TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE

HUFADZ IZZUDIN ROBBANI
NRP 5211 100 104

Dosen Pembimbing
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE

HUFADZ IZZUDIN ROBBANI
NRP 5211 100 104

Dosen Pembimbing
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – KS 141501

DEVELOPING GRAPH-BASED DATABASE VISUALIZATION APPLICATION FOR SQL SERVER USING DYNAMIC MANAGEMENT VIEW IN ORDER TO MAP IMPLICIT RELATIONSHIP IN A DATABASE.

HUFADZ IZZUDIN ROBBANI
NRP 5211 100 104

Supervisor
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

HUFADZ IZZUDIN ROBBANI
NRP 5211 100 104

Surabaya, 24 Januari 2018

PLH KEPALA
DEPARTEMEN SISTEM INFORMASI

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
DEPARTEMEN SISTEM INFORMASI

Edwin Riksakomara, S.Kom., M.T.
NIP. 19690725 200312 1 001

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

HUFADZ IZZUDIN ROBBANI
NRP 5211 100 104

Disetujui Tim Penguji : Tanggal Ujian : 12 Januari 2018
Periode Wisuda : Maret 2018

Radityo Prasetianto W., S.Kom, M.Kom (Pembimbing I)

Nur Aini R., S.Kom.,M.Sc.Eng.,Ph.D. **(Penguji I)**

Faizal Johan A., S. Kom, M.T. **(Penguji II)**

RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE

Nama Mahasiswa : Hufadz Izzudin Robbani
NRP : 5211 100 104
Departemen : Sistem Informasi FTIK-ITS
Dosen Pembimbing I : Radityo Prasetianto Wibowo, S. Kom., M. Kom.

ABSTRAK

Ketersediaan dokumentasi teknologi informasi (TI) adalah kunci sukses pengelolaan TI pada perusahaan. Dokumentasi yang layak sangat berguna agar proses maintenance dan learning lebih mudah serta mebantu knowlegde sharing sebuah organisasi. Kenyataannya dokumentasi TI adalah hal yang sering diabaikan.

Microsoft SQL Server dalam 5 tahun terakhir berhasil bertahan sebagai salah satu database terpopuler. Sikap abai perusahaan pada pendokumentasian SQL Server membuat pemahaman terhadap sistem database menjadi tacit knowledge yang terkesan ekslusif untuk orang-orang senior di perusahaan. Hal ini berdampak pada sulitnya orang baru mempelajari sistem database yang sudah ada sehingga memakan waktu yang lebih lama.

Ketiadaan dokumentasi SQL sebenarnya bisa diatasi dengan tool visualisasi database semisal DBVisualizer. Sayangnya dengan semua kelengkapan fitur dan dukungannya, DBVisualizer tidak mampu melakukan penelusuran pada relasi yang sifatnya implisit. Seringkali rancangan relasi antar tabel SQL tidak memiliki foreign key sehingga program semisal DBVisualizer kesulitan melakukan penelusuran. Karenanya penulis menawarkan solusi untuk memvisualkan database SQL ke dalam bentuk graph visual dengan memanfaatkan salah satu

fitur SQL Server yaitu Dynamic Management View. Tidak berhenti sampai di situ, sistem graph juga diterapkan sebagai basis visualisasi agar informasi yang disajikan lengkap dan akurat namun tidak kehilangan simplicity dan kemudahannya.

Kata Kunci: documentation, parsing, graph database, sql server, visualization, implicit relationship

DEVELOPING GRAPH-BASED DATABASE VISUALIZATION APPLICATION FOR SQL SERVER USING DYNAMIC MANAGEMENT VIEW IN ORDER TO MAP IMPLICIT RELATIONSHIP IN A DATABASE

Student Name : Hufadz Izzudin Robbani
NRP : 5211 100 104
Department : Sistem Informasi FTIK-ITS
Supervisor I : Radityo Prasetianto Wibowo, S. Kom., M. Kom.

ABSTRACT

The availability of information technology (IT) documentation is the key to successful IT management in companies. Worthy documentation is very useful for the process of maintenance and learning easier and mebantu knowlegde sharing of an organization. In fact, IT documentation is something that is often ignored.

In the last 5 years, Microsoft SQL Server managed to survive as one of the most popular database platform. The company's ignorant attitude towards documenting SQL Server makes understanding of database systems become tacit knowledge that impressed exclusively for senior in the company. This makes new employee's learning process become harder and takes a longer time.

The absence of SQL documentation can actually be solved with database visualization tools such as DBVisualizer. Unfortunately with all the excellence of its features and support, DBVisualizer is incapable to map implicit relationships. In many cases, the relationship between SQL tables does not have foreign keys so that programs such as DBVisualizer have difficulty to perform mapping process. Therefore the author

offers a solution to visualize the SQL database into the form of visual graph by utilizing one of the features of SQL Server that is Dynamic Management View. The graph DBMS is also applied as a base platform for the information visualization so it is not only complete and accurate but also simple and elegant.

Keywords: documentation, parsing, graph database, sql server, visualization, implicit relationship

KATA PENGANTAR

Alhamdulillah segala puji bagi Allah subhanahu wa ta'ala yang atas segala nikmat dan karunia-Nya proses pengerjaan Tugas Akhir ini berhasil selesai sesuai waktu yang ditargetkan. Tugas Akhir ini berjudul RANCANG BANGUN APLIKASI VISUALISASI DATABASE SQL SERVER DENGAN DYNAMIC MANAGEMENT VIEW BERBASIS GRAPH NEO4J UNTUK MEMETAKAN RELASI IMPLISIT PADA DATABASE. Penulis berharap dengan penulisan tugas akhir ini banyak pihak yang mendapat manfaat, terutama kalangan profesional Teknologi Informasi dan Komunikasi (TIK) di bidang database.

Banyak pihak yang terlibat baik secara langsung maupun tidak langsung dalam penulisan tugas akhir ini. Karenanya penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua penulis atas dukungan mereka berupa material, moral, dan doa yang tiada pernah putus demi kelancaran dan tercapainya harapan penulis. Tidak akan pernah bisa penulis membalas kebaikan keduanya.
2. Bapak Radityo Prasetianto Wibowo, S.Kom., M.Kom. selaku dosen pembimbing penulis, atas kesabarannya dan ketelatenannya dalam memberikan bimbingan dan *support* untuk pengerjaan tugas akhir ini.
3. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D. yang telah memberikan pelajaran berharga tentang kedisiplinan dan etos kerja.
4. Bapak Arif Wibisono, S.Kom., M.Sc. selaku dosen wali penulis yang tetap memberikan kepercayaan dan dukungan kepada penulis untuk menyelesaikan sisa perkuliahan.
5. Mas Ricky Asrul Sani selaku laboran Akuisisi Data dan Diseminasi Informasi (ADDI) yang telah membantu dalam hal-hal administratif untuk mendukung tugas akhir ini.
6. Teman-teman anggota laboratorium ADDI yang telah memberikan *support* kepada penulis.

7. Teman-teman laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) yang telah menjadi teman cengkrama di saat-saat sulit.
8. Dan kepada banyak pihak yang tidak bisa penulis sebut satu per-satu. Tanpa keterlibatan mereka, pengerjaan tugas akhir ini tidak akan selesai tepat waktu.

Penulis menyadari tugas akhir ini masih banyak kekurangan dan peluang perbaikan. Karenanya penulis membuka diri terhadap segara kritik dan saran yang membangun demi kemajuan ilmu TIK.

Surabaya, 24 Januari 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman sengaja dikosongi

BAB I

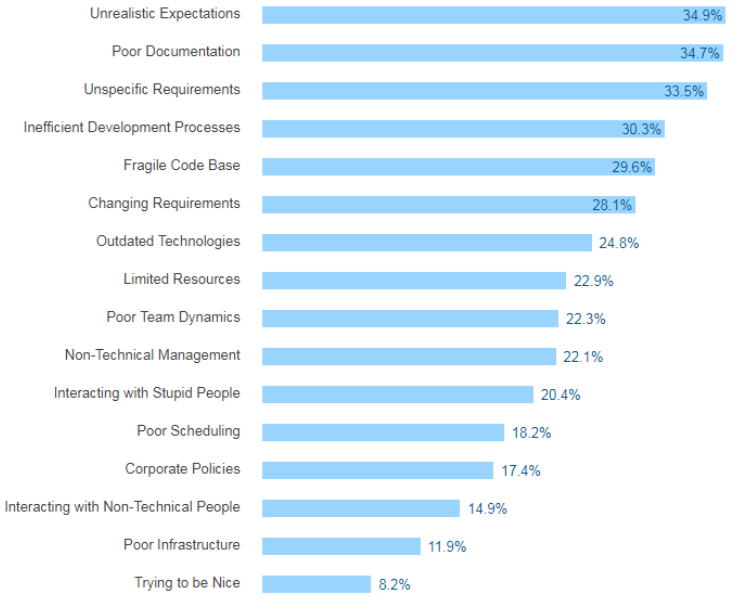
PENDAHULUAN

1.1. Latar Belakang

Dokumentasi sistem perangkat lunak yang benar, lengkap, dan konsisten adalah sebuah alat yang *powerful* yang menjadi penentu kesuksesan *maintenance* sistem [1]. Ketersediaan dokumentasi bermanfaat untuk memfasilitasi pemahaman & komunikasi perangkat lunak, mempermudah *learn and re-learn*, membuat perangkat lunak lebih *maintainable*, dan meningkatkan produktivitas & kualitas kerja seorang *maintainer* (Bauer and Parnas, 1995; Briand, 2003; Card et al., 1987; Carnegie, 1994; Clark et al., 1989; Conwell, 2000; Cook and Visconti, 1994, El Emam et al., 1998; Hogan, 2002; Holt, 1993; Kantner, 1997; Kantner, 2002; Martin and McClure, 1983; Parnas, 2000; Pence and Hon III, 1993; Rombach and Basili, 1987; Saunders, 1989; van Schouwen et al., 1993; Visaggio, 2001; Visconti and Cook, 2000; Visconti and Cook, 2002) [1]. Sebaliknya buruknya dokumentasi sistem merupakan alasan utama *software aging* dan penurunan kualitas (Parnas, 1994; Sousa and Mendes Moreira, 1998; Visaggio, 2001; Visconti and Cook, 2002) [1]. Fungsi dokumentasi bukan hanya untuk mendeskripsikan sistem perangkat lunak tapi juga menjelaskan proses relevan. *Control* dan manajemen yang layak hanya bisa dicapai dengan visibilitas akan proses, *stage* dan *task*-nya, peran-peran eksekutifnya, keputusan-keputusan dan motifnya, dan hasil dari masing-masing tugas proses [1].

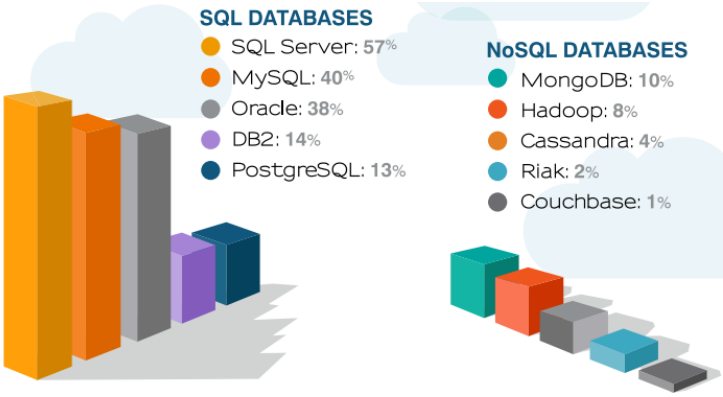
Sayangnya banyak perusahaan yang tidak peduli pada ketersediaan dan kelayakan dokumentasi sistem. Survey yang dilakukan StackOverflow (http://stackoverflow.com) kepada 50.000 pengembang perangkat lunak di tahun 2016 menunjukkan bahwa dokumentasi yang tidak layak menjadi tantangan kerja yang sering dihadapi oleh para pengembang perangkat lunak [2]. Ermine dalam Introduction to Knowledge Management menjabarkan permasalahan *knowledge management* pada kasus Departemen Kesehatan Filipina

disebabkan oleh lemahnya *monitoring* dan evaluasi, serta terbatasnya dokumentasi [3]. Bagian Sistem Informasi Biro Humas dan Hukum Sekretariat Kementerian Pemuda dan Olahraga (Kemenpora) Republik Indonesia juga menyatakan bahwa minimnya dokumentasi dari implementasi menyebabkan penggunaan data bersama belum bisa dilakukan di lingkungan Kemenpora sehingga proses sharing masih dilakukan secara manual [4].

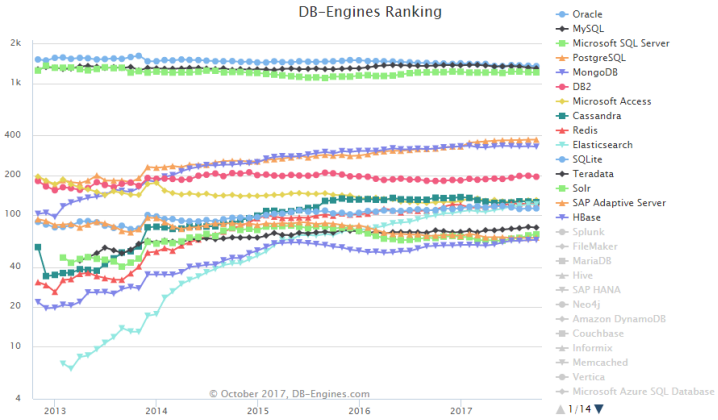


Gambar 1.1 Survey StackOverflow 2016 tentang tantangan kerja [2]

Hingga saat ini database SQL masih menjadi database paling favorit mengalahkan teknologi NoSQL. Hasil survey tahun 2014 yang ditunjukkan oleh gambar 1.2, memperlihatkan Microsoft SQL Server mendominasi pasar SQL mengalahkan MySQL, PostgreSQL, dan Oracle [5][6].



