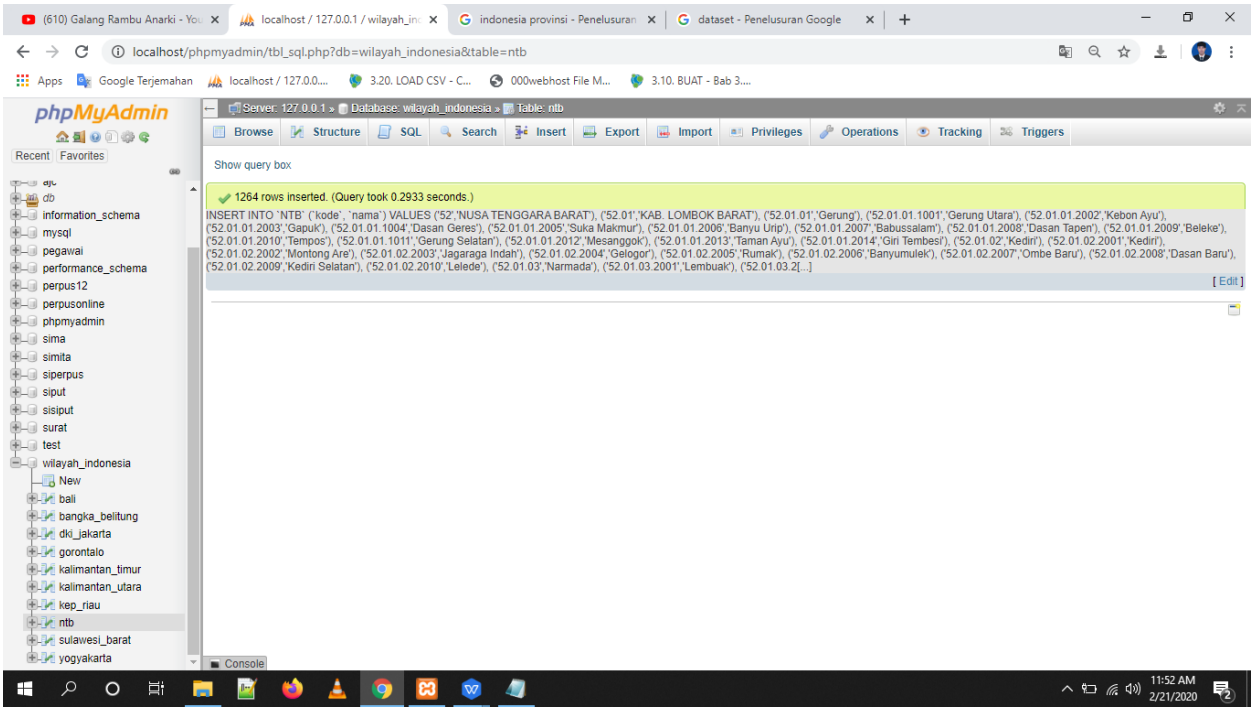
MySQL



Create Ntb

Neo4j

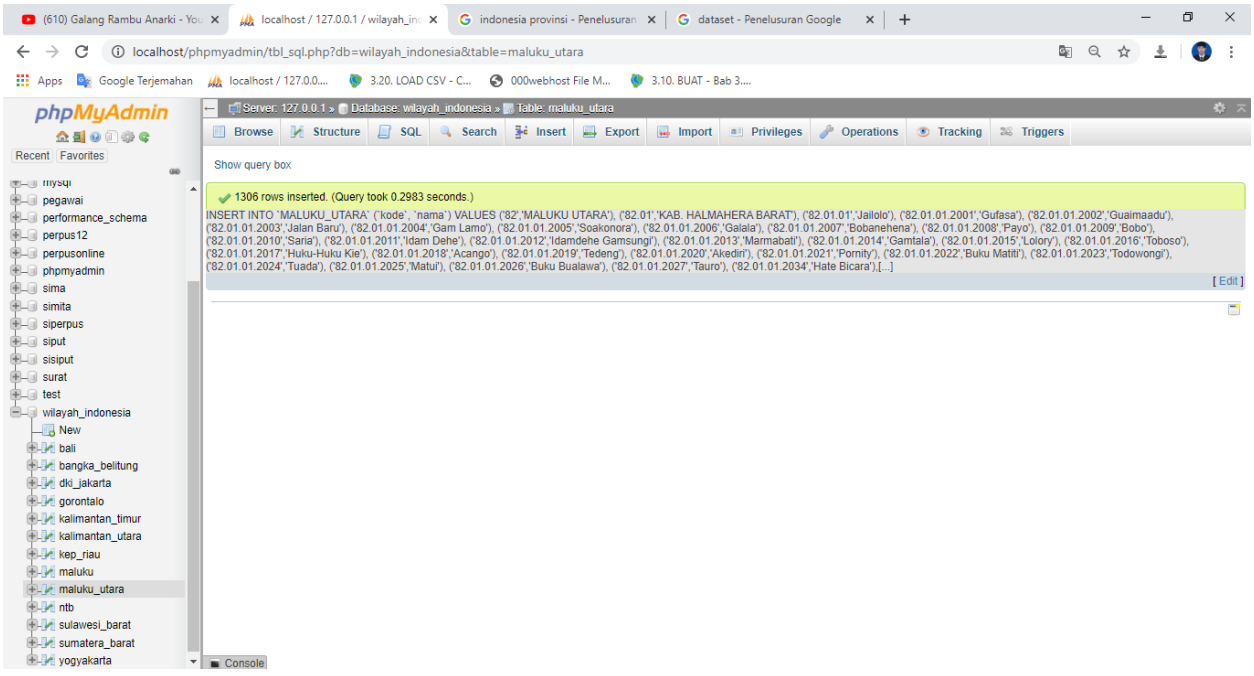
MySQL



Create Maluku utara

Neo4j

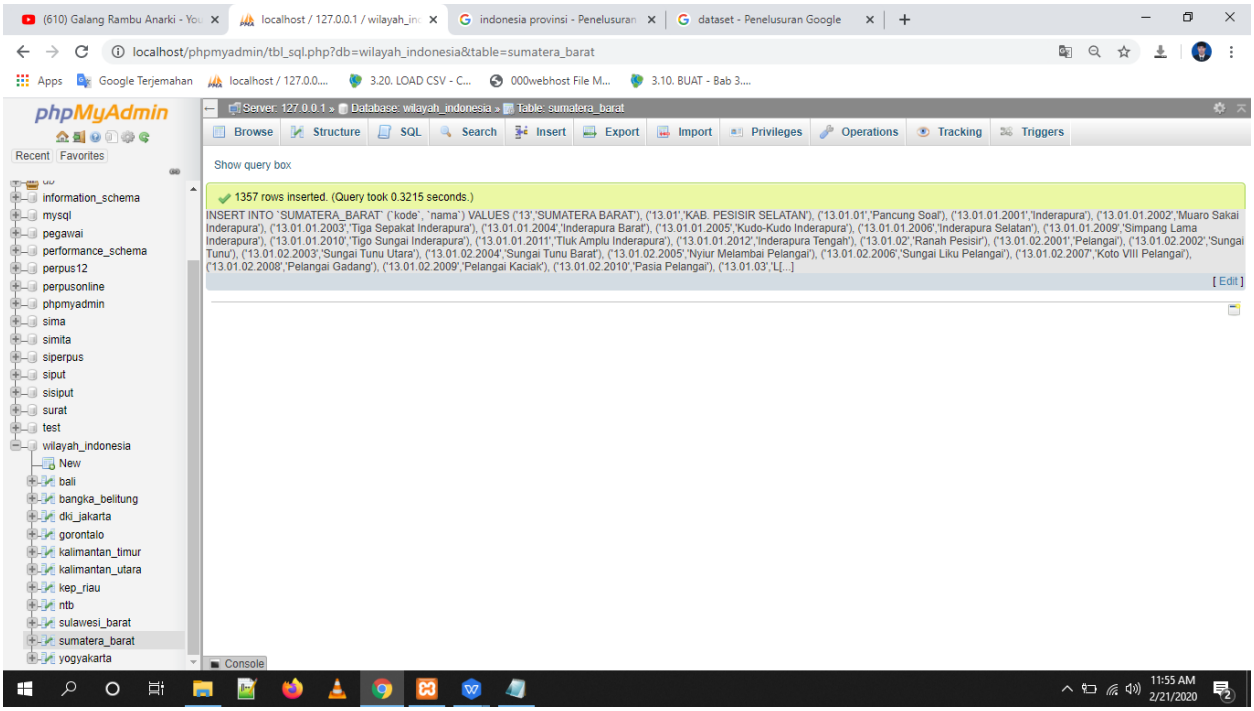
MySQL



Create sumatera barat

Neo4j

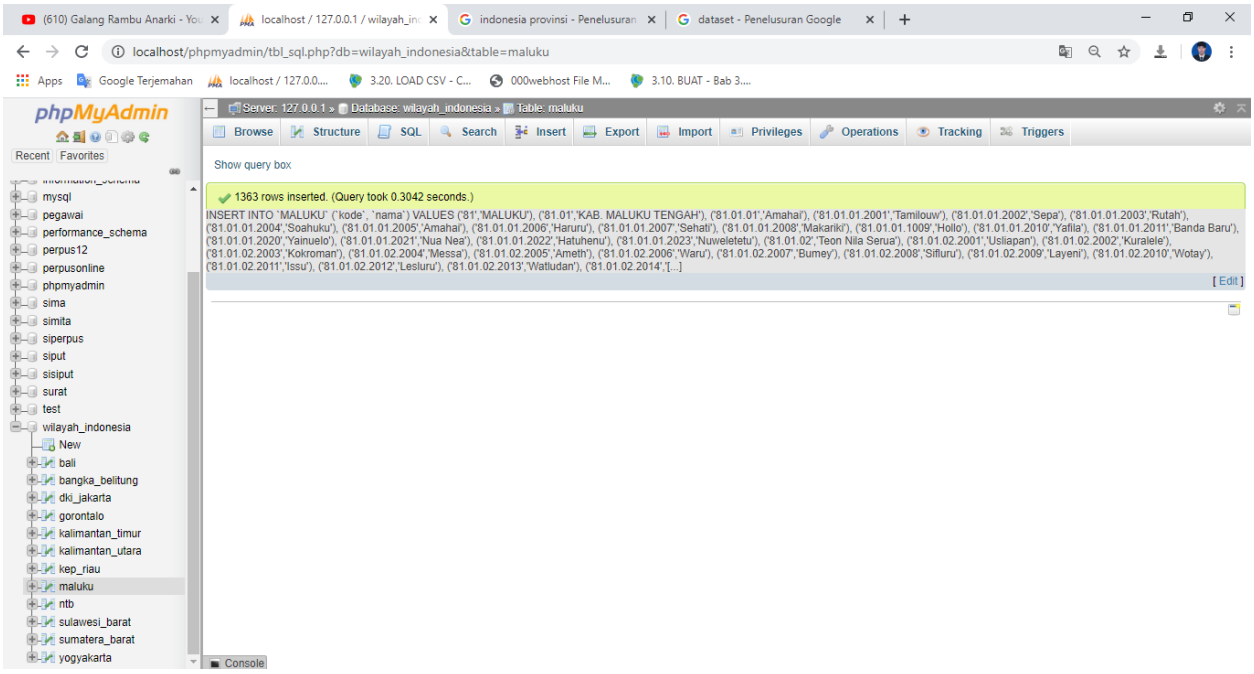
MySQL



Create Maluku

Neo4j

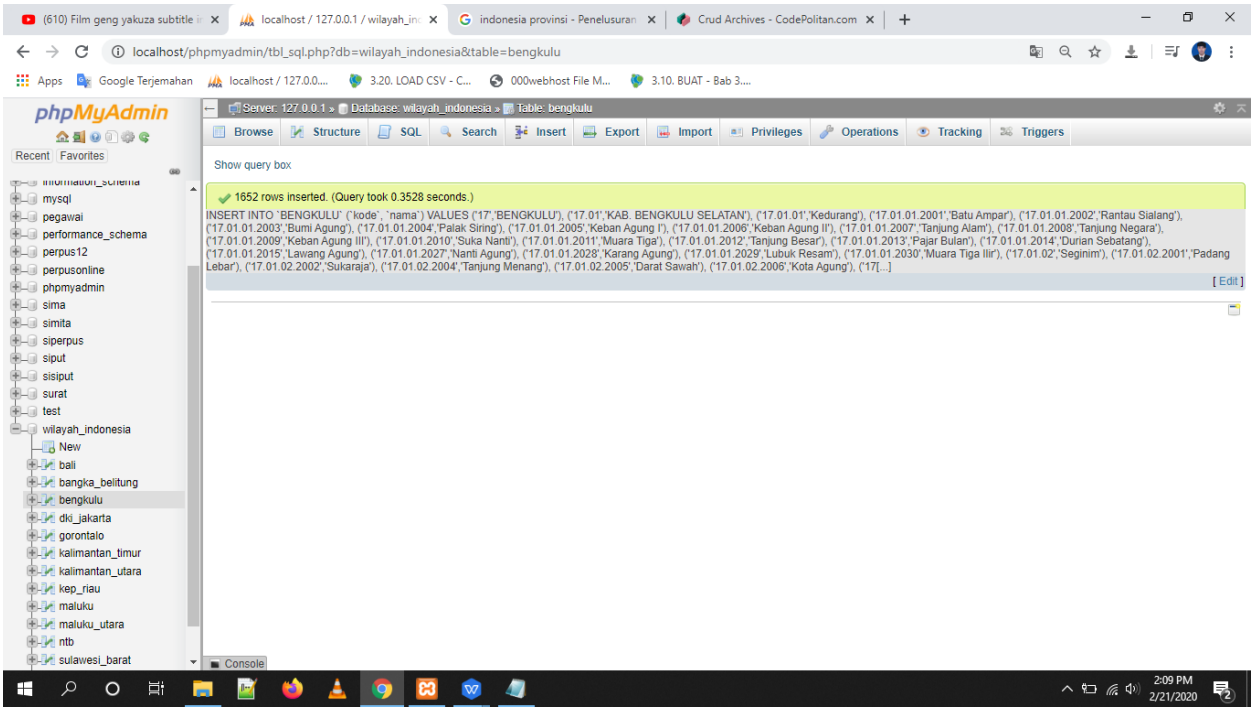
MySQL



Create Bengkulu

Neo4j

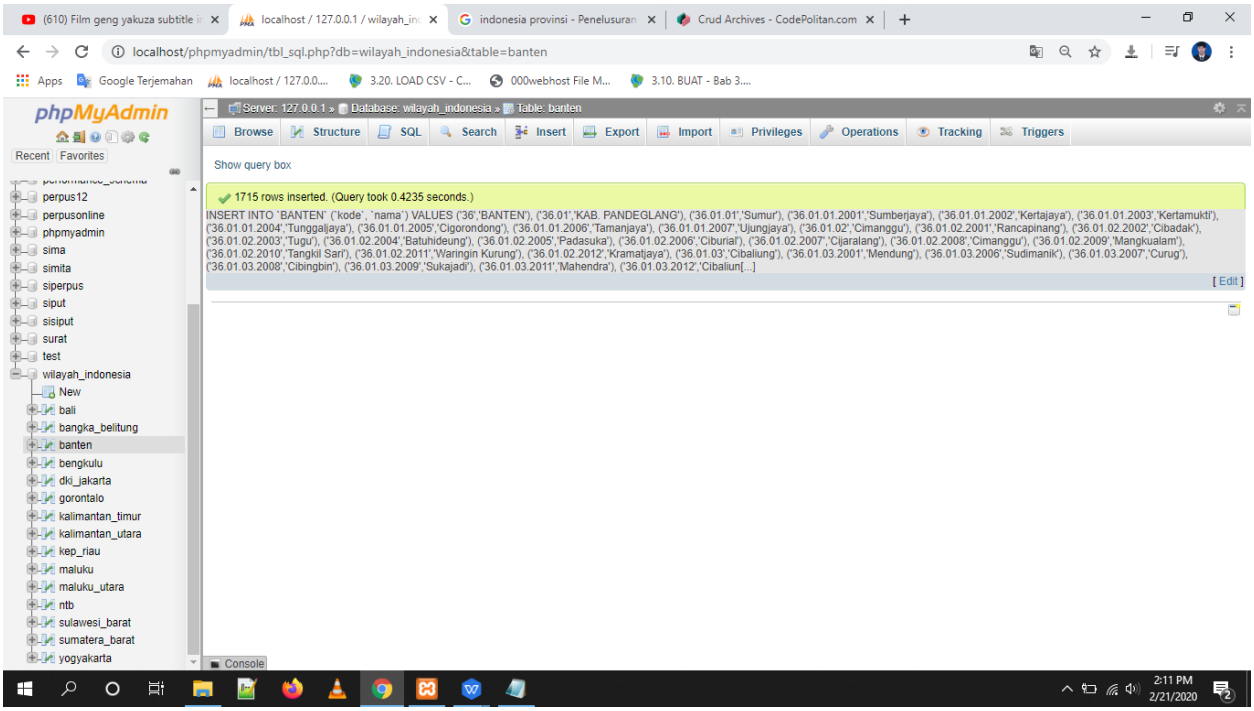
MySQL



Create banten

Neo4j

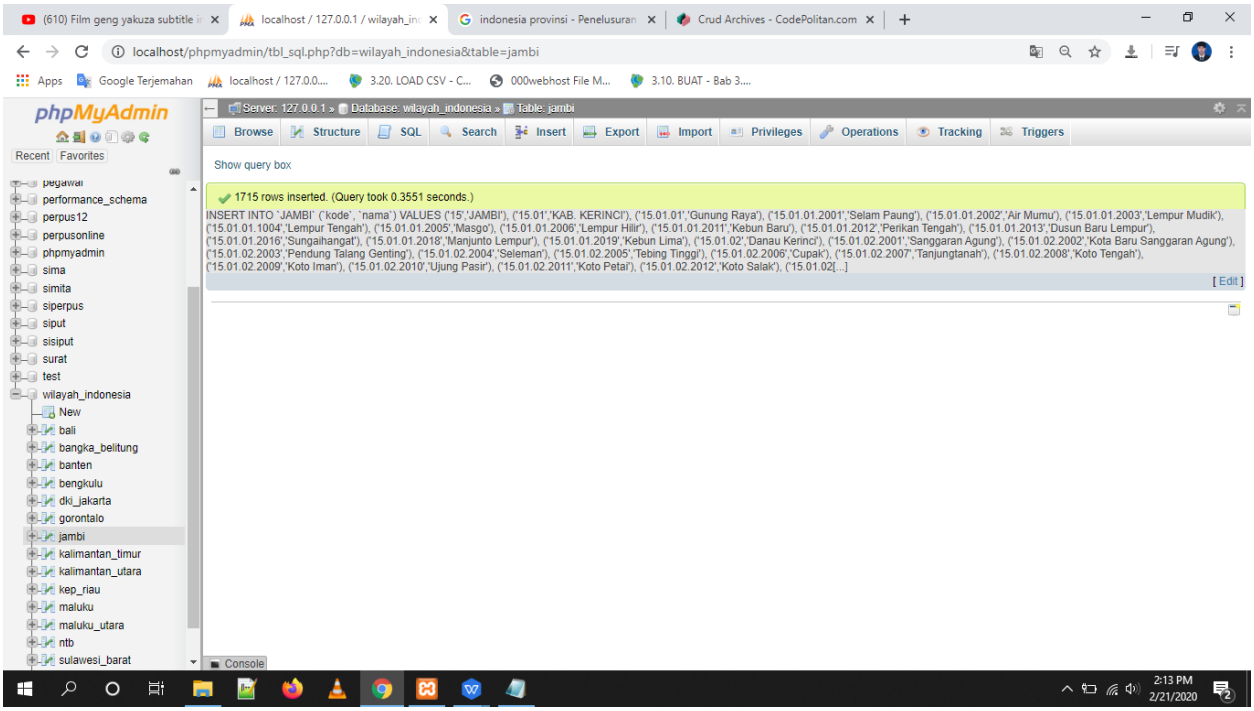
MySQL



Create jambi

Neo4j

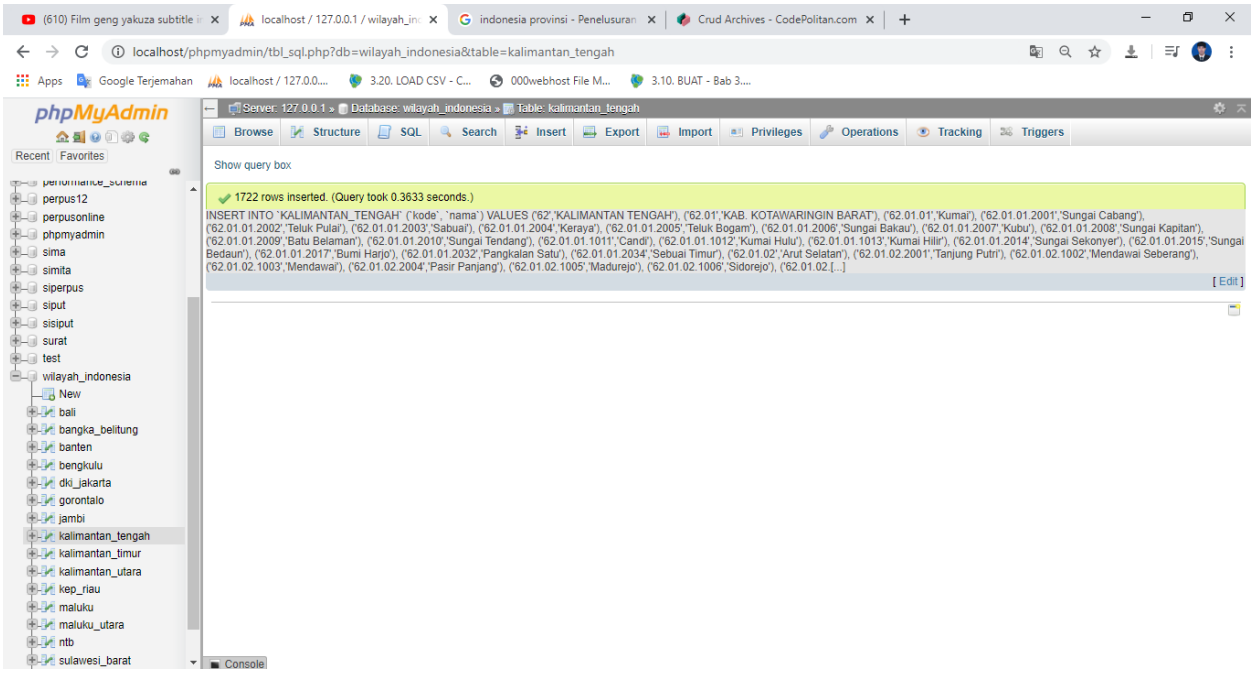
MySQL



Create Kalimantan tengah

Neo4j

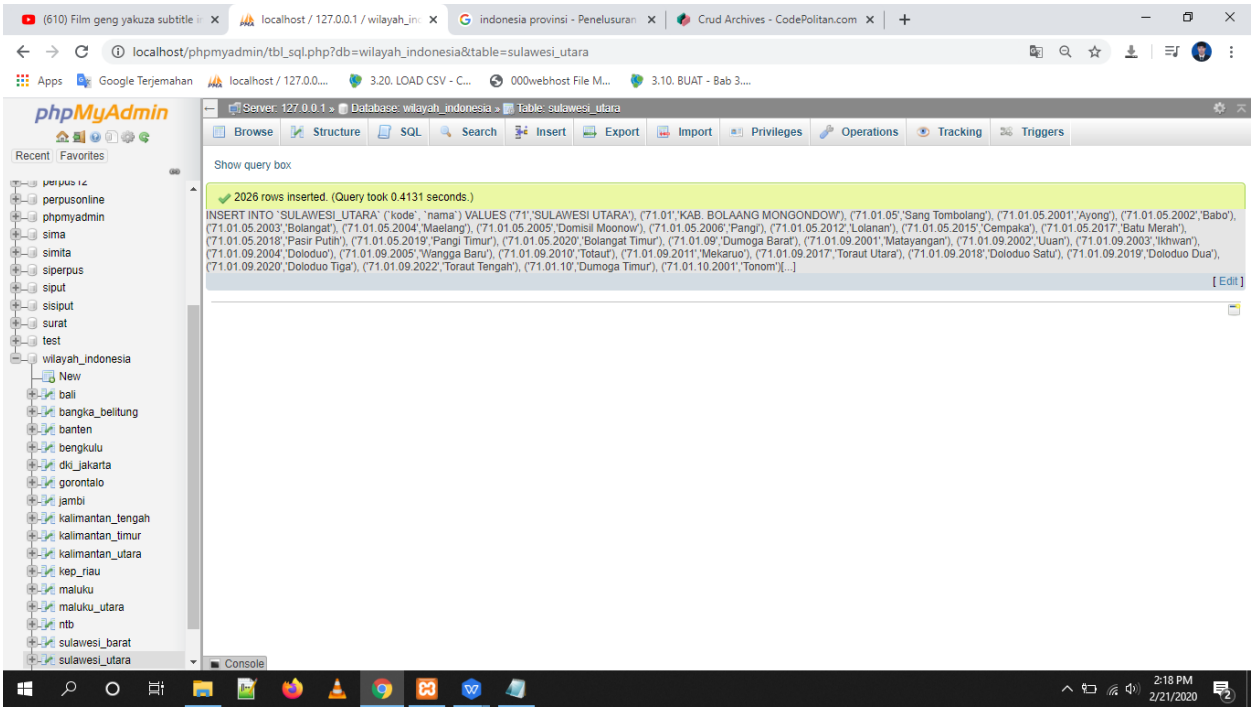