Gambar 1.2 Survei database menurut DB-Engine



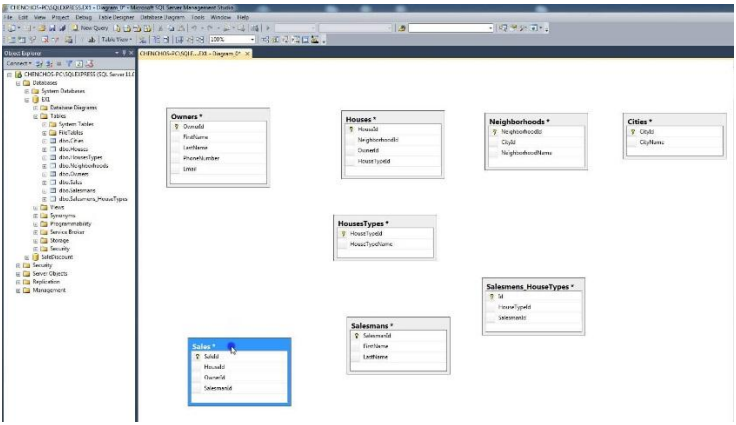
Gambar 1.3 Survey database terpopuler oleh Db-Engine

Sedangkan pada survey DB-Engines (http://db-engines.com) yang ditunjukkan oleh gambar 1.3 menunjukkan hasil yang sedikit berbeda yaitu Microsoft SQL Server menempati peringkat tiga database terpopuler di tahun 2017.

Penggunaan SQL yang begitu populer menjadikan permasalahan dokumentasi database SQL menjadi urgen untuk diselesaikan. *Knowledge* tentang sistem database yang tidak

terdokumentasi dengan baik semakin lama akan mejadi *tacit knowledge* yang hanya dipahami oleh pegawai senior di perusahaan. *Tacit knowledge* merupakan hasil dari *trial & error* sebagian karyawan dan tidak termanfaatkan karena perusahaan tidak tahu apa yang diketahui oleh sebagian karyawan tersebut (O'Dell & Grayson, 1998) [7].

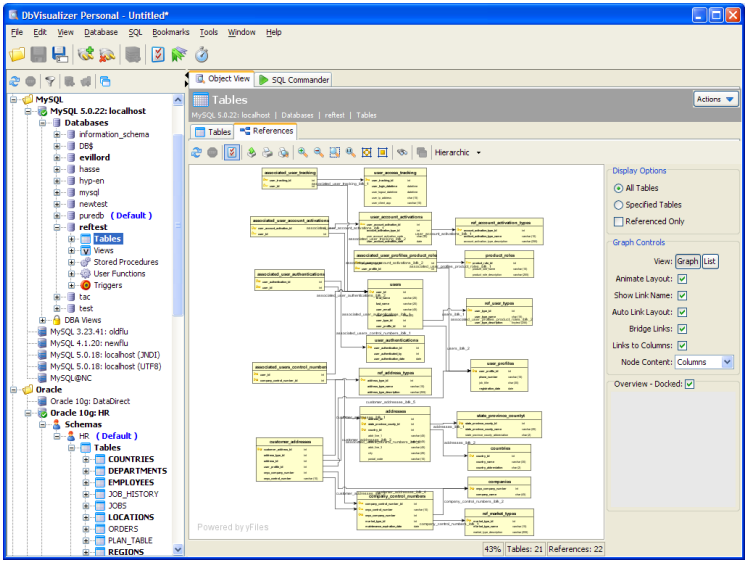
Kurangnya dokumentasi pada *platform* database SQL Server bisa diatasi dengan menggunakan bantuan aplikasi SQL Server Management Studio, sebuah aplikasi *database management* rilisan Microsoft yang dirancang khusus untuk database Microsoft SQL Server. SQL Server Management Studio telah menyediakan fasilitas Database Designer yang merupakan bagian dari fitur Visual DB Tools. Diagram Designer memungkinkan pengguna untuk mendesain dan memvisualisasikan database yang sedang terkoneksi [8].



Gambar 1.4 Fitur Diagram Designer SQL Server Management Studio

Selain menggunakan SQL Server Management Studio, alternatif yang tersedia untuk menghasilkan visualisasi SQL Server adalah dengan menggunakan aplikasi manajemen database semisal DBVisualizer. DBVisualizer memiliki fitur lengkap yang dapat memvisualisasikan *table*, *schema*, *view*, dan juga action query seperti SELECT, ALTER, dan lain-lain [9]. Selain fiturnya yang lengkap DBVisualizer juga mendukung

hingga 42 database populer. Tools lain yang juga bisa digunakan untuk menghasilkan visualisasi database SQL Server adalah JetBrain DataGrip, Sybase Power Designer, DbForge Studio for SQL Server, dan Navicat for SQL Server.



Gambar 1.5 Tampilan DBVisualizer

Sayangnya *tools* telah disebutkan sebelumnya memiliki keterbatasan tidak bisa melacak relasi implisit. Definisi relasi implisit menurut Huang dan Xue adalah sebuah relasi yang tersembunyi di antara entitas yang tidak dimodelkan sebagai relasi primary/foreign key [10]. Relasi implisit hanya muncul ketika dilakukan query SELECT atau membuat view yang di dalamnya terdapat JOIN. Karena relasi JOIN tidak terdefinisi langsung pada database, maka perlu dilakukan cara khusus untuk mendapatkan query dan view yang dijalankan dan mengambil relasi JOIN.

1.2. Perumusan Masalah

Berdasarkan latar belakang tersebut didapati permasalahan sebagai berikut:

1. Bagaimana mendapat informasi *table, schema, view, relation* pada database SQL Server?
2. Bagaimana mendapat informasi relasi implisit dalam database SQL Server?
3. Bagaimana memvisualisasikan informasi *database, table, schema, view,* dan relasi baik relasi berbasis *key* maupun relasi implisit pada ke dalam grafik?

1.3. Batasan Masalah

Dari perumusan masalah yang telah dipaparkan sebelumnya, maka yang menjadi batasan dalam tugas akhir ini adalah sebagai berikut:

1. Database yang digunakan adalah Microsoft SQL Server.
2. Database graph yang digunakan adalah Neo4J yang diinstal pada sistem operasi Microsoft Windows 10.
3. Hasil keluaran tugas akhir berupa aplikasi berbasis web yang hanya bisa dijalankan melalui *browser.*

1.4. Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah membuat aplikasi perangkat lunak berbasis web untuk:

1. Mendapatkan informasi struktur database termasuk *schema, table, column,* dan *view*.
2. Mendapatkan informasi seluruh relasi pada database SQL Server baik relasi berbasis *primary/foreign key* maupun relasi implisit berbasis JOIN.
3. Memvisualisasikan struktur database dan relasi dalam bentuk grafik yang mudah dipahami.

1.5. Manfaat

Manfaat yang dapat diperoleh dari pengerjaan tugas akhir ini antara lain:

1. Memberikan alternatif *tool* dokumentasi database yang menyajikan informasi yang lengkap dan akurat disertai tampilan grafik yang menarik.
2. Melengkapi *tool* dokumentasi database yang telah ada dengan memberikan fitur visibilitas terhadap relasi implisit dalam suatu database.

1.6. Relevansi

Tugas akhir ini berkaitan dengan mata kuliah Analisis dan Desain Perangkat Lunak, Konstruksi dan Pengujian Perangkat Lunak, Interaksi Manusia & Komputer, dan Manajemen Basis Data. Topik tugas akhir ini adalah akuisisi data dan *data visualization* yang merupakan bagian dari payung besar topik penelitian laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI).

Halaman sengaja dikosongi

BAB II

STUDI PUSTAKA

Studi pustaka ini berisi tentang literatur yang berhubungan dengan pengerjaan tugas akhir. Di dalam bab studi pustaka ini terdapat beberapa informasi penelitian sebelumnya dan penjelasan beberapa istilah yang digunakan dalam pembahasan bab-bab selanjutnya seperti: Dynamic Management View, relasi implisit, *parsing*, *graph,* Neo4J, dan D3.js.

2.1. Penelitian sebelumnya

Penenelitian sebelumnya yang berhubungan dengan topik tugas akhir dan menjadi referensi dalam pengerjaan tugas akhir dijelaskan pada tabel 2.1 berikut.

Tabel 2.1 Penelitian sebelumnya

Judul	Pembuatan Tools Pemantauan Kinerja Basis Data SQL Server Berbasis Web (Studi Kasus: pantau.its.ac.id)
Penulis	Erwin Wilbert Mapaliey
Metode	Membuat dashboard performa database SQL Server dengan memanfaatkan beberapa query DMV yang terbagi dalam 3 *metric* dimana tiap *metric* diakomodasi oleh sebuah *stored procedure*. Berikut *metric* yang digunakan: **1. Blocking** ✓ sys.dm_exec_requests ✓ sys.dm_exec_sessions ✓ sys.dm_tran_session_transactions ✓ sys.dm_tran_active_transactions **2. Missing index** ✓ sys.dm_db_missing_index_groups

	✓ sys.dm_db_missing_index_group_stats ✓ sys.dm_db_missing_index_details **3. CPU utilization** ✓ sys.dm_os_sys_info ✓ sys.dm_os_ring_buffers
Kesimpulan	1. Pembuatan tools pemantauan kinerja basis data berbasis web dengan memanfaatkan metrik-metrik pada SQL Server Performance Dashboard, yaitu Blocking, Missing Index, dan CPU Utilization dapat dilakukan dan diimplementasikan dengan baik pada sistem. 2. Metrik-metrik pada SQL Server Performance Dashboard diimplementasikan pada sistem berbasis web dengan cara memanggil stored procedure dari setiap metrik yang sudah tersedia saat menginstal SQL Server Performance Dashboard. 3. Berdasarkan hasil pengujian System Testing, semua fungsi pada sistem dapat berjalan dengan baik dan sesuai dengan output yang diharapkan. 4. Berdasarkan hasil pengujian Version Testing, tidak ada pengaruh terhadap perbedaan versi SQL Server dan SQL Server Performance Dashboard yang digunakan oleh server basis data, karena sistem hanya memanfaatkan stored procedure yang terdapat pada server basis data saat sudah terinstal SQL Server Performance Dashboard.

Judul	*Converting Relational to Graph Databases*
Penulis	1. Roberto De Virgilio 2. Antonio Maccioni 3. Riccardo Torlone
Metode	Sebuah relasi R dapat direpresentasikan dalam graph dengan mempertimbangkan *keys* dan *foreign keys* yang terlibat. Ide dasarnya adalah menyimpan data-data yang akan diambil bersamaan ke dalam satu graph g. Beberapa relasi R tergabung menjadi *schema path* SP. Prosedur melakukan iterasi pada SP, masing-masing elemen pada SP ubah jadi node A1,A2, hingga node terakhir (*sink node*). Tiap iterasi SP selesai, masukkan semua node ke dalam visited attributes VS.
Kesimpulan	Dalam tulisan ini dipresentasikan sebuah pendekatan untuk migrasi data dan query otomatis dari relasional ke *graph database*. Migrasi memanfaatkan *integrity constraints* yang ada di atas sumber untuk menyusun database target dengan tepat dimana jumlah akses yang dibutuhkan untuk menjawab query dikurangi. Paper juga membahas sebuah sistem yang mengimplementasi teknik migrasi untuk menilai tingkat *feasibility*. Agenda penelitian selanjutnya adalah memperbaiki teknik yang diusulkan pada paper ini mendapatkan target yang lebih *compact*.

2.2. Dynamic Management View

SQL Server mempunyai fitur Dynamic Management View (DMV) yang merupakan *tool* untuk mendapatkan informasi kondisi server yang bisa digunakan untuk memonitor kesehatan

server, diagnosis masalah, dan peningkatan kinerja. Query DMV berada di dalam sys schema. Penamaan query DMV mengikuti penamaan dm_*. Untuk menggunakan DMV nama fungsi atau view harus di-*prefix* di dalam schema sys. Ada dua jenis fungsi DMV yaitu *server scope* yang membutuhkan *permission* pada server dan *database scope* yang membutuhkan *permission* pada database [10].

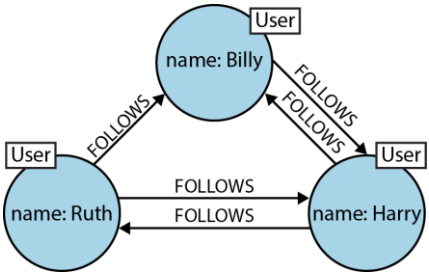
2.3. Relasi implisit

Untuk membuat relasi antar entitas data dalam database, *primary key* dan *foreign* didefinisikan. Namun dalam kasus nyata adanya *primary key* dan *foreign key* hanya menjelaskan sebagian dari relasi keseluruhan. Banyak relasi dimodelkan dengan view dan melakukan *selection query* pada tabel dan view. Sebagai contoh penggunaan query SELECT dan JOIN pada tabel EMPLOYEE dan DEPARTMENT menyiratkan relasi WORK AT di antara 2 tabel tersebut. Relasi semacam inilah yang diistilahkan oleh Huang dan Xue sebagai relasi implisit [11].

2.4. Parsing & Syntax Analysis

Setiap bahasa pemrograman memiliki aturan yang menentukan struktur *syntax* program yang baik (*well-formed*). *Syntax* bahasa pemrograman bisa dideskripsikan dalam bentuk notasi *context-free grammar*. *Parsing* dalam bahasa pemrograman juga disebut *syntax analysis* yaitu upaya untuk menentukan apakah suatu *syntax* bisa menghasilkan *string* dengan memanfaatkan input pada tahap sebelumnya yakni *lexical analysis.* Singkatnya parsing adalah menentukan apakah suatu syntax bisa membentuk susunan *string* berbentuk *abstract syntax tree* atau *parse tree* ataukah menghasilkan *error* [12].

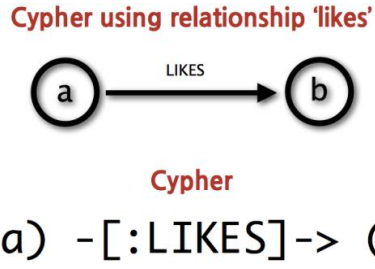
2.5. Graph



Gambar 2.1 Contoh graph sederhana [13]

Graph pada dasarnya terdiri dari *node* dan *relationship* atau juga disebut *edge*. *Graph* mendefinisikan entitas dengan *node* dan bagaimana hubungan antar entitas dengan *relationship*. *Graph* bisa ditemui dimana-mana. *Graph* sangat berguna untuk memahami keberagaman *dataset* di bidang-bidang semisal sains, pemerintahan, dan bisnis [13]. Gartner (http://www.gartner.com) menjelaskan ada 5 bidang *graph* yang bisa ditemui di dunia bisnis: *social graph*, *intent graph*, *consumption graph*, *interest graph*, dan *mobile graph* [14]. Contoh sederhana struktur *graph* dan hubungan antar entitasnya bisa digambarkan oleh gambar 2.1. Pada gambar 2.1 terdapat *node* yang diberi label User. Masing-masing *node* dihubungkan oleh *relationship*, yang membentuk konteks semantik: Billy dan Harry saling memfollow. Ruth dan Harry saling memfollow. Ruth memfollow Billy tapi tidak sebaliknya [13].

2.6. Neo4J *graph* DBMS



Gambar 2.2 Node dan relationship pada Cypher

Graph database management system (selanjutnya disebut basis data *graph*) adalah sistem manajemen basis data online dengan metode *create, read, update, delete* (CRUD) yang mengekspos sebuah model data *graph* [13]. Ada 2 properti basis data *graph*:

1. ***the underlying storage***
   Beberapa basis data *graph* menggunakan *native graph storage* yang dioptimalkan untuk menyimpan dan mengelola *graph.*
2. ***the processing engine***
   Beberapa definisi mengharuskan sebuah *graph* menggunakan *index-free adjacency*, yang artinya *nodes* yang terhubung, secara fisikal saling menunjuk (*point*) di dalam basis data. Bukan seperti model basis data *graph* yang dihasilkan dari operasi CRUD.

Neo4J adalah sistem manajemen basis data graph yang dikembangkan oleh Neo4J, Inc., sebuah perusahaan yang berbasis di San Francisco Bay Area, Amerika Serikat [15]. Neo4J adalah sebuah sistem basis data *transactional* dengan *native graph storage* dan *native processing* yang telah memenuhi ACID-*compliant* [16]. Hasil perhitungan peringkat DB-Engines menunjukkan Neo4J menempati peringkat pertama sistem basis data *graph* paling populer mengalahkan Microsoft Azure Cosmos DB, OrientDB, Virtuoso, dan beberapa sistem manajemen basis data *graph* lain [17]. Neo4J hadir dalam 3 lisensi: *community*, *enterprise*, dan *government*

[18]. Lisensi yang digunakan pada tugas akhir ini adalah lisensi *community.*

Berikut adalah contoh pendefinisian node You yang berlabel sebagai Person:

```
CREATE (you:Person {name:"You"}) RETURN you
```

Berikut adalah contoh pendefinisian hubungan atau *relationship* antar node:

```
MATCH  (you:Person {name:"You"})
CREATE (you)-[like:LIKE]->(neo:Database
{name:"Neo4j" })
RETURN you,like,neo)
```

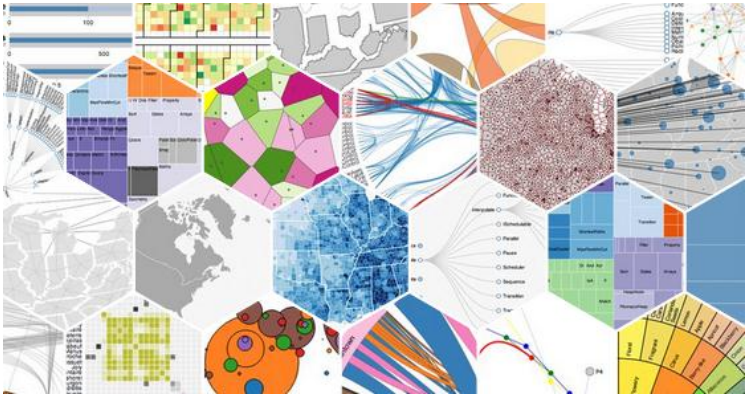
Neo4J memiliki bahasa query bernama Cypher Query Language. Cypher adalah sebuah bahasa deklaratif yang terinspirasi dari SQL untuk mendeskripsikan pola *graph* [19]. Cypher pada awalnya dikembangkan eksklusif untuk Neo4J tapi kemudian dengan dirilisnya openCypher [20] kini Cypher bisa diterapkan pada SAP Hana [21] dan AgensGraph [22]. Cypher Query mendefinisikan node *relationship* dengan syntax CREATE dan MATCH.

2.7. D3.js

Gambar 2.3 Logo D3.js

D3.js adalah library javascript berlisensi BSD untuk manipulasi dokumen berbasis data. D3 adalah kepanjangan dari Data-Driven Documents [23]. D3.js memungkinkan untuk mengikat *arbitrary data* ke sebuah Document Object Model (DOM) yang selanjutnya dapat diterapkan *data-driven transformation.* Sebagai contoh D3.js bisa digunakan untuk membuat tabel

berformat HTML dari sebuah array. Dengan data yang sama pula, sebuah diagram batang (*bar chart*) berformat SVG bisa dihasilkan. D3.js bisa diakses melalui repositori GitHub (http://github.com/d3/d3). D3.js mulai dikembangkan pada 18 Februari 2011. Saat ini versi terbaru D3.js adalah 4.11.0 yang dirilis pada 4 Oktober 2017 [24].



Gambar 2.4 Variasi visual yang dihasilkan D3.js [23]

Pada versi terbarunya D3.js menyediakan 30 modul yang bisa digunakan antara lain: arrays, geographies, hierarchies, polygons, dll. Contoh langkah-langkah penggunaan D3.js pada aplikasi HTML untuk membuat SVG *circle chart* yaitu:

Instalasi D3.js pada folder aplikasi dimana D3.js diterapkan. Apabila menggunakan NPM bisa dengan menuliskan `npm install d3`. Bisa juga dengan mengunduhnya secara manual pada repositori GitHub

(https://github.com/d3/d3/releases/download/v4.11.0/d3.zip) dan kemudian mengekstraknya ke folder aplikasi.

Insiasi D3.js pada header HTML

```
<script src="https://d3js.org/d3.v4.min.js">
</script>
```

Melakukan proses *data-binding* yaitu proses untuk mendefinisikan data yang digunakan.

```
var data = [
{name:"Indonesia",gdp:932,color: "green"},
{name:"Singapore",gdp:297,color: "blue"},
{name:"Malaysia",gdp:296,color:"red"}];
```

Membuat SVG container beserta attributnya.

```
.attr("width", 250)
.attr("height", 250)
.style("background-color", "#DEDEDE");
```

Membuat *circle chart* dengan mengambil data yang telah di-*binding*.

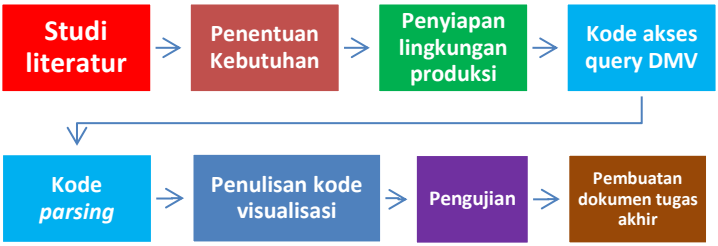
```
svg.selectAll("circle")
.data(data)
.enter()
.append("circle")
.attr("id", function(d) { return d.name })
.attr("cx", function(d) { return d.gdp })
.attr("fill", function(d) { return d.color });
```

Halaman sengaja dikosongi

BAB III

METODOLOGI

Pada bab ini akan dijelaskan tentang metodologi yang akan digunakan dalam pengerjaan tugas akhir. Metodologi akan digunakan sebagai panduan dalam penyusunan tugas akhir agar sistematis dan terarah. Adapun urutan dari pengerjaan tugas akhir dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur metode pengerjaan tugas akhir

3.1. Studi literatur

Studi literatur yang dilakukan adalah pembelajaran dan pemahaman literatur yang berkaitan dengan penyelesaian masalah. Beberapa hal yang akan dipelajari antara lain:

1. Konsep Microsoft SQL Server dan Dynamic Management View (DMV)
2. Konsep *graph database* dengan Neo4J
3. Konsep *parsing* dan *abstract syntax tree*
4. Teknik mendapatkan data DMV dengan bahasa PHP
5. Algoritma *parsing* dengan bahasa PHP
6. Teknik akses Neo4J dan penulisan query Cypher dengan bahasa PHP
7. Teknik pembuatan chart dari graph Neo4J
8. Penelitian sebelumnya yang telah disebutkan pada bab 2

Keluaran dari tahap pertama ini adalah pemahaman konsep dan teknis yang dibutuhkan untuk pembuatan aplikasi serta knowledge dari penelitian sebelumnya.

3.2. Penentuan kebutuhan fungsional

Untuk memastikan proses pembuatan aplikasi berjalan sesuai jadwal yang direncanakan dan sebagai acuan dalam pengujian aplikasi, maka perlu ditentukan kebutuhan fungsional. Dengan merujuk pada rumusan masalah yang ditetapkan pada bab I, berikut kebutuhan fungsional aplikasi:

1. Program bisa mengambil query dari DMV Microsoft SQL Server 2016.
2. Program bisa menyimpan query DMV yang telah diambil pada langkah 1.
3. Program bisa melakukan parsing query DMV.
4. Program bisa mengidentifikasi *database*, *schema*, *table*, relasi yang ada termasuk relasi berbasis *foreign key* dan relasi implisit berbasis JOIN.
5. Program bisa menyimpan hasil *parsing* ke dalam graph database Neo4J.
6. Program bisa mengolah data graph Neo4J yang telah didapat menjadi bentuk visual berupa *relation chart*.

3.3. Penyiapan lingkungan produksi

Untuk memastikan keberhasilan pembuatan aplikasi ada beberapa hal terkait lingkungan produksi yang perlu disiapkan. Beberapa hal yang perlu disiapkan antara lain:

1. Bahasa pemrograman yang digunakan adalah PHP: Hypertext Preprocessor atau yang lebih dikenal dengan PHP saja. Versi PHP yang digunakan adalah 5.6.
2. Instalasi Microsoft SQL Server 2016 dan SQL Server Management Studio (SSMS)
   - ✓ Mengunduh Microsoft SQL Server 2016 yang disediakan Institut Teknologi Sepuluh Nopember melalui layanan Microsoft Imagine
   - ✓ Melakukan instalasi Microsoft SQL Server 2016

- ✓ Menetapkan *username* dan *password* untuk keperluan otorisasi
- ✓ Mengunduh SSMS pada website resmi Microsoft (https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms)
- ✓ Melakukan instalasi SSMS
- ✓ Mengakses SQL Server melalui SSMS dengan login yang telah ditetapkan sebelumnya

3. Instalasi graph database Neo4J
   - ✓ Mengunduh Neo4J untuk Windows 10 64bit di website (https://neo4j.com/download/?ref=product)
   - ✓ Melakukan instalasi Neo4J seperti instalasi perangkat lunak Windows pada umumnya
   - ✓ Menentukan direktori graph
   - ✓ Mengakses Neo4J browser melalui alamat (http://127.0.0.1:7474/browser/) dengan username dan password default yaitu neo4j
   - ✓ Menetapkan password baru
4. Instalasi server lokal untuk eksekusi program PHP. Server lokal yang digunakan adalah XAMPP
   - ✓ Unduh XAMPP terbaru melalui website ApacheFriends (https://www.apachefriends.org/download.html)
   - ✓ Pilih XAMPP yang mendukung PHP 5.5
   - ✓ Lakukan instalasi seperti biasa
5. Instalasi D3.js
   Instalasi D3.js dengan versi yang sudah dimodifikasi khusus untuk visualisasi Neo4J yaitu neo4jd3.js yang bisa diunduh di GitHub (https://github.com/eisman/neo4jd3). Instalasi bisa dilakukan dengan NPM atau copy-paste manual.
6. Text Editor yang digunakan adalah Sublime Text 3 (build 3134) dengan plugin PHP snippet.
7. Instalasi GraphAware PHP Client for Neo4J

- ✓ Unduh GraphAware PHP Client melalui GitHub (https://github.com/graphaware/neo4j-php-client).
- ✓ Inisiasi dengan menuliskan kode berikut pada PHP

```
<?php
require_once 'vendor/autoload.php';
use
GraphAware\Neo4j\Client\ClientBuilder;
$client = ClientBuilder::create()-
>addConnection('bolt',
'bolt://neo4j:password@localhost:7687')-
>build();
$query = "MATCH (n:Person)-[:FOLLOWS]-
>(friend) RETURN n.name, collect(friend)
as friends";
$result = $client->run($query);
foreach ($result->getRecords() as
$record) {echo sprintf('Person name is :
%s and has %d number of friends',
$record->value('name'), count($record-
>value('friends'));}
```

3.4. Penulisan kode akses query DMV

Pada tahap ini *coding* dilakukan untuk mendapatkan *query* dari DMV untuk mendapat informasi table, views, schema, dan relasi, baik relasi berbasis *key* maupun relasi implisit berbasis JOIN yang terdapat pada database SQL Server. Berikut langkah-langkah pengerjaannya:

- ➢ **Koneksi ke SQL Server menggunakan PDO**
  Driver yang digunakan untuk melakukan koneksi ke SQL Server adalah PDO. Alasan menggunakan PDO adalah fleksibilitas PDO yang membuatnya bisa digunakan di lintas database berbeda. Berikut penulisan kode untuk koneksi:

```
$this->hostname = $hostname;
$this->port = $port;
$this->dbname = $dbname;
```

```
$this->username = $username;
$this->pwd = $pwd;
$this->connect();
```