MySQL



Create Sulawesi utara

Neo4j

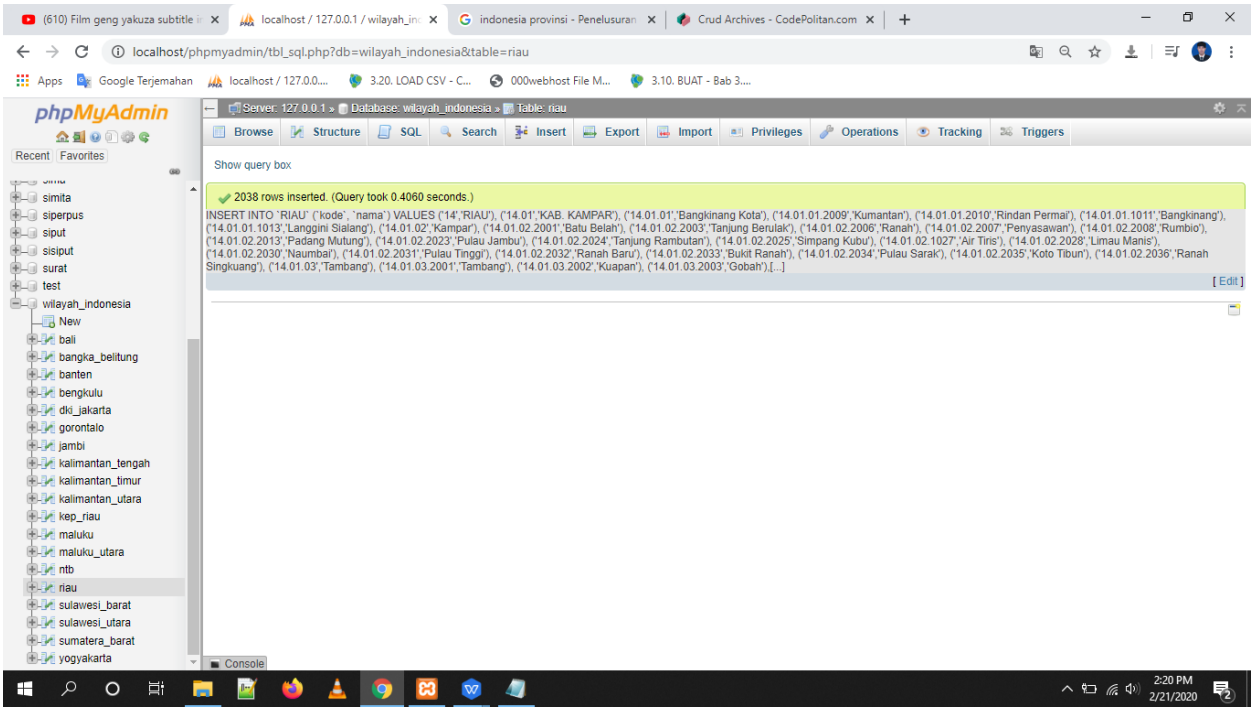
MySQL



Create riau

Neo4j

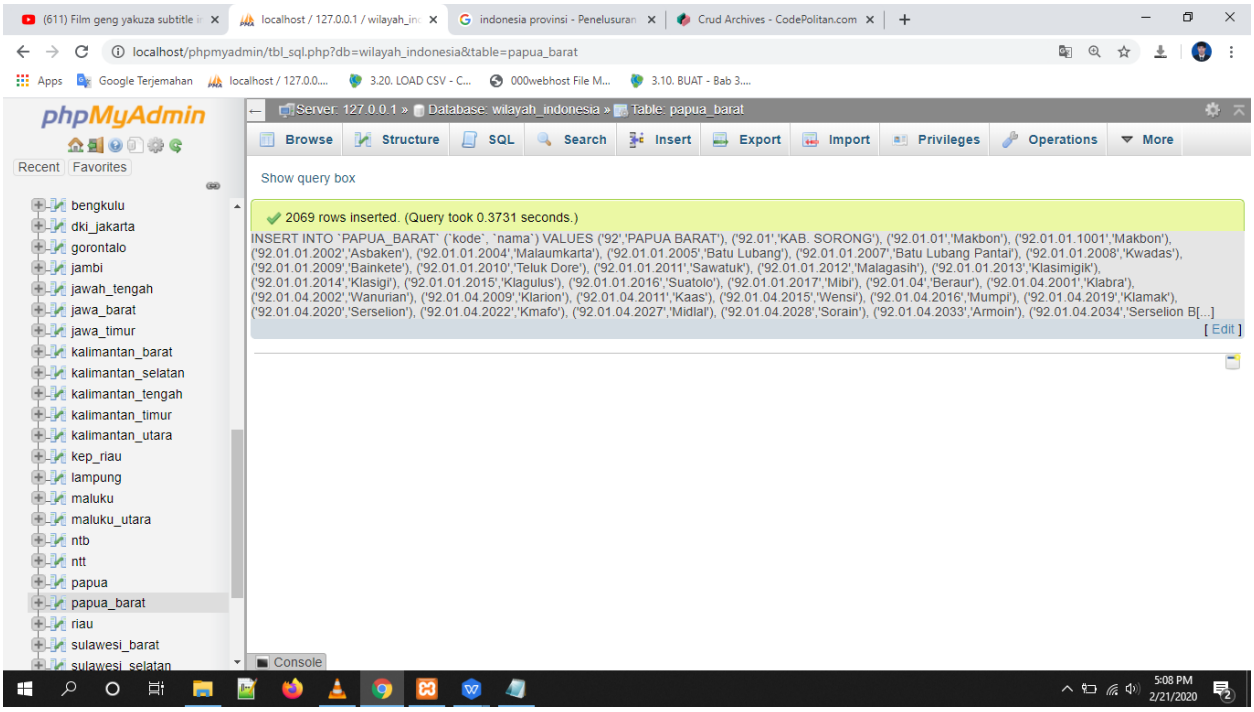
MySQL



Create papua barat

Neo4j

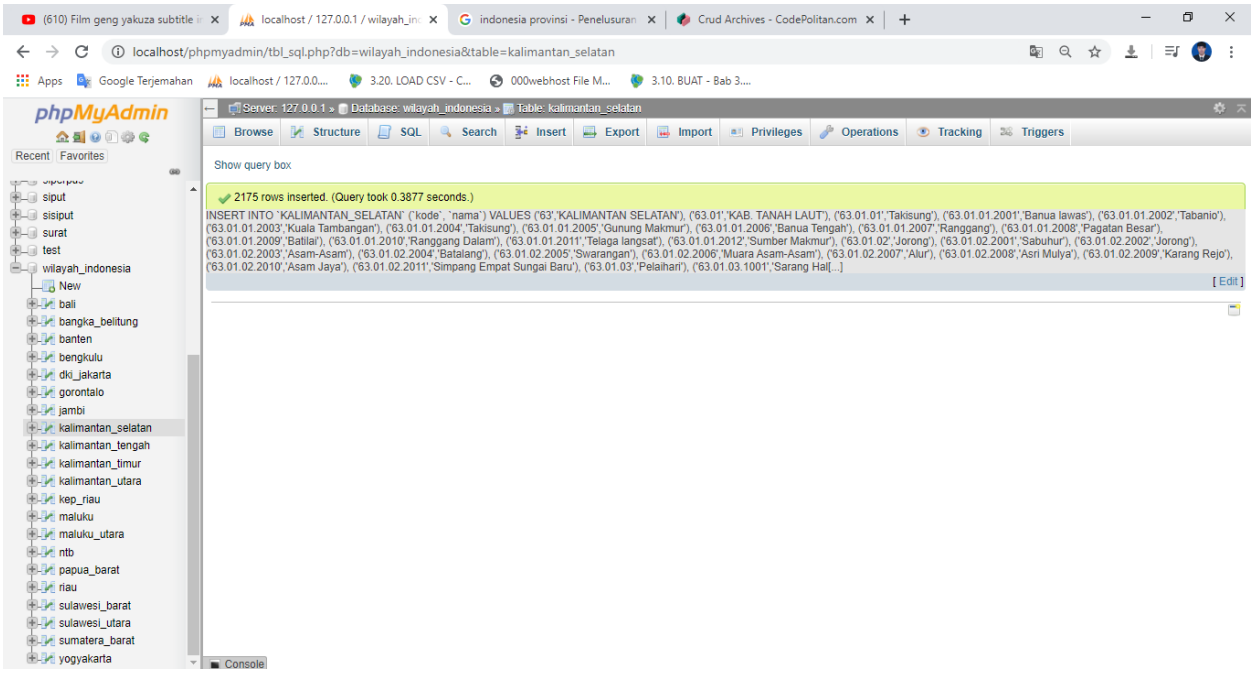
MySQL



Create Kalimantan selatan

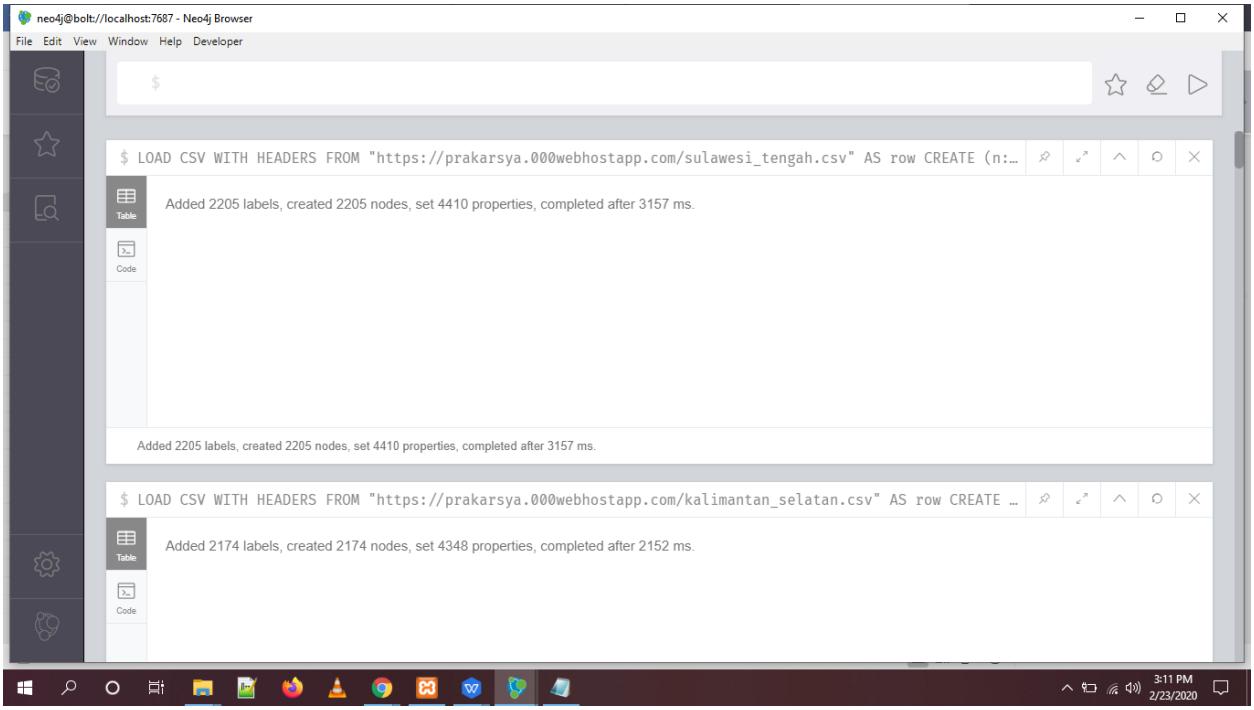
Neo4j

MySQL

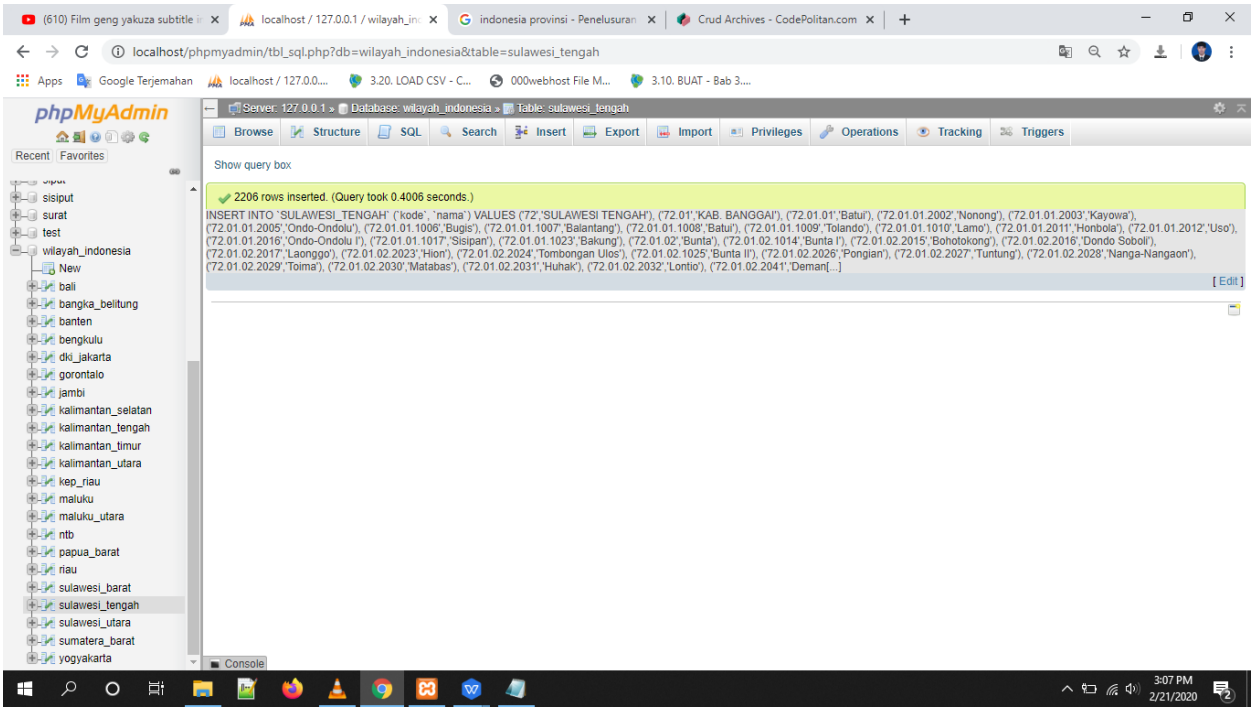


Create Sulawesi tengah

Neo4j



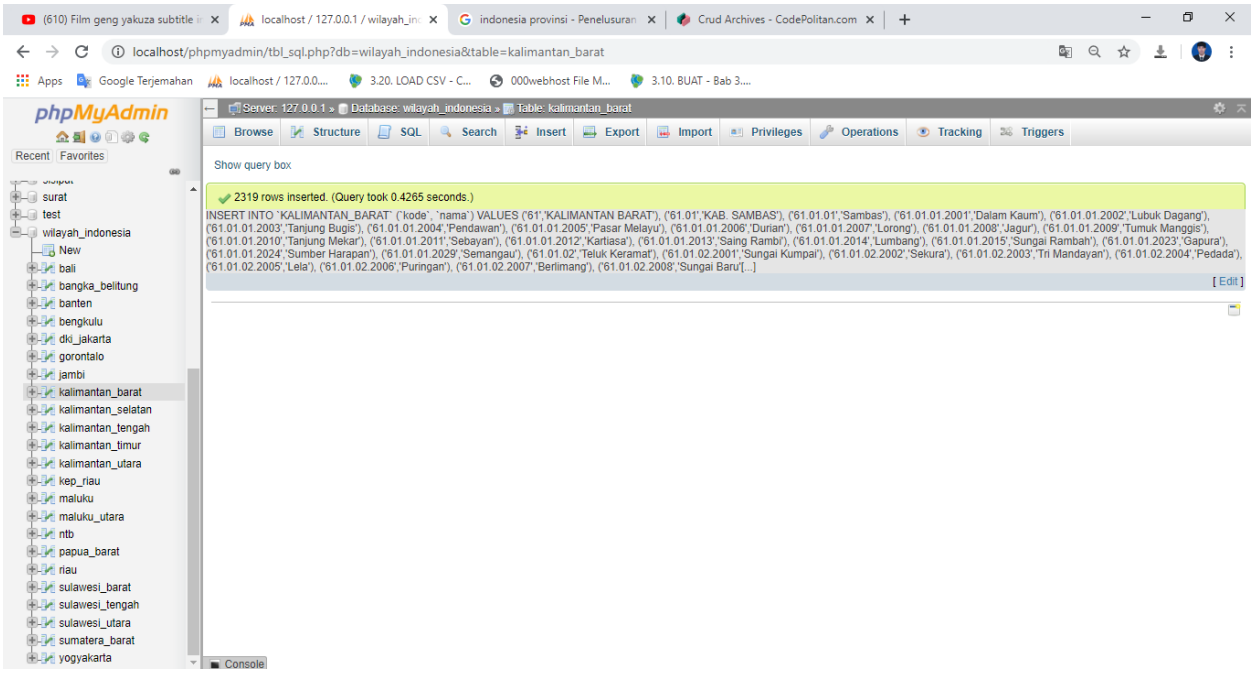
MySQL



Create Kalimantan barat

Neo4j

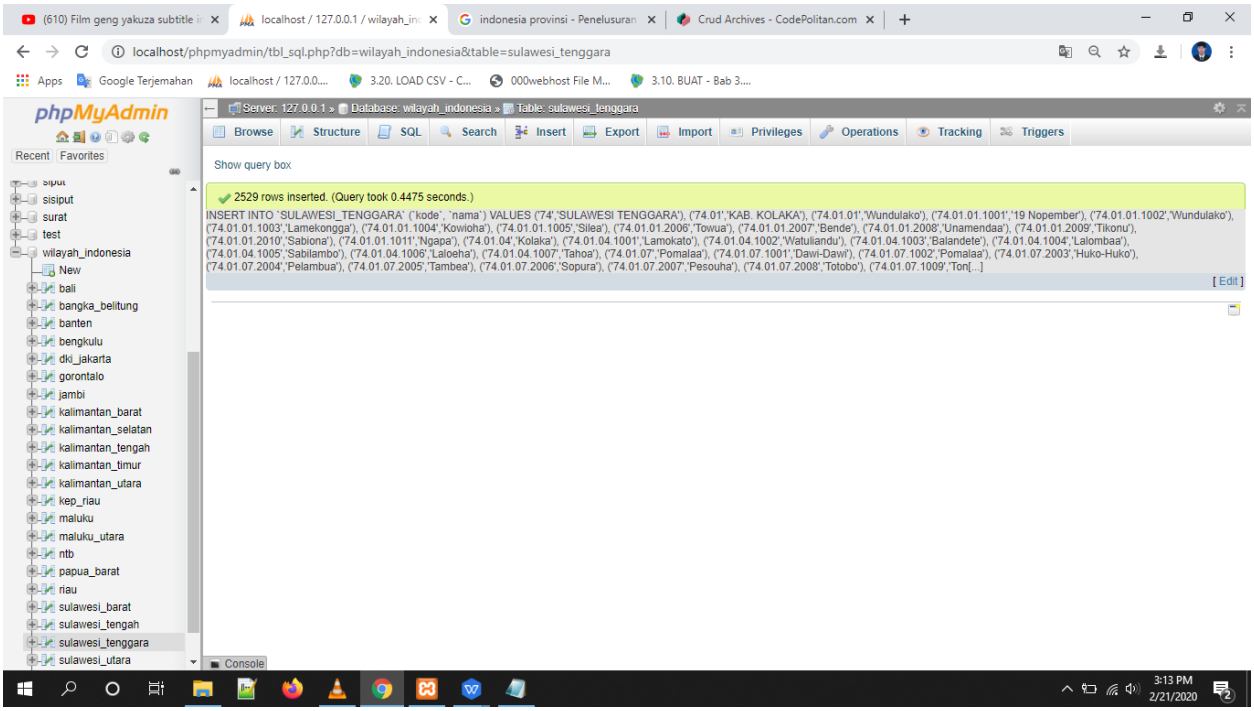
MySQL



Create Sulawesi tenggara

Neo4j

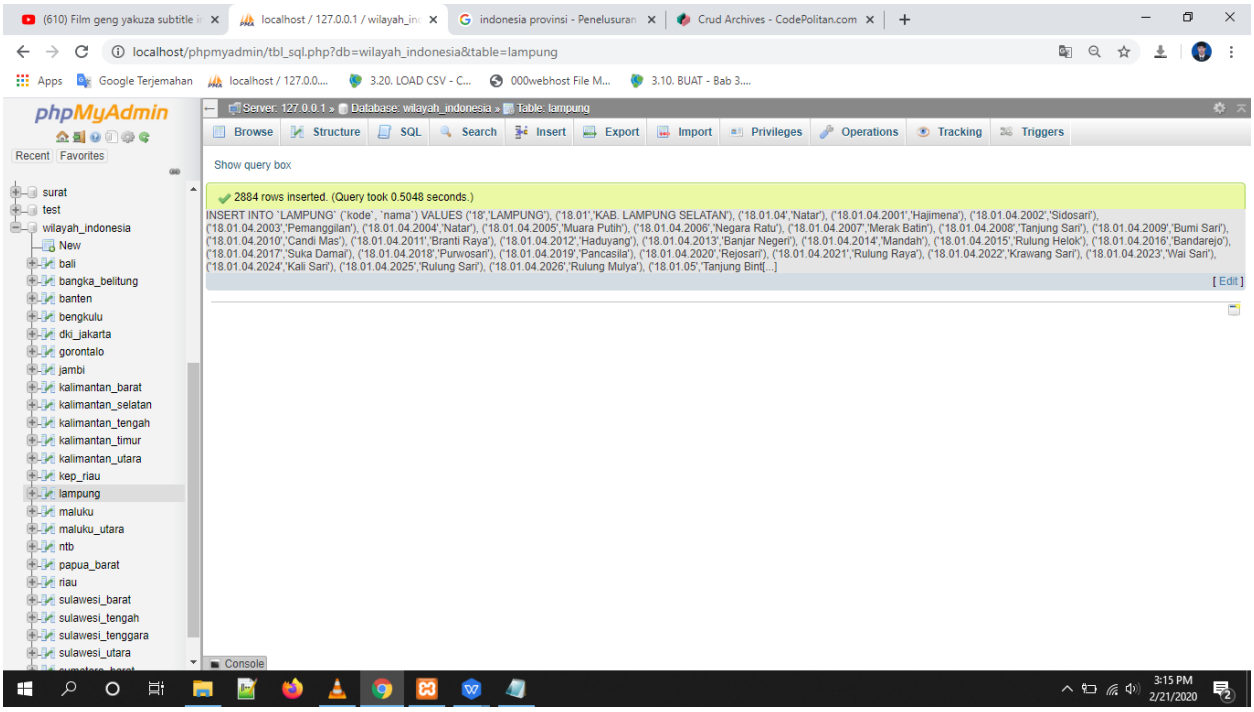
MySQL



Create lampung