- **Mendapatkan session id**
  Untuk mendapatkan informasi query
- **Query ke DMV untuk mendapat informasi**
  Secara umum jenis informasi yang dibutuhkan ada 2 macam yaitu: informasi struktur database yang eksplisit dan struktur database yang implisit. Untuk mendapatkan data struktur database yang eksplisit digunakan fitur System Catalog dari SQL Server yakni dengan beberapa function seperti: sys.tables, sys.objects, sys.all_tables, sys.foreign_keys, sys.sql_dependencies, sys.schemas.
  Untuk mendapatkan informasi implisit hal yang dilakukan adalah mendapatkan
  Contoh penulisan kode PHP untuk mendapatkan query yang pernah dijalankan pada SQL Server:

```
public function get_dmv($spid){
$stmt = $this->db->prepare("SELECT t.text
as query FROM sys.dm_exec_requests AS r
CROSS APPLY
sys.dm_exec_sql_text(r.sql_handle) AS t
WHERE session_id = $spid");
$stmt->execute();
while($row=$stmt-
>fetch(PDO::FETCH_ASSOC)){
   $query_dmv [] = $row[‘query’];
   }
}
```

3.5. Penulisan kode *parsing*

Tahap coding 3.5 ini bertujuan untuk mem-*parsing* query SQL yang telah didapat dari Dynamic Management View dari tahap sebelumnya. Tahap ini memanfaatkan kode *library* yang telah dibuat oleh Justin Swanhart. Kode *library* bisa diakses melalui repositori GitHub (https://github.com/greenlion/PHP-SQL-Parser). Parsing dilakukan dengan memanfaatkan constructor

PHPSQLParser dan method **parse()**. Berikut contoh penulisan kode program untuk parsing query:

```
$query = “SELECT a from some_table an_alias
WHERE d > 5”;

$parser = new PHPSQLParser($query);

print_r($parser->parsed);
```

Hasil parsing **PHPSQLParser** berbentuk array dengan format sebagaimana ditunjukkan oleh gambar 2.3.

```
[SELECT] => Array
    (
        [0] => Array
            (
                [expr_type] => colref
                [base_expr] => a
                [sub_tree] =>
                [alias] => `a`
            )

        [1] => Array
            (
                [expr_type] => colref
                [base_expr] => b
                [sub_tree] =>
                [alias] => `b`
            )

        [2] => Array
            (
                [expr_type] => colref
                [base_expr] => c
                [sub_tree] =>
                [alias] => `c`
            )

    )
```

Gambar 3.2 Format array hasil PHP SQL Parsing

Dari hasil *parsing* tersebut bisa didapatkan informasi *column* dan *table* dari sebuah query. Nama dan keterangan atribut yang melekat pada *table, schema* selanjutnya masing-masing disimpan dalam variable khusus.

3.6. Penulisan hasil parsing ke dalam query *graph*

Pada tahap ini hasil parsing pada tahap 3.6 dimasukkan ke dalam graph Neo4J dengan query Cypher. Karena program dibuat dengan bahasa PHP dengan memanfaatkan GraphAware PHP Client, maka penulisan query mengikuti aturan yang telah ditentukan GraphAware PHP Client.

Sebelum melakukan query graph dengan PHP perlu dilakukan koneksi menuju graph database. Berikut kode yang digunakan untuk membuat koneksi program PHP ke database graph:

```
use GraphAware\Neo4j\Client\ClientBuilder;
$client = ClientBuilder::create()-
>addConnection('default','http://neo4j:passwo
rd@localhost:7474')->addConnection('bolt',
'bolt://neo4j:password@localhost:7687')
->build();
```

Berikut adalah contoh penulisan query graph dengan PHP:

```
$client->run('CREATE ($db_name:Database
{name:“$db_name"})');
```

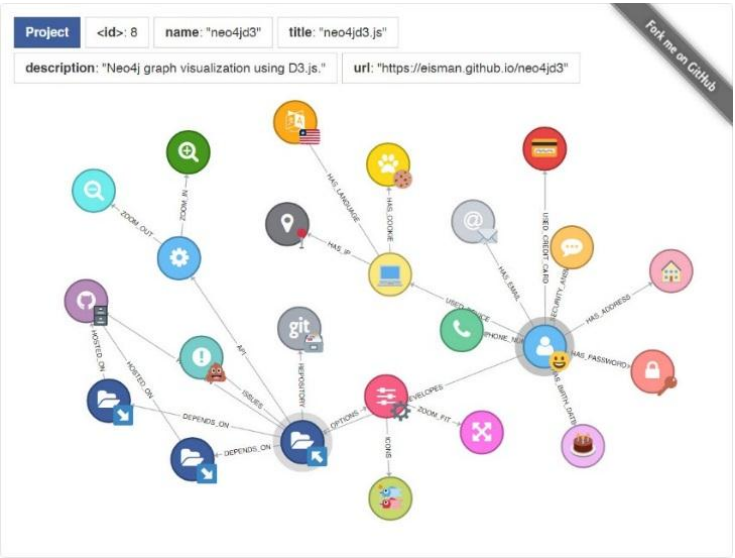
Query cypher juga bisa digunakan dengan sistem *stack* yaitu menumpuk beberapa query sekaligus dan dijalankan di akhir.

```
$stack = $client->stack();
$stack->push('CREATE (n:Person {uuid: {uuid}
})', ['uuid' => '123-fff']);

$stack->push('MATCH (n:Person {uuid: {uuid1}
}), (n2:Person {uuid: {uuid2} }) MERGE (n)-
```

```
[:FOLLOWS]->(n2)', ['uuid1' => '123-fff',
'uuid2' => '456-ddd']);

$results = $client->runStack($stack);
```

3.7. Penulisan kode visualisasi program

Kode program juga didukung library D3.js untuk melakukan visualisasi graph database yang telah dibuat. Visualisasi menggunakan neo4jd3.js yang merupakan modifikasi library D3.js yang disesuaikan untuk Neo4J. Selain dirancang untuk data dari Neo4J, library ini juga menagkomodasi penggunaan node dari text, Font Awesome, maupun Twitter Emoji.



Gambar 3.3 Contoh visualisasi menggunakan Neo4jD3.js

Library neo4jd3.js memanfaatkan fasilitas yang disediakan oleh D3.js dengan menggunakan masukan berupa file JSON. Neo4jd3.js mengakomodasi format JSON standar dari D3.js sebagaimana pada gambar 3.4 atau format JSON standar dari Neo4j sebagaimana pada gambar 3.5. File JSON tersebut

menyimpan informasi nodes dan relationship dari graph database beserta atribut yang melekat, semisal name, title, description, address, dan lainnya.


```
{
    "nodes": [
        {
            "id": "1",
            "labels": ["User"],
            "properties": {
                "userId": "eisman"
            }
        },
        {
            "id": "8",
            "labels": ["Project"],
            "properties": {
                "name": "neo4jd3",
                "title": "neo4jd3.js",
                "description": "Neo4j graph visualization using D3.js.",
                "url": "https://eisman.github.io/neo4jd3"
            }
        }
    ],
    "relationships": [
        {
            "id": "7",
            "type": "DEVELOPES",
            "startNode": "1",
            "endNode": "8",
            "properties": {
                "from": 1470002400000
            },
            "source": "1",
            "target": "8",
            "linknum": 1
        }
    ]
}
```


Gambar 3.4 Contoh format JSON standar dari D3.js

```
"results": [
    {
        "columns": ["user", "entity"],
        "data": [
            {
                "graph": {
                    "nodes": [
                        {
                            "id": "1",
                            "labels": ["User"],
                            "properties": {
                                "userId": "eisman"
                            }
                        },
                        {
                            "id": "8",
                            "labels": ["Project"],
                            "properties": {
                                "name": "neo4jd3",
                                "title": "neo4jd3.js",
                                "description": "Neo4j graph visualization using D3.js.",
                                "url": "https://eisman.github.io/neo4jd3"
                            }
                        }
                    ],
                    "relationships": [
                        {
                            "id": "7",
                            "type": "DEVELOPES",
                            "startNode": "1",
                            "endNode": "8",
                            "properties": {
                                "from": 1470002400000
                            }
```


Gambar 3.5 Contoh format JSON standar dari Neo4j

3.8. Pengujian

Pengujian dilakukan untuk memastikan program berjalan sesuai dengan kebutuhan fungsional yang telah ditetapkan sebelumnya pada tahap 3.2. Pengujian akan dilakukan pada dua macam database yakni database *sample* database Sales untuk database dengan skala kecil dan database Resource ITS (RESITS) untuk database skala besar. Database RESITS adalah database yang berisi data-data publikasi ilmiah dosen peneliti Institut Teknologi Sepuluh Nopember. Pengujian menggunakan database RESITS tidak hanya terbatas pada database RESITS tapi juga seluruh database yang berada dalam satu server dengan database RESITS.

Pengujian lebih banyak dilakukan pada level database yaitu dengan menguji query yang digunakan di dalam aplikasi untuk

dijalankan pada SQL Server. Hal ini dilakukan karena secara interaksi aplikasi ini sebagian besarnya dijalankan lewat command line, sehingga Selain itu pengujian juga dilakukan dengan memeriksa database graph Neo4j untuk memastikan 2 hal yaitu: aplikasi berhasil menyimpan data ke dalam graph database Neo4j dan aplikasi berhasil mengambil data graph. Pengujian juga dilakukan pada aplikasi yang sudah jadi dengan menjalankan fungsi filtering untuk menguji seberapa akurat fungsi filtering menyajikan data yang lebih spesifik yang diminta pengguna.

Pengujian juga dilakukan pada penggunaan library PHP SQL Parser untuk mengetahui seberapa akurat library mengenali nama database, schema, table, column, dan relasi JOIN yang terjadi. Pengujian dilakukan dengan menyiapkan beberapa skenario kasus query yang berbeda. Sebuah script sederhana disiapkan untuk hanya melakukan parsing pada skenario query yang telah disiapkan.

Halaman sengaja dikosongi

BAB IV

PERANCANGAN

Dalam bab perancangan ini akan dijelaskan tentang proses analisis kebutuhan dan desain perangkat lunak yang akan diimplementasikan.

4.1. Analisis Kebutuhan

Analisis kebutuhan dilakukan berdasarkan fungsi utama perangkat lunak dengan pengguna sistem, kemudian dihasilkan daftar kebutuhan fungsional sistem yang akan dibuat selanjutnya.

4.1.1. Fungsi Utama Aplikasi Perangkat Lunak

Pembuatan aplikasi visualisasi SQL Server berbasis *graph* ini bertujuan untuk memetakan dan memvisualisasi struktur relasi database Microsoft SQL Server. Selain itu aplikasi juga bertujuan menampilkan relasi antar tabel baik yang sifatnya eksplisit berbasis *foreign key* maupun yang implisit dengan menggunakan query JOIN.

4.1.2. Pengguna Aplikasi

Berikut adalah pengguna aplikasi visualisasi basisdata berbasis graph dan aktivitas-aktivitas yang dilakukan oleh pengguna aplikasi.

Tabel 4.1 Tabel pengguna aplikasi

Pengguna	Deskripsi	Aktivitas pada sistem
Database administrator	Pihak yang bertanggung jawab mengelola database	• Melihat informasi database, schema, table, column, foreign key, primary key, dan indeks.

		• Melihat informasi semua relasi yang ada pada database. • Melihat informasi relasi JOIN pada database.

4.1.3. Kebutuhan Fungsional

Pada subbab 3.2 kebutuhan fungsional aplikasi visualisasi database berbasis graph ini adalah sebagai berikut:

1. Program bisa mengambil query dari DMV Microsoft SQL Server 2016.
2. Program bisa menyimpan query DMV yang telah diambil pada langkah 1.
3. Program bisa melakukan parsing query DMV.
4. Program bisa mengidentifikasi *database*, *schema*, *table*, relasi yang ada termasuk relasi berbasis *foreign key* dan relasi implisit berbasis JOIN.
5. Program bisa menyimpan hasil *parsing* ke dalam graph database Neo4J.
6. Program bisa mengolah data graph Neo4J yang telah didapat menjadi bentuk visual berupa *relation chart*.

Dari kebutuhan fungsional yang tertera di atas, bila diperinci sesuai dengan alur aplikasi nyata, kebutuhan fungsionalnya akan menjadi sebagai berikut:

1. Aplikasi dapat melakukan koneksi ke SQL Server 2016
2. Aplikasi dapat mengambil informasi query dari fitur Dynamic Management View.
3. Aplikasi dapat mengambil informasi instance (linked server, remote server, local server) dan daftar database yang tersedia dari fitur System Catalog.
4. Aplikasi dapat menyimpan informasi query DMV pada kebutuhan No.2 ke dalam database MariaDB.

5. Aplikasi dapat mengambil informasi schema, table, dan column pada suatu database dengan menggunakan fitur System Catalog.
6. Aplikasi dapat mengambil informasi instance dan database yang berasosiasi dengan schema, table, dan column yang didapat dari kebutuhan No. 5 dengan menggunakan fitur metadata dan configuration pada SQL Server 2016.
7. Aplikasi dapat mengambil informasi relasi berbasis Foreign Key pada seluruh tabel yang didapat pada kebutuhan No. 5.
8. Aplikasi dapat menampilkan informasi relasi berbasis *foreign key* pada database.
9. Aplikasi dapat menjalankan kebutuhan No. 5,6,7,8 pada semua database yang didapat dari kebutuhan No. 3.
10. Aplikasi dapat mengambil query DMV yang telah disimpan pada database MariaDB.
11. Aplikasi dapat mengambil informasi schema, table, column, yang memiliki relasi JOIN dari hasil parsing query DMV yang didapat dari kebutuhan No. 10.
12. Aplikasi dapat menghasilkan array sesuai standar data output library Neo4jD3.js dan menyimpannya ke dalam sebuah file JSON.
13. Aplikasi dapat mengolah file JSON yang telah dibuat pada kebutuhan No. 13 menjadi bentuk graph diagram lengkap dengan seluruh informasi node dan relationship.
14. Aplikasi dapat melakukan filtering untuk menampilkan informasi pada pilihan database, schema, table, column, foreign key relationship, dan JOIN relationship tertentu.

4.2. Desain Aplikasi

Bagian ini menjelaskan desain aplikasi secara lebih detail mulai dari desain arsitektur dan juga desain query yang digunakan. Bagian ini tidak mencantumkan arsitektur informasi karena sudah tercantum dalam desain arsitektur aplikasi.

4.2.1. Query yang Digunakan

Aplikasi ini banyak sekali menggunakan akses tools SQL Server yang hanya bisa diakses melalui query khusus yang disediakan oleh sistem SQL Server. Berikut ini adalah beberapa query yang digunakan untuk aplikasi ini.