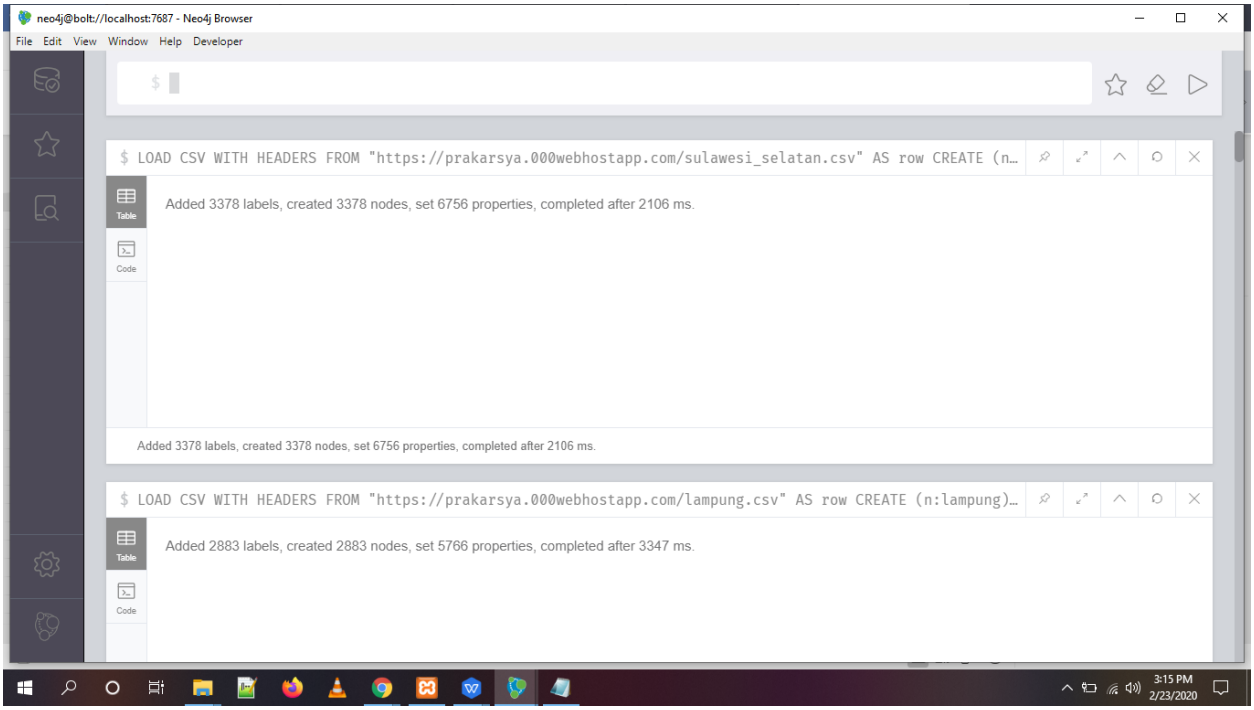
Neo4j

MySQL

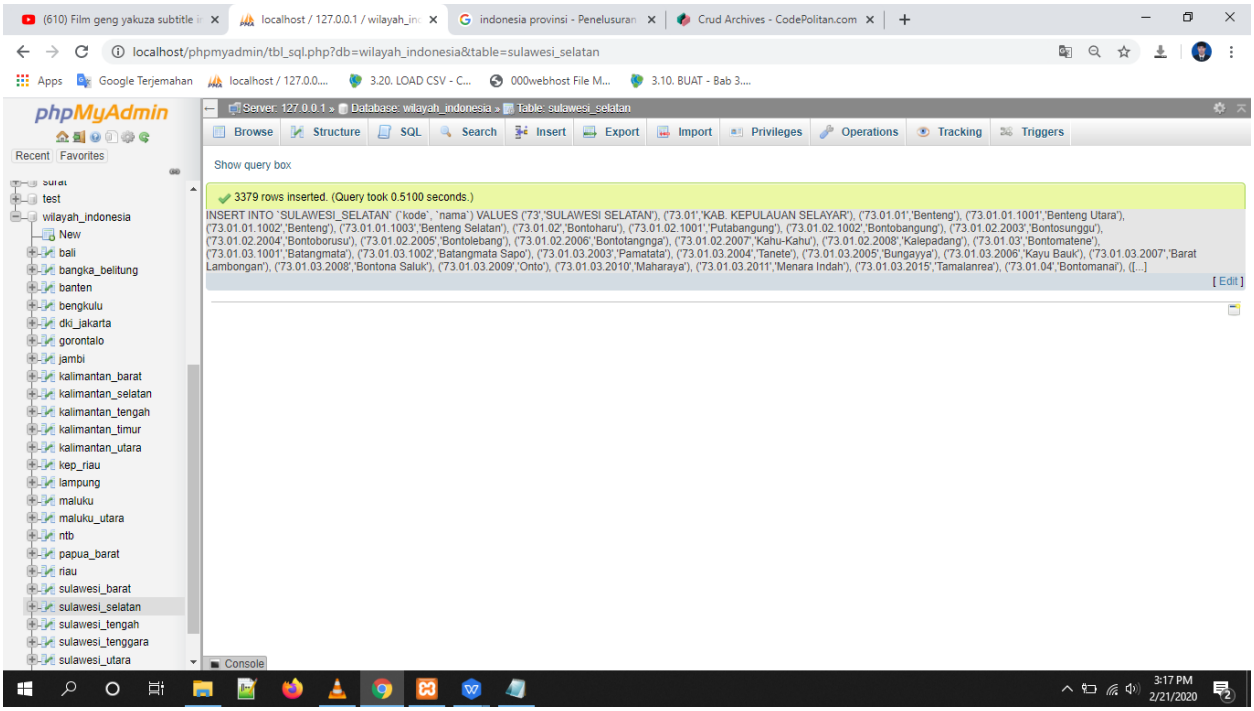


Create Sulawesi selatan

Neo4j



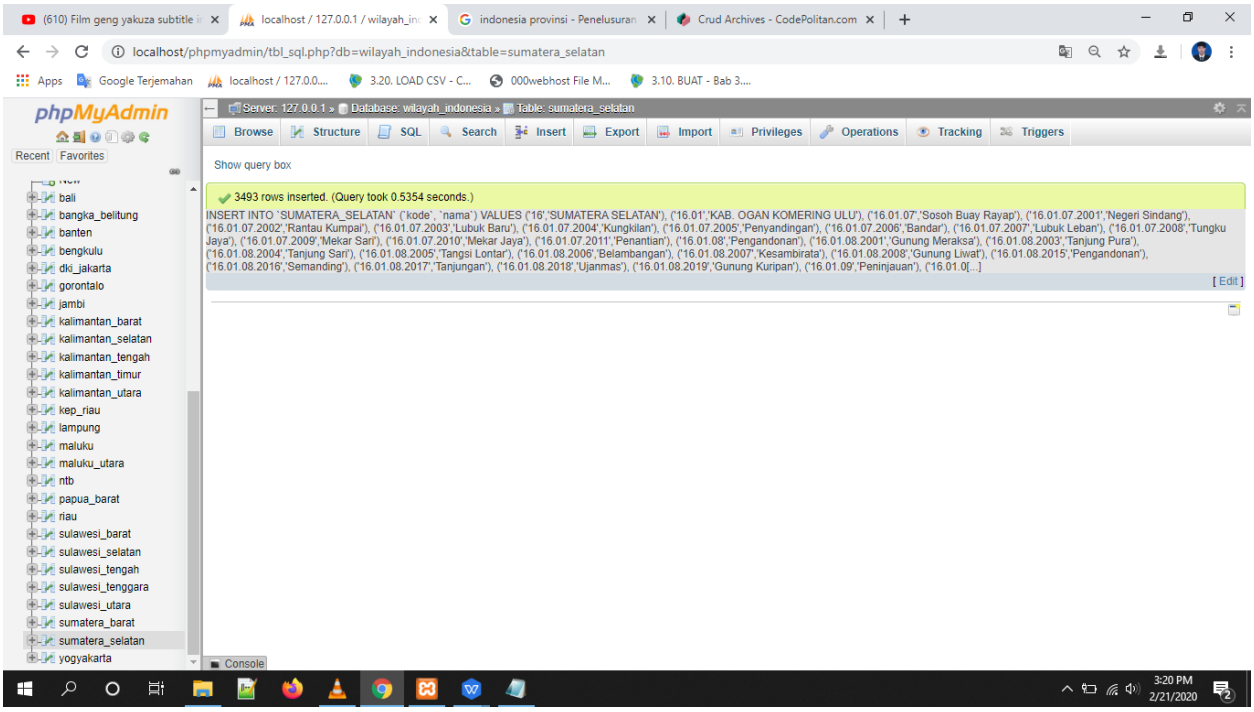
MySQL



Cerate sumatera selatan

Neo4j

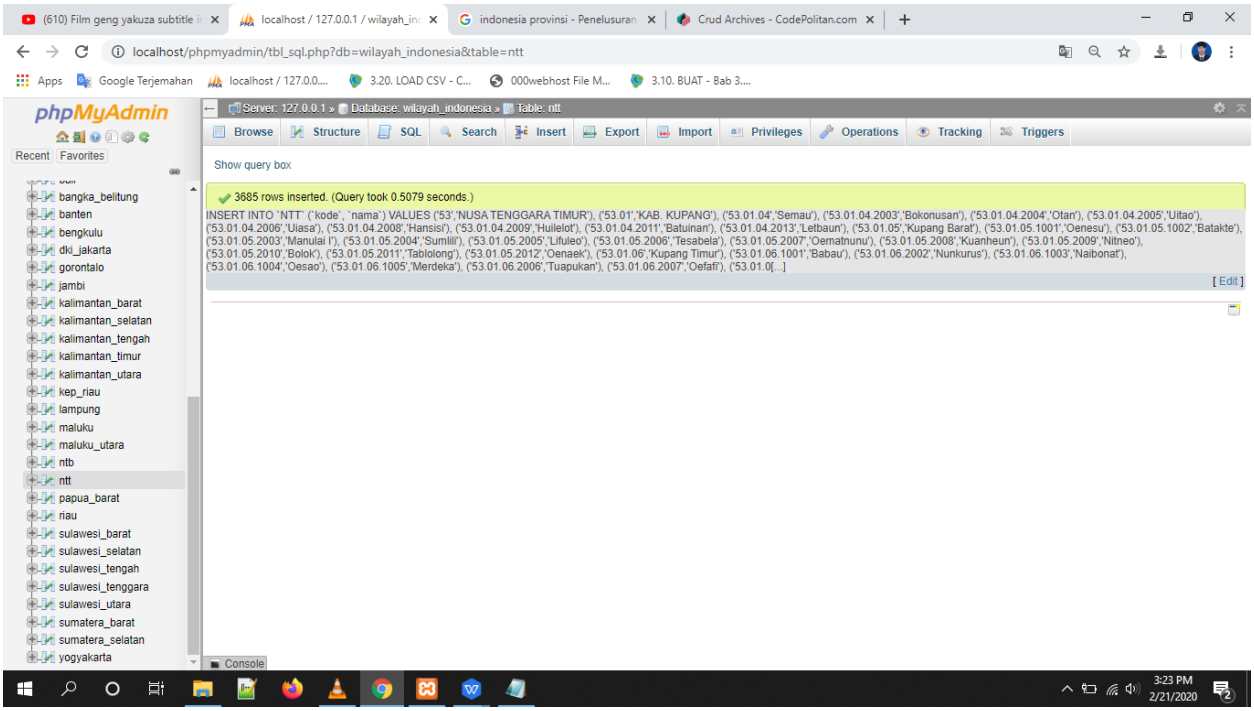
MySQL



Create ntt

Neo4j

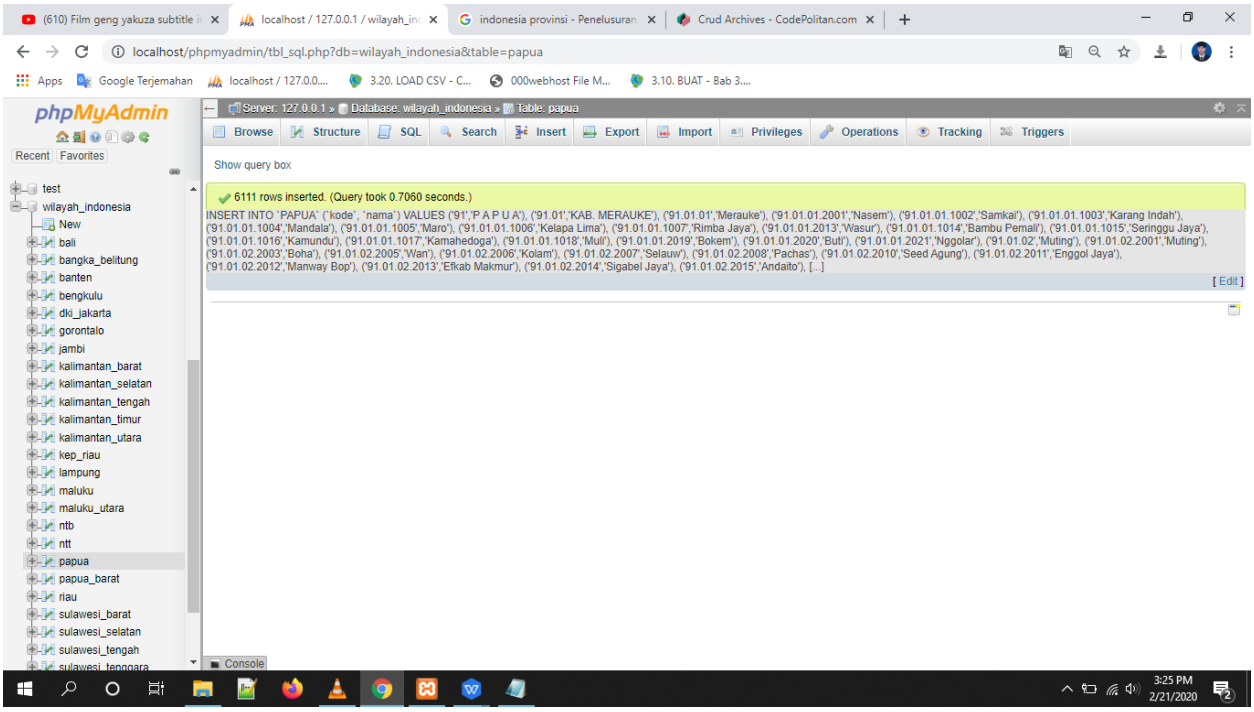
MySQL



Create papua

Neo4j

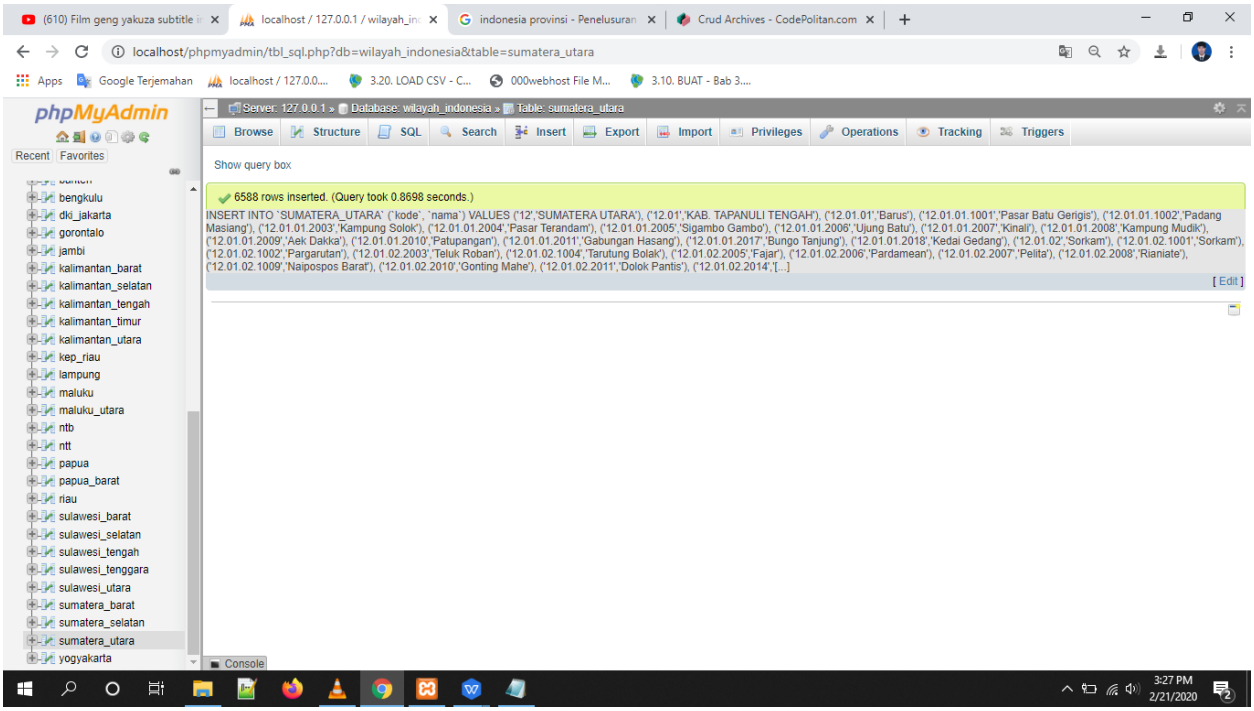
MySQL



Create sumatera utara

Neo4j

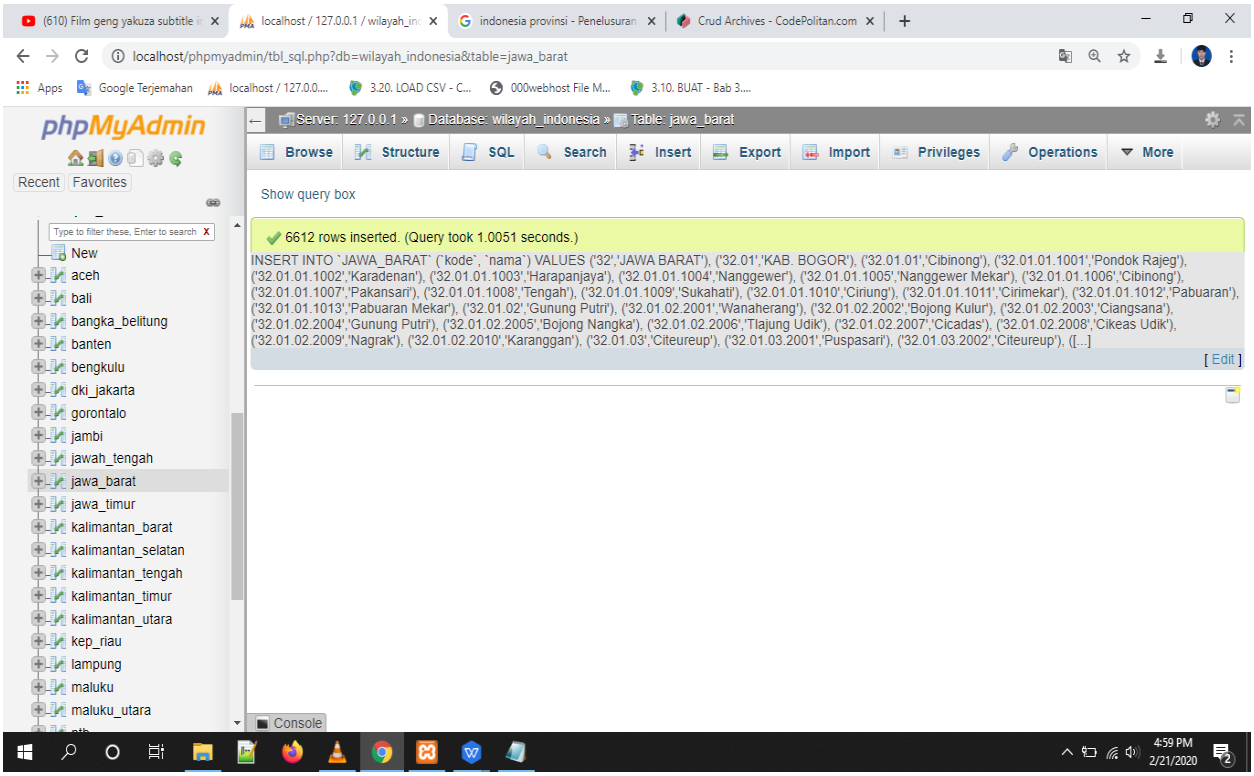
MySQL



Create jawa barat

Neo4j

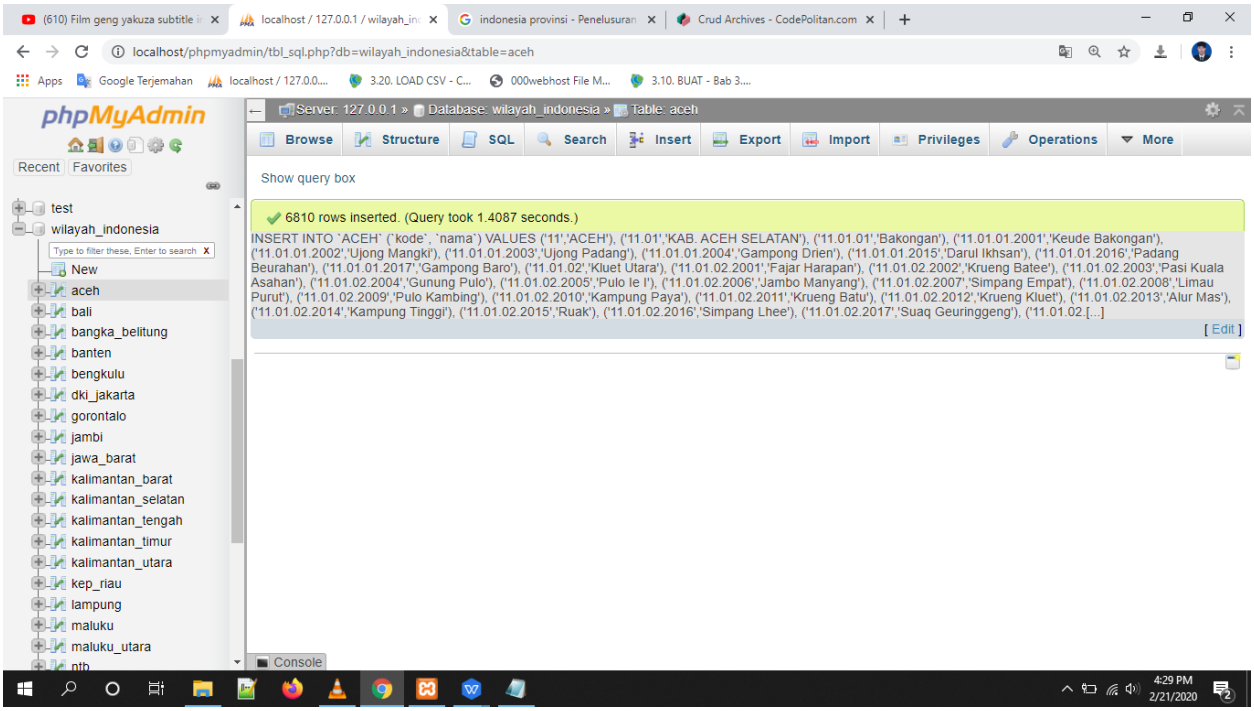
MySQL



Create aceh

Neo4j

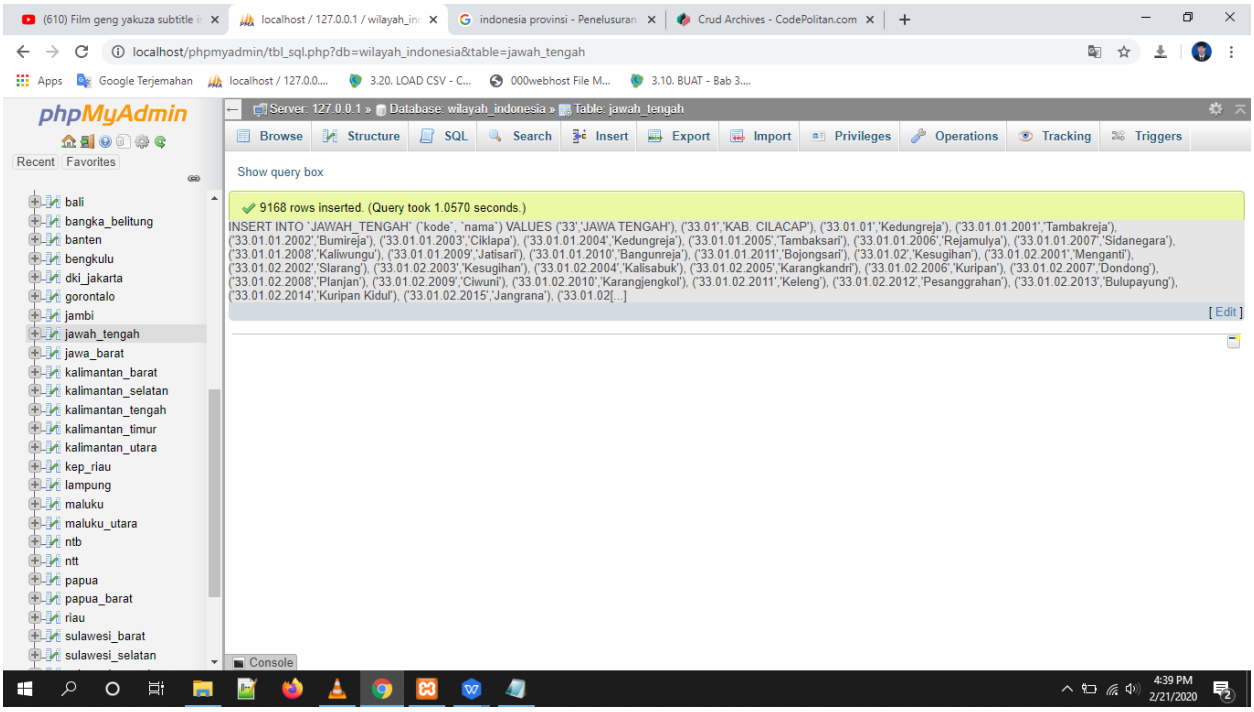
MySQL



Create jawa tengah

Neo4j