4.2.1.1. System Catalog, Configuration, dan Metadata

Untuk memenuhi kebutuhan fungsional nomor 1 sampai 6 perlu digunakan query untuk mendapatkan informasi daftar nama server, database, schema, table, column, dan relasi foreign key pada sistem yang hendak dituju. Untuk mendapatkannya digunakan beberapa query khusus yang telah disediakan oleh SQL Server yaitu System Catalog untuk mendapatkan nama table, column, dan relasi foreign key. Query System Catalog dikombinasikan dengan Configuration dan Metadata untuk mendapatkan informasi server, database , dan schema tempat dijalankannya query System Catalog. Sehingga akan didapati asosiasi antara nama tabel yang telah didapat dari System Catalog dengan nama Database dan Schema yang didapat dari Metadata dan nama Server yang didapat dari Configuration.

Query pertama yang digunakan adalah query **sys.servers** untuk mendapatkan informasi instance atau server yang tersedia di sistem database. Informasi yang bisa didapatkan yaitu informasi Linked Server, Remote Server, dan Local Server. Berikut query yang akan digunakan:

```
SELECT name as srv from sys.servers
```

Query kedua yang digunakan adalah query **sys.databases** yaitu query yang digunakan untuk mendapatkan informasi database yang tersedia pada server. Informasi yang dibutuhkan hanya nama database. Database yang didapat haruslah database yang dibuat oleh user dan bukan database sistem atau database reporting. Berikut query yang dibutuhkan:

```
SELECT NAME AS db
FROM   sys.databases
WHERE  NAME NOT LIKE '%master%'
```

```
        AND NAME NOT LIKE '%tempdb%'
        AND NAME NOT LIKE '%reportserver%'
        AND NAME NOT LIKE '%model%'
      AND NAME NOT LIKE '%msdb%'
```

Query System catalog untuk mendapatkan nama database adalah **sys.table** yang di-JOIN dengan **sys.column** sehingga didapat asosiasi kolom dan tabel. Selanjutnya ditambahkan query metadata yaitu DB_ID untuk mendapatkan ID dari sys.table dan sys.column yang telah didapat. Supaya didapat nama databasenya, digunakan DB_NAME() sehingga menjadi DB_NAME(DB_ID()). Mendapatkan informasi schema hampir sama dengan database yaitu dengan SCHEMA_NAME(schema_id). Kemudian query @@SERVERNAME digunakan untuk mendapatkan nama server. Sehingga bila dikombinasikan akan menjadi seperti berikut:

```
SELECT @@SERVERNAME           AS srv,
       Db_name(Db_id())       AS db,
       Schema_name(schema_id) AS sch,
       sys.tables.NAME        AS tbl,
       sys.columns.NAME       AS col
FROM   sys.tables INNER JOIN sys.columns ON
sys.tables.object_id = sys.columns.object_id
```

Query keempat yang dibutuhkan adalah kombinasi penggunaan **sys.foreign_key_columns** dengan **sys.schemas, sys.tables, sys objects,** dan **sys.columns.**

```
SELECT
      @@SERVERNAME as srv,
      DB_NAME(DB_ID()) as db,
      obj.name AS fk_rel,
      sch2.name AS par_sch,
      tab2.name AS par_tbl,
      col2.name AS par_col,
      sch1.name AS ref_sch,
```

```
      tab1.name AS ref_tbl,
      col1.name AS ref_col
FROM sys.foreign_key_columns fkc
INNER JOIN sys.objects obj
  ON obj.object_id = fkc.constraint_object_id
INNER JOIN sys.tables tab1
  ON tab1.object_id = fkc.parent_object_id
INNER JOIN sys.columns col1
  ON col1.column_id = parent_column_id
  AND col1.object_id = tab1.object_id
INNER JOIN sys.tables tab2
  ON tab2.object_id=fkc.referenced_object_id
INNER JOIN sys.columns col2
  ON col2.column_id = referenced_column_id
  AND col2.object_id = tab2.object_id
INNER JOIN sys.schemas sch1
  ON tab1.schema_id = sch1.schema_id
INNER JOIN sys.schemas sch2
  ON tab2.schema_id = sch2.schema_id
```

4.2.1.2. Dynamic Management View

Dynamic Management View (DMV) adalah tool yang disediakan oleh Microsoft SQL Server untuk mendapatkan informasi kinerja query yang pernah dijalankan. Untuk mendapatkan informasi dari kebutuhan fungsional nomor 7 dan untuk melengkapi nama database, schema, table, dan column, yang tidak berhasil didapat oleh System Catalog, Metadata, dan Configuration, maka digunakanlah DMV. Hasil query DMV ini akan mengembalikan daftar query yang pernah dijalankan pada sistem. Yang selanjutnya akan digunakan sebagai masukan untuk diparsing dan didapat nama database, schema, dan table.

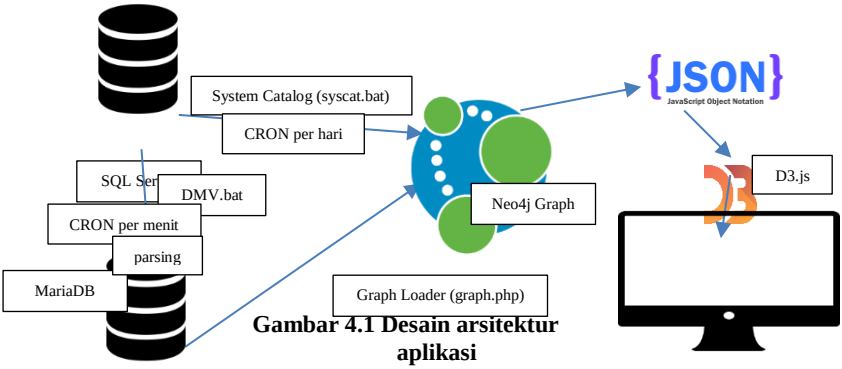
Tidak semua query yang pernah dijalankan layak dijadikan masukan untuk diparsing. Query yang digunakan dibatasi hanya yang mengandung unsur JOIN dan ON untuk memenuhi kebutuhan fungsional No. 7. Selain itu query haruslah bukan query system karena data yang didapat tidak akan ada

hubungannya dengan tabel yang dibuat user. Selanjutnya query yang dimaksud juga haruslah query yang bukan query untuk membuat stored procedure. Karena query untuk membuat stored procedure jumlah karakternya tidak bisa diakomodasi oleh library parsing yang digunakan. Hasil query diurutkan berdasarkan waktu eksekusi terakhir untuk memastikan informasi yang didapat adalah yang terbaru. Sehingga berikut adalah query dari seluruh ketentuan di atas:

```
SELECT      t.[text]
FROM        sys.dm_exec_cached_plans AS p
INNER JOIN  sys.dm_exec_query_stats  AS s
ON          p.plan_handle = s.plan_handle
CROSS apply
sys.dm_exec_sql_text(p.plan_handle) AS t
WHERE t.[text] LIKE N'%join%on%'
AND   t.[text] NOT LIKE N'%sys%'
AND   t.[text] NOT LIKE N'%create procedure%'
AND   t.[text] NOT LIKE N'%INSERT%'
ORDER BY    s.last_execution_time DESC
```

Query `sys.dm_exec_cached_plans` membutuhkan *permission* untuk akses hingga level instance SQL Server.

4.2.2. Desain Arsitektur Aplikasi



Gambar 4.1 Desain arsitektur aplikasi

Terdapat beberapa proses yang berjalan pada aplikasi ini. Proses pertama adalah proses mendapatkan seluruh informasi server, database, schema, table, column, dan foreign key realtionship pada database SQL Server. Di proses pertama inilah query system catalog, metadata, dan configuration dijalankan. Query tersebut dijalankan dalam sebuah script PHP. Informasi yang diperoleh dari proses ini selanjutnya dimasukkan ke graph DBMS Neo4j. Proses memasukkan data ke dalam graph Neo4j menggunakan bahasa query khusus yaitu cypher. Proses pertama ini akan diotomatisasi untuk berjalan selama setiap hari sekali.

Proses kedua yang berjalan adalah proses mendapatkan daftar query yang pernah dijalankan pada database SQL Server. Pada proses inilah query DMV dijalankan. Singkatnya proses ini adalah melakukan query ke SQL Server untuk mendapatkan query yang pernah dijalankan. Query yang telah didapat, dikumpulkan di dalam database MariaDB. Proses kedua ini diotomatisasi untuk dijalankan setiap menit karena fitur DMV hanya menyajikan informasi query yang sering dijalankan atau yang terakhir dijalankan.

Proses ketiga adalah proses parsing yaitu memotong-motong query menjadi serangkaian array dengan tujuan untuk memperoleh informasi yang dibutuhkan. Informasi yang

dibutuhkan aplikasi visualisasi database ini adalah informasi tabel dan kolom yang dihubungkan dengan JOIN. Hasil parsing ini selanjutnya disimpan pada graph database Neo4j untuk melengkapi struktur database yang dihasilkan oleh proses pertama.

Proses keempat adalah proses mendapatkan informasi yang telah disimpan di Neo4j. Informasi yang didapatkan tersebut selanjutnya dimasukkan ke dalam sebuah array. Array inilah yang selanjutnya akan di-*encode* menjadi sebuah file json.

Proses kelima adalah melakukan mengolah file json yang telah dihasilkan pada proses sebelumnya menjadi bentuk diagram graph. Pada proses inilah penentuan icon untuk tiap jenis node dilakukan.

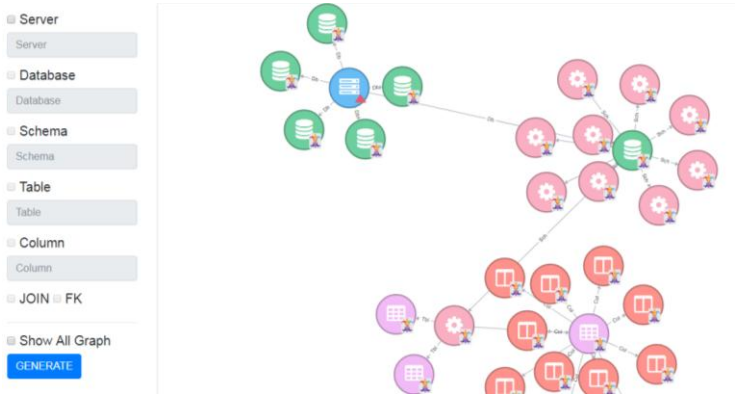
4.2.3. Desain Antarmuka

Sebagian besar aplikasi visualisasi database SQL Server dengan Dynamic Management View berbasis graph Neo4j ini dijalankan melalui command line dan task scheduling. Antarmuka yang tersedia hanyalah pada script *graph loader*. Antarmuka pada script ini berfungsi menyajikan graph dalam bentuk diagram yang bisa diakses pada browser HTTP biasa.

Tampilan script ini secara umum terbagi menjadi ruas kanan dan kiri. Ruas kiri berfungsi untuk memasukkan kriteria *filtering*. Pada ruas kiri terdapat 5 input *text*: Server, Database, Schema, Table, dan Column untuk menampilkan node dan relationship pada posisi server, database, schema, table, atau column tertentu. Selain itu juga terdapat 5 input *checkbox* dengan nama yang sama yang posisinya mendahului masing-masing input *text* di atas. Selain itu juga ada 3 *checkbox* tambahan yaitu FK yaitu untuk menampilkan *foreign key relationship* pada kolom tertentu, JOIN untuk menampilkan JOIN *relationship* pada kolom tertentu, dan Show All Graph untuk menampilkan semua informasi yang tersimpan dalam graph Neo4j. Di bagian paling bawah ruas kiri terdapat tombol

biru bertuliskan GENERATE yang berfungsi untuk mengeksekusi kriteria *filtering* yang telah dimasukkan.

Ruas kanan berfungsi sebagai penampil *graph diagram*. Tampilan *default* ruas kanan adalah ruang kosong berwarna putih. Apabila kriteria *filtering* telah dimasukkan dan ditekan tombol GENERATE maka *graph diagram* ditampilkan pada ruas kanan ini. Sehingga keseluruhan antarmuka aplikasi bisa dilihat pada gambar 4.2.



Gambar 4.2 Desain antarmuka keseluruhan

BAB V

IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi terhadap perancangan sistem yang telah dilakukan pada bab sebelumnya. Pada bagian implementasi ini akan dijelaskan mengenai lingkungan implementasi, pembuatan fungsi-fungsi sistem dalam bentuk kode program, serta proses pengujian sistem yang akan dilakukan.

5.1. Lingkungan Implementasi

Aplikasi perangkat lunak berjalan pada server lokal yang beralamat di http://localhost. Berikut adalah spesifikasi komputer yang digunakan pada pembuatan dan pengujian aplikasi:

Tabel 5.1 Spesifikasi komputer server *localhost*

Processor	Intel(R) Celeron(R) CPU N2830 @2.16 GHz.
Memory	2 GB
Operating System	Windows 7 Professional

Pembuatan aplikasi ini juga menggunakan beberapa teknologi pendukung sebagaimana terlampir pada tabel 5.2.

Tabel 5.2 Teknologi pendukung pengembangan aplikasi

Webserver	Apache 2.4.29
Bahasa Pemrograman	• PHP 5.6 with Composer • JavaScript
RDBMS	MariaDB 10.1.28
Graph DBMS	Neo4J 3.3.1 Enterprise

Text Editor	Sublime 3.0 dengan plugin: • PHP Snippet • PHP Autocomplete
Web Browser	Google Chrome Versi 63.0.3239.132 (32bit)
Library	• Bootstrap 3.3.7 • Font Awesome 5 • D3.js 4.12.0 • PHP SQL Parser by greenlion • Neo4jD3.js by eismann • PDO & ODBC 11 • GraphAware PHP Client
OS Component	• Windows Task Scheduler • Command Prompt (cmd.exe)

Lingkungan aplikasi pada tabel 5.1 dan 5.2 harus dipastikan berjalan dengan baik. Secara default Apache Server berjalan pada port 80. Apabila Apache Server tidak berjalan, ada kemungkinan port 80 sedang digunakan oleh SQL Server Reporting Service. Ada beberapa cara untuk memastikan Apache Server bisa berjalan. Cara pertama adalah dengan mengubah port Apache yang digunakan pada sistem dengan mengubah nilai port pada file PHP.ini. Cara kedua adalah dengan mematikan SQL Reporting Service. Pada pengerjaan tugas akhir ini cara yang digunakan adalah cara kedua. Cara mematikan SQL Server Reporting Service bisa dengan lewat Reporting Service Configuration Management.

5.2. Konfigurasi Aplikasi

Untuk memastikan aplikasi perangkat lunak terkoneksi dengan DBMS, diperlukan konfigurasi koneksi yang membutuhkan alamat server DBMS, nama *database*, *username*, dan *password*. Konfigurasi ini disatukan dalam satu file yaitu **config.php**.

Konfigurasi yang dibutuhkan pada aplikasi ini ada 3 macam: Microsoft SQL Server, MariaDB, dan Neo4j.

Sebelum memulai konfigurasi dan koneksi ke DBMS, perlu dilakukan instalasi dan pemanggilan beberapa library: PDO untuk koneksi ke SQL Server & Maria DB, dan GraphAware PHP Client untuk koneksi ke Neo4j. Instalasi PDO dilakukan dengan mendownload library PDO lewat website resmi Microsoft dan menginstalnya pada folder C:\xampp\php\ext\. Instalasi GraphAware PHP Client dilakukan dengan Composer dengan mengetik *command* :

```
composer require graphaware/neo4j-php-client:^4.0
```

Neo4j client for PHP perlu dipanggil dengan kode seperti diperlihatkan pada gambar 5.1.

```
//initiate library
require_once 'vendor/autoload.php';
use GraphAware\Neo4j\Client\ClientBuilder;
```

Gambar 5.1 Kode pemanggilan GraphAware Neo4j for PHP

Setelah library dipanggil yang perlu dilakukan selanjutnya adalah melakukan koneksi ke Neo4j. Pada prinsipnya Neo4j for PHP akan otomatis melakukan koneksi ke database Neo4j yang sedang aktif pada suatu alamat server. Sehingga yang dibutuhkan pada konfigurasi hanya alamat server lokasi Neo4j. Database Neo4j bisa diakses lewat kanal HTTP pada port 7474, kanal Bolt pada port 7687, dan HTTPS pada port 7473. Pada aplikasi ini digunakan koneksi pada Bolt sehingga kode koneksi ke Neo4j nampak seperti pada gambar 5.2.

```
//inisiasi koneksi graph
$neo = ClientBuilder::create()
    ->addConnection('bolt','bolt://neo4j:password@localhost:7687')
    ->build();
```

Gambar 5.2 Koneksi ke *graph database* Neo4j

Langkah selanjutnya adalah melakukan koneksi ke SQL Server. Alamat server, username, dan password masing-masing akan dibuatkan variabel supaya lebih mudah untuk apabila hendak mengganti alamat, username, atau password kedepannya. Potongan kode konfigurasi dan koneksi ke SQL Server ditunjukkan pada gambar 5.3.

```
$srv="CAKTON27\SQLEXPRESS";
$user="admin";
$pwd="0318413468";
ini_set('max_execution_time', 30000);
//inisiasi koneksi Microsoft SQL Server
try {$con = new PDO("sqlsrv:Server=$srv;Database=master",$user,$pwd);
} catch (PDOException $e) {echo $e->getMessage();}
```

Gambar 5.3 Koneksi ke SQL Server

Setelah itu perlu juga dilakukan koneksi ke MariaDB. Karena database MariaDB yang digunakan hanya satu, maka konfigurasi cenderung tetap. Karena itu alamat server, username, dan password tidak perlu disendirikan pada variabel khusus. Potongan kode konfigurasi dan koneksi ke MariaDB ditunjukkan pada gambar 5.4.

```
//inisiasi koneksi MySQL
try {$cons = new PDO(
"mysql:host=localhost;dbname=query","root","0318413468");
}catch (PDOException $e) {echo $e->getMessage();}
```

Gambar 5.4 Koneksi ke MariaDB

5.3. Pemahaman Array Hasil Parsing

Sebelum mengimplementasikan library PHP SQL Parser pada aplikasi, terlebih dahulu perlu dilakukan pemahaman terhadap array yang dihasilkan oleh library ini. Hal ini dilakukan agar diketahui dimana posisi informasi yang dibutuhkan aplikasi pada array yang dihasilkan library. Selain itu tahapan ini dilakukan agar diketahui sejauh mana kemampuan library PHP SQL Parser dalam menyelesaikan parsing dari berbagai variasi penulisan query SQL Server.

Untuk itu dibuatlah sebuah script sederhana bernama **coba.php** yang memanggil library PHP SQL Parser. Pada script juga disiapkan variable $query sebagai penampung query yang

hendak dipakai sebagai ujicoba. Query yang telah dimasukkan ke dalam variable akan dieksekusi dan menghasilkan luaran berbentuk array hasil parsing yang ditampung pada variable $data. Selanjutnya array $data dicetak di layar. Selanjutnya dilakukan pengamatan pada posisi array tingkat kedalaman berapa letak informasi schema, table, column, jenis JOIN, dan tabel serta column yang terlibat relasi JOIN. Kode **coba.php** bisa disimak pada gambar 5.5.

```php
namespace PHPSQLParser;
require 'vendor/autoload.php';

$query = '';

$parser = new PHPSQLParser($query, true);
$data = $parser->parsed;

print_r($data);
```

Gambar 5.5 Kode script coba.php

Selanjutnya siapkan contoh query yang mengandung relasi JOIN di dalamnya. Agar hasil pengamatan semakin akurat, disiapkan beberapa skenario bentuk query yaitu:

A. Basic SELECT query with JOIN

Skenario dengan menggunakan query select dengan relasi JOIN. Query yang digunakan untuk skenario ini adalah:

```
SELECT dbo.mahasiswa.mahasiswa_name,
DBO.MATKUL.matkul_name,dbo.skem.skem_id
FROM dbo.mahasiswa
LEFT JOIN dbo.matkul
ON matkul.matkul_mhs_id=mahasiswa.mahasiswa_id
LEFT JOIN dbo.skem
ON skem.skem_mhs_id=mahasiswa.mahasiswa_id
```

B. Basic UPDATE query with JOIN

Skenario dengan menggunakan query select dengan relasi JOIN. Query yang digunakan untuk skenario ini adalah:

```
UPDATE dbo.mahasiswa
LEFT JOIN dbo.matkul
ON matkul.matkul_mhs_id=mahasiswa.mahasiswa_id
```

```
WHERE matkul.matkul_id IS NOT NULL
```

C. Nested SELECT with SELECT 2 level

Skenario dengan menggunakan query SELECT di dalam SELECT. Query yang digunakan untuk skenario ini adalah:

```
SELECT matakul.* FROM (SELECT
dbo.mahasiswa.mahasiswa_name,
dbo.matkul.matkul_name,dbo.skem.skem_id
FROM win.ta.dbo.mahasiswa
LEFT JOIN dbo.matkul ON
matkul.matkul_mhs_id=mahasiswa.mahasiswa_id)
matakul
```

D. Nested INSERT with SELECT 2 level

Skenario dengan menggunakan query SELECT di dalam INSERT. Query yang digunakan untuk skenario ini adalah:

```
INSERT INTO (SELECT dbo.mahasiswa.mahasiswa_name,
DBO.MATKUL.matkul_name,dbo.skem.skem_id
FROM win.ta.dbo.mahasiswa
LEFT JOIN dbo.matkul
ON matkul.matkul_mhs_id=mahasiswa.mahasiswa_id)
```

E. Create Procedure

Skenario dengan menggunakan query pembuatan stored *procedure* pada SQL Server. Ciri-ciri query ini diawali dengan CREATE PROCEDURE. Query yang digunakan pada pengujian adalah salah satu query CREATE PROCEDURE yang diambil dari database RESITS:

```
CREATE PROCEDURE Usp_mahasiswa_lulus AS
  BEGIN
      TRUNCATE TABLE mahasiswa_lulus
      INSERT INTO mahasiswa_lulus
      SELECT a.nrp,
             a.periodesem,
             b.nama,
             d.nama AS nama_prodi,
             d.jenjang
FROM [its-dw].akademik_reporting.aktivitasmhs a
JOIN [its-dw].akademik_reporting.mahasiswa b
       ON a.nrp = b.nrp
```

```
JOIN [its-dw].akademik.v_kode_fakultas_lengkap c
      ON
Substring(b.nrp,1,2)+Substring(b.nrp,5,3) =
c.prodi_kode_lama_lengkap
JOIN [its-dw].akademik.prodi d
      ON c.prodi_id = d.id
WHERE  a.statusmahasiswa = 'L'
END
```

F. Penggunaan CONCAT

Skenario dengan menggunakan query yang mengandung tag CONCAT. Query yang digunakan untuk pengujian adalah sebagai berikut:

```
select distinct m.prodi_lama, p.nama
from forlap.dbo.msmhs m
inner join [its-dw].akademik.prodi p
on m.prodi_lama =
CONCAT(p.fakultas_kode_lama,p.jurusan_kode_lama,p.pro
di_kode_lama)
order by prodi_lama
```

G. Openquery

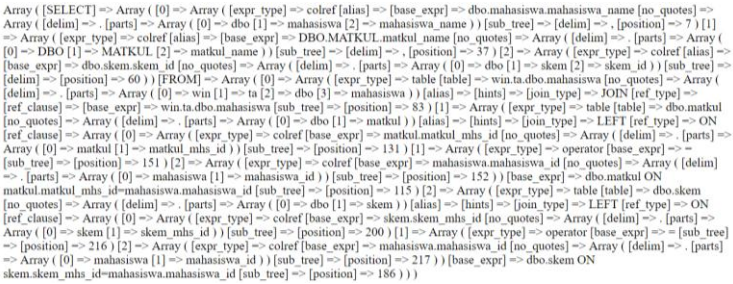
Skenario dengan menggunakan query yang mengandung tag openquery. Query yang digunakan untuk pengujian adalah sebagai berikut:

```
SELECT * FROM OPENQUERY (OracleSvr, 'SELECT
name FROM joe.titles WHERE name =
''NewTitle''');
```

Hasil Pengujian Query di atas adalah sebagai berikut:

A. Basic SELECT with JOIN

Hasil parsing dari basic SELECT query adalah sebagaimana ditunjukkan oleh gambar.


Array ([SELECT] => Array ([0] => Array ([expr_type] => colref [alias] => [base_expr] => dbo.mahasiswa.mahasiswa_name [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => mahasiswa [2] => mahasiswa_name)) [sub_tree] => [delim] => , [position] => 7) [1] => Array ([expr_type] => colref [alias] => [base_expr] => DBO.MATKUL.matkul_name [no_quotes] => Array ([delim] => . [parts] => Array ([0] => DBO [1] => MATKUL [2] => matkul_name)) [sub_tree] => [delim] => , [position] => 37) [2] => Array ([expr_type] => colref [alias] => [base_expr] => dbo.skem.skem_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => skem [2] => skem_id)) [sub_tree] => [delim] => [position] => 60)) [FROM] => Array ([0] => Array ([expr_type] => table [table] => win.ta.dbo.mahasiswa [no_quotes] => Array ([delim] => . [parts] => Array ([0] => win [1] => ta [2] => dbo [3] => mahasiswa)) [alias] => [hints] => [join_type] => JOIN [ref_type] => [ref_clause] => [base_expr] => win.ta.dbo.mahasiswa [sub_tree] => [position] => 83) [1] => Array ([expr_type] => table [table] => dbo.matkul [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => matkul)) [alias] => [hints] => [join_type] => LEFT [ref_type] => ON [ref_clause] => Array ([0] => Array ([expr_type] => colref [base_expr] => matkul.matkul_mhs_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => matkul [1] => matkul_mhs_id)) [sub_tree] => [position] => 131) [1] => Array ([expr_type] => operator [base_expr] => = [sub_tree] => [position] => 151) [2] => Array ([expr_type] => colref [base_expr] => mahasiswa.mahasiswa_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => mahasiswa [1] => mahasiswa_id)) [sub_tree] => [position] => 152)) [base_expr] => dbo.matkul ON matkul.matkul_mhs_id=mahasiswa.mahasiswa_id [sub_tree] => [position] => 115) [2] => Array ([expr_type] => table [table] => dbo.skem [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => skem)) [alias] => [hints] => [join_type] => LEFT [ref_type] => ON [ref_clause] => Array ([0] => Array ([expr_type] => colref [base_expr] => skem.skem_mhs_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => skem [1] => skem_mhs_id)) [sub_tree] => [position] => 200) [1] => Array ([expr_type] => operator [base_expr] => = [sub_tree] => [position] => 216) [2] => Array ([expr_type] => colref [base_expr] => mahasiswa.mahasiswa_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => mahasiswa [1] => mahasiswa_id)) [sub_tree] => [position] => 217)) [base_expr] => dbo.skem ON skem.skem_mhs_id=mahasiswa.mahasiswa_id [sub_tree] => [position] => 186)))

Gambar 5.6 Array hasil parsing query basic SELECT

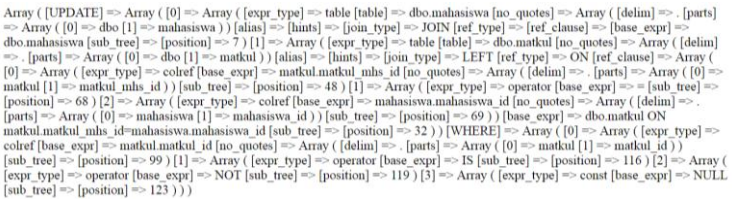
Dari query tersebut bisa diketahui letak server, database, schema, table, dan column yang terlibat JOIN. Berikut adalah posisi array dari masing-masing informasi relasi JOIN:

1. Schema pada posisi
2. Table pada posisi
3. JOIN *type* pada posisi
4. Column ON 1pada posisi
5. Column ON 2 pada posisi

Hasil dari pemahaman ini yang selanjutnya dijadikan acuan pada script parsing.