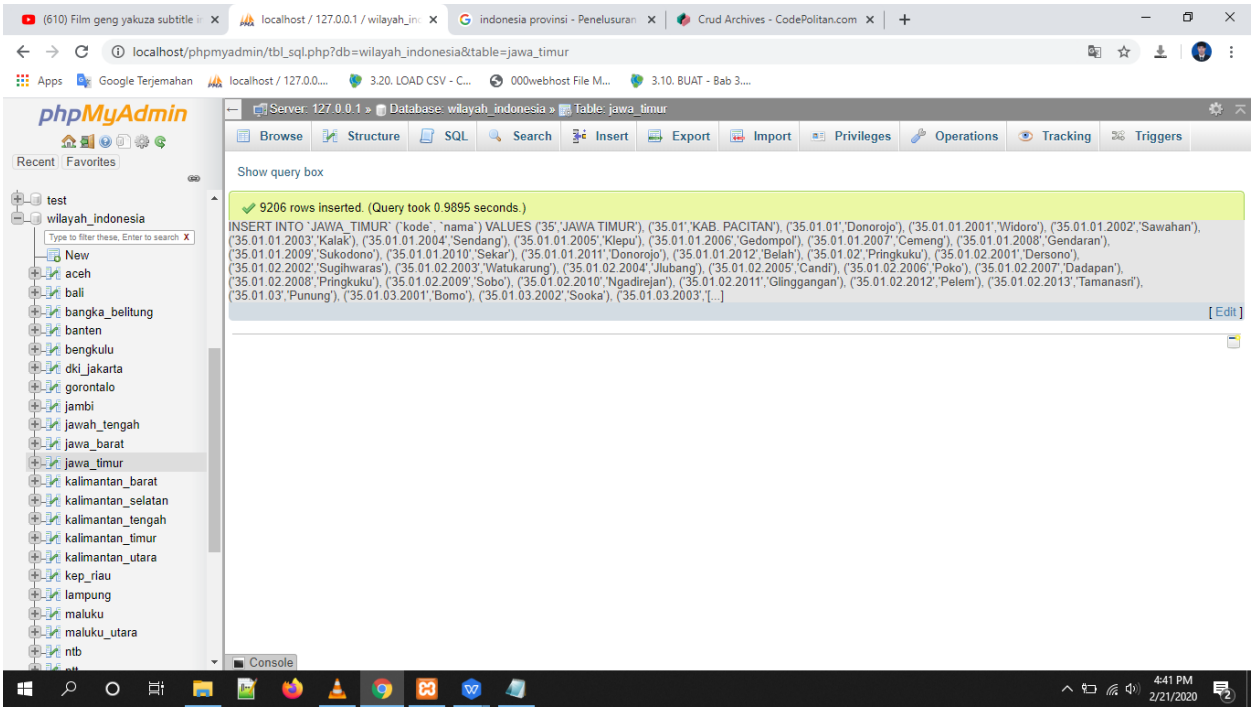
MySQL



Create jawa timur

Neo4j

MySQL



Tabel Hasil *runtime* dan rata-rata *create*

Tabel runtime create

Data	MySQL	NEO4J
317	0.1398	2.674
445	0.2101	3.684
493	0.2372	1.928
521	0.2786	2.913
540	0.1655	4.088
723	0.2433	2.941
782	0.2397	2.639
812	0.1958	1.790
1151	0.2853	5.728
1263	0.2933	2.842
1305	0.2983	1.756
1356	0.3215	2.541
1362	0.3042	1.875
1651	0.3528	2.580
1714	0.4235	2.065
1714	0.3551	5.123
1721	0.3633	2.094
2025	0.4131	5.534
2037	0.4060	3.517
2079	0.3731	5.685
2174	0.3877	2.152
2205	0.4006	3.157
2318	0.4265	2.374
2528	0.4475	2.113
2883	0.3182	3.347
3378	0.5100	2.106
3492	0.5354	2.850