B. Basic UPDATE with JOIN

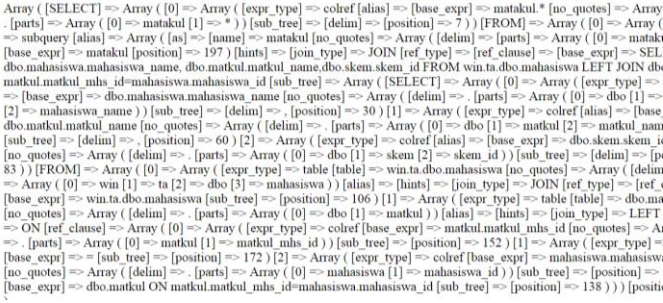
Hasil parsing dari basic SELECT query adalah sebagaimana ditunjukkan oleh gambar 5.7.


Array ([UPDATE] => Array ([0] => Array ([expr_type] => table [table] => dbo.mahasiswa [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => mahasiswa)) [alias] => [hints] => [join_type] => JOIN [ref_type] => [ref_clause] => [base_expr] => dbo.mahasiswa [sub_tree] => [position] => 7) [1] => Array ([expr_type] => table [table] => dbo.matkul [no_quotes] => Array ([delim] => . [parts] => Array ([0] => dbo [1] => matkul)) [alias] => [hints] => [join_type] => LEFT [ref_type] => ON [ref_clause] => Array ([0] => Array ([expr_type] => colref [base_expr] => matkul.matkul_mhs_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => matkul [1] => matkul_mhs_id)) [sub_tree] => [position] => 48) [1] => Array ([expr_type] => operator [base_expr] => = [sub_tree] => [position] => 68) [2] => Array ([expr_type] => colref [base_expr] => mahasiswa.mahasiswa_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => mahasiswa [1] => mahasiswa_id)) [sub_tree] => [position] => 69)) [base_expr] => dbo.matkul ON matkul.matkul_mhs_id=mahasiswa.mahasiswa_id [sub_tree] => [position] => 32)) [WHERE] => Array ([0] => Array ([expr_type] => colref [base_expr] => matkul.matkul_id [no_quotes] => Array ([delim] => . [parts] => Array ([0] => matkul [1] => matkul_id)) [sub_tree] => [position] => 99) [1] => Array ([expr_type] => operator [base_expr] => IS [sub_tree] => [position] => 116) [2] => Array ([expr_type] => operator [base_expr] => NOT [sub_tree] => [position] => 119) [3] => Array ([expr_type] => const [base_expr] => NULL [sub_tree] => [position] => 123)))

Gambar 5.7 Hasil parsing query basic UPDATE

C. Nested SELECT with SELECT

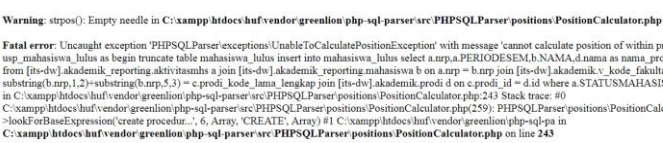
Hasil parsing dari nested SELECT with SELECT adalah sebagaimana ditunjukkan oleh gambar 5.8.



Gambar 5.8 Hasil parsing nested SELECT with SELECT

D. CREATE PROCEDURE

Hasil parsing pada query yang mengandung CREATE PROCEDURE menghasilkan tampilan seperti pada gambar 5.9. Dari gambar tersebut didapati informasi bahwa library PHP SQL Parser tidak bisa mengkalkulasi posisi pada usp_mahasiswa_lulus hingga query selesai.



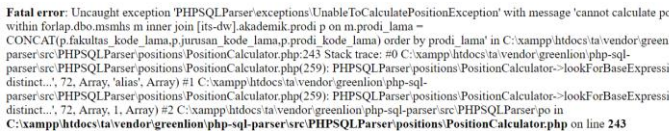
Gambar 5.9 Hasil parsing query CREATE PROCEDURE

Hal ini disebabkan karena library PHP SQL Parser tidak mendukung query yang mendukung CREATE PROCEDURE. Solusi dari permasalahan ini bisa dengan membuat algoritma tambahan untuk mendeteksi dan memisahkan query pada bagian INSERT, UPDATE, SELECT, dan beberapa query dasar yang didukung library. Query yang telah dipisahkan tersebut selanjutnya dilakukan *parsing* oleh library seperti biasa. Alasan dari cara ini adalah karena pada dasarnya query CREATE PROCEDURE ini mengandung query INSERT, UPDATE, SELECT di dalamnya. Informasi yang dibutuhkan hanya pada table dan column yang terlibat JOIN. Alternatif solusi kedua yang bisa dilakukan adalah dengan membatasi query hanya pada yang tidak terdapat CREATE PROCEDURE untuk menghindari error. Karena dihadapkan pada keterbatasan

waktu, maka solusi kedua inilah yang diterapkan pada tugas akhir ini.

E. CONCAT

Hasil parsing pada query yang mengandung Openquery menghasilkan tampilan seperti pada gambar 5.10.

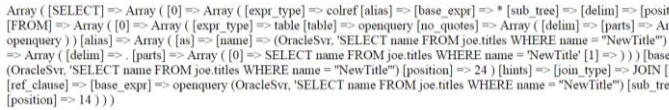


Gambar 5.10 Hasil parsing query CONCAT

Untuk menghindari error ini, ditambahkan exception untuk query CONCAT.

F. Openquery

Hasil parsing pada query yang mengandung Openquery menghasilkan tampilan seperti pada gambar 5.11.



Gambar 5.11 Hasil parsing openquery

Kesimpulan Pemahaman Array

Ada beberapa kesimpulan yang bisa diambil dari pengujian skenario query dan pemahaman array yang telah dilakukan:

1. Library tidak mendukung query CREATE PROCEDURE. Sehingga dalam query DMV yang telah direncanakan pada bab 4 ditambahkan query untuk mengecualikan query yang mengandung CREATE PROCEDURE
2. Array yang dihasilkan dari hasil SELECT dan UPDATE berbeda sehingga posisi informasi yang dibutuhkan aplikasi juga berbeda. Sehingga perlu ada proses percabangan untuk memisahkan query SELECT dan UPDATE yang selanjutnya diberi perlakuan berbeda.

3. Array hasil dari parsing nested query pada dasarnya sama dengan query biasa hanya saja posisi indeks menjadi bertambah sesuai dengan level nested arraynya.
4. Array hasil openquery pada dasarnya sama dengan nested array hanya posisi JOIN berada pada sub base_expr.
5. Library tidak mendukung query yang mengandung CONCAT.

5.4. Pembuatan Aplikasi

Pada bagian ini akan dijelaskan implementasi desain sistem pada bagian sebelumnya ke dalam bahasa pemrograman PHP yang didukung sejumlah library Javascript.

5.4.1. Pembuatan dan Scheduling script DMV

Script untuk melakukan query DMV diberi nama **dmv.php**. Script ini dimulai dengan memanggil **config.php** dengan include. Sehingga **dmv.php** memungkinkan untuk melakukan koneksi yang telah dikonfigurasi pada **config.php**. Potongan kode untuk menjalankan query DMV ditunjukkan gambar 5.5.

```
//Get DMV queries (should be scheduled per minute)
    $sql = $con->prepare("
            SELECT t.[text]
            FROM sys.dm_exec_cached_plans AS p
            INNER JOIN sys.dm_exec_query_stats AS s
               ON p.plan_handle = s.plan_handle
            CROSS APPLY sys.dm_exec_sql_text(p.plan_handle) AS t
            WHERE t.[text] LIKE N'%join%'
            AND t.[text] NOT LIKE N'%sys%'
            ORDER BY s.last_execution_time DESC;");
    $sql->execute();
```

Gambar 5.12 Kode untuk menjalankan DMV

Selanjutnya script menjalankan fungsi untuk memasukkan hasil query DMV yang didapat dari langkah sebelumnya ke dalam database MariaDB bernama Query pada tabel text. Kode program langkah ini ditunjukkan pada gambar 5.7.

```
while($row=$sql->fetch(PDO::FETCH_ASSOC)){
    //insert per row result ke MySQL
    $mysql = $cons->prepare("
        INSERT INTO `text` (`text`) VALUES ('".$row['text']."')");
    $mysql->execute();
} ?>
```

Gambar 5.13 Memasukkan hasil DMV ke tabel Query.Text

Script **dmv.php** yang telah selesai dibuat selanjutnya dijalankan pada lewat command prompt dan dijadwalkan untuk dijalankan tiap satu menit sekali. Hal ini dilakukan karena query DMV hanya bisa mendapatkan query yang baru dijalankan atau sering dijalankan. Sehingga untuk mengantisipasi query yang tidak memenuhi kriteria keduanya, script DMV harus dijalankan dengan frekuensi tinggi untuk memastikan query yang didapat adalah query terbaru.

5.4.2. Pembuatan bat script syscat.php

Script ini dimulai dengan memanggil config.php lewat perintah include. Selanjutnya script ini menjalankan query system catalog sys.databases dan sys.servers untuk mendapatkan daftar database dan server yang berasosiasi dengan sistem. Kode untuk melakukan query ini ditunjukkan pada gambar 5.8.

```
//get System Catalog Server & Database
$sql2=$con->prepare('SELECT name as srv from sys.servers');
$sql2->execute();
$sql3=$con->prepare("SELECT name as db from sys.databases where
NAME NOT LIKE '%master%' AND name NOT LIKE '%tempdb%' AND name NOT
LIKE '%reportserver%' AND name NOT LIKE '%model%' AND name NOT
LIKE '%msdb%'");
$sql3->execute();
```

Gambar 5.14 Query untuk mendapatkan nama Database dan Server

Selanjutnya dari tiap nama database yang didapat, akan digunakan sebagai masukan untuk script selanjutnya. Script selanjutnya adalah script untuk menjalankan system catalog sys.table yang digabungkan via JOIN dengan sys.column. Potongan kode fungs ini bisa dilihat di gambar 5.9.

```
//Get System Catalog (db,schema,table,column)
$sql1 = $conx->prepare('
    SELECT @@SERVERNAME as srv,DB_NAME(DB_ID()) as
    db,SCHEMA_NAME(schema_id) as sch, sys.tables.name as
    tbl,sys.columns.name as col FROM sys.tables inner join
    sys.columns on sys.tables.object_id = sys.columns.object_id');
$sql1->execute();
```

Gambar 5.15 Query untuk mendapatkan table, database, dan column

Selanjutnya, hasil dari query system catalog disimpan ke dalam graph Neo4j dengan menggunakan query cypher yang dijalankan melalui library Neo4j for PHP Client. Query dijalankan dengan menggunakan fitur stack yaitu mendaftar

dahulu query apa saja yang hendak djalanan dan dijalankan secara keseluruhan di akhir bagian. Cara penulisan di kode PHP ditunjukkan pada gambar 5.10.

```
//Fetch result System Catalog
while($row=$sql1->fetch(PDO::FETCH_ASSOC)){
    $dbname = $row['srv']."."$row['db'];
    $schname = $dbname."."$row['sch'];
    $tblname = $schname."."$row['tbl'];
    $colname = $tblname."."$row['col'];
    $stack = $neo->stack();
    //create node Schema if doesn't exist
    $stack->push('MERGE(x:Schema) SET n+={i}',['i'=>['name'=>$row['sch'],'path'=>$schname
    ]]);
    //create node Table if doesn't exist
    $stack->push('MERGE(x:Table) SET n+={i}',['i'=>['name'=>$row['tbl'],'path'=>$tblname]]);
    //create node Column if doesn't exist
    $stack->push('MERGE(x:Column) SET n+={i}',['i'=>['name'=>$row['col'],'path'=>$colname
    ]]);
    //create relationship Table to Column
    $stack->push('MATCH (a:Table {path: {path1} }),(b:Column {path: {path2} }) MERGE
    (a)-[:Col]->(b)',['path1'=>$tblname,'path2'=>$colname]);
    //create relationship Schema to Table
    $stack->push('MATCH (a:Schema {path: {path1} }),(b:Table {path: {path2} }) MERGE
    (a)-[:Tbl]->(b)',['path1'=>$schname,'path2'=>$tblname]);
    //create relationship Database to Schema
    $stack->push('MATCH (a:Database {path: {path1} }),(b:Schema {path: {path2} }) MERGE
    (a)-[:Sch]->(b)',['path1'=>$dbname,'path2'=>$schname]);
    //create relationship Server to Database
    $stack->push('MATCH (a:Server{path:{path1}}),(b:Database {path: {path2}}) MERGE
    (a)-[:Db]->(b)',['path1'=>$row['srv'],'path2'=>$dbname]);
    $neo->runStack($stack);
}
```

Gambar 5.16 Kode untuk menyimpan hirarki SQL Server ke dalam graph sebagai *node* dan *relationship*

Selanjutnya script menjalankan query untuk memperoleh foreign key relationship. Potongan kode akses query pada langkah ini bisa dilihat di gambar 5.11.

```
//Get FK Relationship from System Catalog
$sqlx = $conx->prepare('SELECT @@SERVERNAME as srv,
                DB_NAME(DB_ID()) as db,
                obj.name AS fk_rel,
                sch2.name AS par_sch,
                tab2.name AS par_tbl,
                col2.name AS par_col,
                sch1.name AS ref_sch,
                tab1.name AS ref_tbl,
                col1.name AS ref_col
            FROM sys.foreign_key_columns fkc
            INNER JOIN sys.objects obj
                ON obj.object_id = fkc.constraint_object_id
            INNER JOIN sys.tables tab1
                ON tab1.object_id = fkc.parent_object_id
            INNER JOIN sys.columns col1
                ON col1.column_id = parent_column_id AND col1.object_id = tab1.object_id
            INNER JOIN sys.tables tab2
                ON tab2.object_id = fkc.referenced_object_id
            INNER JOIN sys.columns col2
                ON col2.column_id = referenced_column_id AND col2.object_id = tab2.object_id
            INNER JOIN sys.schemas sch1
                ON tab1.schema_id = sch1.schema_id
            INNER JOIN sys.schemas sch2
                ON tab2.schema_id = sch2.schema_id');
$sqlx->execute();
```

Gambar 5.17 Kode query untuk mendapatkan foreign key relationship

Setelah informasi foreign key relationship didapat, langkah selanjutnya adalah memasukkan informasi tersebut ke dalam graph database. Diawali dengan membuka iterasi untuk semua baris didapat, program mengambil informasi yang berkaitan dengan foreign key relationship yaitu: kolom induk (*parent column*), kolom referensi (*reference column*), dan nama *constraint* relasinya. Potongan kode untuk menjalankan langkah ini ditunjukkan pada gambar 5.12.

```
while($row=$sqlx->fetch(PDO::FETCH_ASSOC)){
    $ref_col = $row['srv']."."$row['db']."."$row['ref_sch']."."$row['ref_tbl'].
    "."$row['ref_col'];
    $par_col = $row['srv']."."$row['db']."."$row['par_sch']."."$row['par_tbl'].
    "."$row['par_col'];
    //create relationship Server to Database
    $neo->run('MATCH (a:Column {path: {path1} }),(b:Column {path: {path2} })
    MERGE (a)-[:'.$row['fk_rel'].']->(b)',['path1'=>$par_col,'path2'=>$ref_col]);
}
```

Gambar 5.18 Kode untuk mendapatkan foreign key relationship

5.4.3. Pembuatan script parsing query

Script ini bertujuan untuk mengambil data dari database Query dan melakukan parsing secara iteratif untuk tiap baris query yang tersimpan. Script ini relatif lebih pendek dari script lainnya karena hanya memanggil library PHP SQL Parser dan menjalankan fitur parse. Script ini dibuat terpisah untuk mengantisipasi bentrok namespace antara namespace yang digunakan untuk memanggil PHP SQL Parser dengan namespace default. Script ini diberi nama **pars.php**. Gambar berikut menampilkan kode lengkap **pars.php**.

```php
<?php
namespace PHPSQLParser;
require 'vendor/autoload.php';

class Pars {
    public function parse($query){
        $parser = new PHPSQLParser($query, true);
        return $data = $parser->parsed;
    }
}
$pars = new Pars();
?>
```

Gambar 5.19 Kode file Pars.php

Script ini mengembalikan data hasil parsing yang selanjutnya diproses di script eksekusi parsing.