3684	0.5079	2.413
6110	0.7060	2.990
6587	0.8698	2.909
6611	10.051	3.832
6809	14.087	2.819
9167	10.570	3.982
9205	0.9895	3.142

Tabel. Rata-rata time create

Rata-Rata MySQL	Rata-RataNEO4J
11569,33333	3.064

Tabel Hasil *runtime* dan rata-rata *Read*

Tabel runtime *Read*

Data	SQL	NEO4J
317	0,0032	0,01
445	0,0024	0,011
493	0,0018	0,017
521	0,0022	0,012
540	0.0016	0.007
723	0.0037	0.009
782	0.0020	0.011
812	0.0027	0.006
1151	0.0029	0.010
1263	0.0026	0.028
1305	0.0045	0.063
1356	0.0054	0.018
1362	0.0030	0.013
1651	0.0028	0.014
1714	0.0026	0.013
1714	0.0397	0.014
1721	0.0414	0.016
2025	0.0028	0.015
2037	0.0023	0.018
2079	0.0452	0.083
2174	0.0371	0.023
2205	0.0028	0.022
2318	0.0548	0.019
2528	0.0026	0.021
2883	0.0032	0.030
3378	0.0023	0.031
3492	0.0024	0.033

3684	0.0026	0.079
6110	0.0451	0.069
6587	0.0425	0.061
6611	0.0575	0.132
6809	0.0661	0.069
9167	0.0525	0.136
9205	0.0620	0.067

Tabel. Rata-rata time Read

Rata-Rata MySQL	Rata-Rata NEO4J
0,018151429	0,033714286

Tabel Hasil *runtime* dan rata-rata *Update*

Tabel runtime *Update*

Data	MySQL	NEO4J
317	0.1944	2.219
445	0.2893	2.673
493	0.5533	5.916
521	0.3506	4.689
540	0.3079	6.487
723	0.4292	5.061
782	0.5651	5.474
812	0.5465	9.756
1151	12.156	13.812
1263	12.551	13.893
1305	12.746	14.355
1356	14.987	10.848
1362	15.098	10.896
1651	21.211	19.812
1714	20.721	20.568
1714	23.836	20.568
1721	22.487	20.652

2025	24.841	24.376
2037	22.448	24.444
2079	31.146	24.948
2174	24.461	26.088
2205	23.985	26.46
2318	24.385	27.816
2528	34.034	30.336
2883	35.333	34.596
3378	43.228	40.536
3492	42.891	41.904
3684	45.934	44.208
6110	107.251	73.327
6587	92.195	79.044
6611	173.273	79.332
6809	144.539	81.708
9167	231.389	110.004
9205	231.506	110.465

Tabel. Rata-rata time *Update*

Rata-Rata MySQL	Rata-Rata NEO4J
57255,07692	31539,72727

Tabel Hasil *runtime* dan rata-rata *Delete*

Tabel runtime *Delete*

Data	MySQL	NEO4J
317	0.4451	0.334
445	0.4517	0.028
493	0.3729	0.018
521	0.3193	0.010
540	0.3865	0.007
723	0.6252	0.007

782	0.2837	0.009
812	0.3613	0.007
1151	0.4026	0.012
1263	0.4798	0.011
1305	0.3351	0.015
1356	0.2926	0.010
1362	0.2563	0.009
1651	0.9954	0.006
1714	0.4943	0.006
1714	0.4179	0.006
1721	0.3689	0.006
2025	0.3484	0.005
2037	0.2833	0.005
2079	0.4981	0.010
2174	0.6050	0.006
2205	0.2216	0.006
2318	0.2089	0.006
2528	0.3025	0.007
2883	0.2452	0.006
3378	0.2649	0.007
3492	0.2757	0.005
3684	0.2068	0.007
6110	0.4971	0.010
6587	0.3682	0.010
6611	0.2246	0.035
6809	0.2359	0.009
9167	0.3356	0.028
9205	0.2190	0.075

Tabel. Rata-rata time *Update*

Rata-Rata MySQL	Rata-Rata NEO4J
0,36084	0,021085714