5.4.4. Pembuatan script eksekusi parsing

Parsing query yang dihasilkan oleh library PHP SQL Parser berbentuk array $data. Data yang dihasilkan akan langsung dimasukkan ke dalam graph dengan cypher query. Nama tabel, baik tabel tunggal maupun tabel yang ikut dalam JOIN berada pada subarray $data['FROM']. Sehingga script hanya akan mengeksekusi subarray $data['FROM'].

```
if($data['FROM']){
    foreach($data['FROM'] as $d){
        $join = $d['join_type'];
        $i = 1;
        foreach($d['no_quotes']['parts'] as $r){
```

Gambar 5.20 Mendapatkan subarray $data['FROM']

Informasi tabel yang mempunyai relasi JOIN berada pada $data['FROM'][‘x’][‘ join_type’]['no_quotes']['parts']. Apabila x adalah 0 maka tabel yang muncul adalah tabel setelah FROM. Apabila x adalah angka apapun selain 0, maka tabel adalah hasil JOIN. Sehingga hasil dari $data['FROM'][‘x’][‘ join_type’]['no_quotes']['parts'] ini pun masih tergabung antara schema dan tabel. Iterasi ganjil menghasilkan Schema, iterasi genap menghasilkan nama table. Sehingga perlu dilakukan percabangan untuk memisahkan ganjil dan genap. Kode untuk melakukan percabangan ini ditampilkan pada gambar.

```
foreach($d['no_quotes']['parts'] as $r){
    if($i % 2 == 0){
        $i++;
        $stack->push('MERGE (x:Table {name: {name} })',[
        'name' => trim($r,"[]")]);
    }
    else {
        $i++;
        $stack->push('
            MERGE (x:Schema {name: {name} })',
            ['name' => trim($r,"[]")]);
        $o = $r+1;
        //create relationship Schema to Table
        $stack->push('MATCH (a:Schema {name: {name1}
        }),(b:Table {name: {name2} }) MERGE
        (a)-[:Tbl]->(b)',['name1'=>$r,'name2'=>$o]);
    }
}
```

Gambar 5.21 Mendapatkan schema dan table yang terhubung JOIN

Sedangkan untuk mendapatkan kolom apa yang berada pada posisi ON dalam query digunakan sub array

$data['FROM'][‘x’]['ref_clause']['y']['no_quotes']['parts']['1']. Jika y adalah 2 maka itu adalah kolom pada tabel asal. Jika x adalah 0 maka itu adalah kolom pada tabel JOIN.

```
$stack->push('MATCH (a:Column {name: {name1} }),(b:Column
{name: {name2} }) MERGE (a)<-[:'.$d['join_type'].']->(b)',[
'name1'=>$d['ref_clause']['0']['no_quotes']['parts']['1'],
'name2'=>$d['ref_clause']['2']['no_quotes']['parts']['1']]);
```

Gambar 5.22 Mendapatkan column yang berada di posisi JOIN ON

Pada library PHP SQL Parser JOIN hanya dibedakan menjadi 3 macam yaitu: JOIN, LEFT JOIN, dan RIGHT JOIN. Jenis JOIN yang pertama termasuk di dalamnya JOIN biasa atau INNER JOIN. Outer JOIN belum didukung oleh library. Data jenis JOIN didapat dari subarray $data[‘FROM’][‘join_type’].

5.4.5. Pembuatan fungsi filtering graph

Pada tahapan dibuat kode program untuk *filtering* pada graph yang hendak ditampilkan. Fungsi ini dibuat sebagai antisipasi apabila graph yang ditampilkan berukuran besar dan membutuhkan waktu yang lama untuk di-*load* pada *client-side*. Dengan memanfaatkan fungsi *filtering* ini, graph yang ditampilkan hanya fokus pada bagian-bagian yang diminta tanpa perlu *loading* semua graph.

Fungsi ini juga memanfaatkan cypher query yang dieksekusi oleh library GraphAware Neo4j PHP Client sebagaimana juga seluruh aktivitas program sebelumnya yang berhubungan dengan graph Neo4j dieksekusi. Fungsi ini memanfaatkan tampilan yang terdiri dari input *checkbox* dan *text* sebagai sumber masukan untuk dieksekusi oleh cypher query di graph database Neo4j. Potongan kode HTML tampilan ini adalah sebagaimana yang ditunjukkan oleh gambar 6. dan 6. Di bawah ini.

```
<form method="post">
        <div class='form-group'>
         <div class="checkbox">
             <label><input type="checkbox" name="srvcheck" id='srvcheck'>
             Server
             </label>
         </div>
            <input type="text" id="srv" name="srv" placeholder="Server" class=
            "form-control" disabled>
        </div>
    <div class='form-group'>
        <div class="checkbox">
            <label><input type="checkbox" name="dbcheck" id='dbcheck' disabled>
            Database
            </label>
        </div>
        <input type="text" id="db" name="db" placeholder="Database" class=
        "form-control" disabled>
    </div>
    <div class='form-group'>
        <div class="checkbox">
            <label><input type="checkbox" name="schcheck" id='schcheck' disabled>
            Schema
            </label>
        </div>
        <input type="text" id="sch" name="sch" placeholder="Schema" class=
        "form-control" disabled>
    </div>
```

Gambar 5.23 Kode input *checkbox* dan *text*

```
<div class='form-group'>
    <div class="checkbox">
        <label><input type="checkbox" name="tblcheck" id='tblcheck'
        disabled>
        Table
        </label>
    </div>
    <input type="text" id="tbl" name="tbl" placeholder="Table" class="form-control" disabled>
</div>
<div class='form-group'>
    <div class="checkbox">
        <label><input type="checkbox" name="colcheck" id='colcheck' disabled>
        Column
        </label>
    </div>
    <input type="text" id="col" name="col" placeholder="Column" class="form-control" disabled>
</div>
<div class='checkbox'>
    <label><input type="checkbox" name="jocheck" id='jocheck' disabled>
        JOIN
    </label>
    <label><input type="checkbox" name="fkcheck" id='fkcheck' disabled>
        FK
    </label>
</div>
<div><hr>
    <div class="checkbox">
        <label><input type="checkbox" name="allcheck" id='allcheck'>
        Show All Graph
        </label>
    </div>
    <div><button type='submit' class='btn btn-primary' name='btn-gen'>GENERATE</button></div>
```

Gambar 5.24 Kode input *checkbox* dan *text* (lanjutan)

Input *text* tersebut secara *default* statusnya *disabled* atau tidak bisa diisi. Input *text* baru bisa diisi setelah input *checkbox* di bagian atasnya dicentang. Hal ini berlaku untuk input Server, Database, Schema, Table. Sedangkan untuk input Column, apabila dicentang selain juga mengaktifkan input Column di

bawahnya juga mengaktifkan *checkbox* FK dan JOIN. Berikut kode javascript yang digunakan untuk mengatur perubahan status pada input tersebut.

```
<script type="text/javascript">
document.getElementById('srvcheck').onchange = function() {
    document.getElementById('srv').disabled = !this.checked;
    document.getElementById('dbcheck').disabled = !this.checked;
};
document.getElementById('dbcheck').onchange = function() {
    document.getElementById('db').disabled = !this.checked;
    document.getElementById('schcheck').disabled = !this.checked;
};
document.getElementById('schcheck').onchange = function() {
    document.getElementById('sch').disabled = !this.checked;
    document.getElementById('tblcheck').disabled = !this.checked;
};
document.getElementById('tblcheck').onchange = function() {
    document.getElementById('tbl').disabled = !this.checked;
    document.getElementById('colcheck').disabled = !this.checked;
};
document.getElementById('colcheck').onchange = function() {
    document.getElementById('col').disabled = !this.checked;
    document.getElementById('fkcheck').disabled = !this.checked;
    document.getElementById('jocheck').disabled = !this.checked;
};
document.getElementById('allcheck').onchange = function(){
    document.getElementById('col').disabled = this.checked;
    document.getElementById('colcheck').disabled = this.checked;
    document.getElementById('tbl').disabled = this.checked;
    document.getElementById('tblcheck').disabled = this.checked;
    document.getElementById('sch').disabled = this.checked;
    document.getElementById('schcheck').disabled = this.checked;
    document.getElementById('db').disabled = this.checked;
    document.getElementById('dbcheck').disabled = this.checked;
    document.getElementById('srv').disabled = this.checked;
    document.getElementById('srvcheck').disabled = this.checked;
    document.getElementById('fkcheck').disabled = this.checked;
    document.getElementById('jocheck').disabled = this.checked;
}
</script>
```

Gambar 5.25 Kode javascript untuk mengatur status input

Masing-masing *checkbox* memiliki script untuk eksekusi cypher query khusus yang bisa mengakomodasi jenis input yang disesuaikan dengan jenis node-nya. Berikut ini kode yang ada pada masing-masing *checkbox.*

```
if(isset($_POST['btn-gen'])){
    if(isset($_POST['allcheck'])){
        $resnode = $neo->run("match (n) return n.name as name,id(n) as id,labels(n) as label")
        $resrel = $neo->run("MATCH (a)-[r]->(b) return id(r) as id,id(a) as start,id(b) as
        end, type(r) as type");
    }
    //jika server terisi
    if(isset($_POST['srv'])){
        $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
        return n.name as name,id(n) as id,labels(n) as label");
    }
    //jika input database tercentang tapi tidak terisi
    if(isset($_POST['dbcheck']) && empty($_POST['db'])){
        $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
        return n.name as name,id(n) as id,labels(n) as label UNION match (n:Server)-[r]->(m)
        where n.name contains '".$_POST['srv']."' return m.name as name,id(m) as id,labels(m)
        as label");
        $resrel = $neo->run("MATCH (a)-[r:Db]->(b) where a.name contains '".$_POST['srv']."'
        return id(r) as id,id(a) as start,id(b) as end, type(r) as type");
    }
    //jika input database tercentang dan terisi
    if(isset($_POST['db'])){
        $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
        return n.name as name,id(n) as id,labels(n) as label UNION match (n:Server)-[r]->(m)
        where n.name contains '".$_POST['srv']."' AND m.name contains '".$_POST['db']."'
        return m.name as name,id(m) as id,labels(m) as label");
        $resrel = $neo->run("MATCH (a)-[r:Db]->(b) where a.name contains '".$_POST['srv']."'
        and b.name contains '".$_POST['db']."' return id(r) as id,id(a) as start,id(b) as end,
        type(r) as type");
```

Gambar 5.26 Kode filtering I

```
if(isset($_POST['schcheck']) && empty($_POST['sch'])){
    $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
    return n.name as name,id(n) as id,labels(n) as label union match (n:Server)-[r]->(m)
    where n.name contains '".$_POST['srv']."' return m.name as name,id(m) as id,labels(m)
    as label union match (n:Server)-[r]->(m) where n.name contains '".$_POST['srv']."' AND
    m.name contains '".$_POST['db']."' return m.name as name,id(m) as id,labels(m) as
    label UNION match (m:Database)-[r]->(o) where m.name contains '".$_POST['db']."'
    return id(o) as id,o.name as name,labels(o) as label");
    $resrel = $neo->run("MATCH (a)-[r:Db]->(b) where a.name contains '".$_POST['srv']."'
    return id(r) as id,id(a) as start,id(b) as end, type(r) as type UNION MATCH
    (b)-[r:Sch]->(c) where b.name contains '".$_POST['db']."' return id(r) as id,id(b) as
    start,id(c) as end, type(r) as type");
}
//jika input Table tercentang tapi tidak terisi
if(isset($_POST['tblcheck']) && empty($_POST['tbl'])){
    $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
    return n.name as name,id(n) as id,labels(n) as label union match (n:Server)-[r]->(m)
    where n.name contains '".$_POST['srv']."' return m.name as name,id(m) as id,labels(m)
    as label union match (n:Server)-[r]->(m) where n.name contains '".$_POST['srv']."' AND
    m.name contains '".$_POST['db']."' return m.name as name,id(m) as id,labels(m) as
    label UNION match (m:Database)-[r]->(o) where m.name contains '".$_POST['db']."'
    return id(o) as id,o.name as name,labels(o) as label union match (o:Schema)-[r]->(p)
    where o.name contains '".$_POST['sch']."' return p.name as name,id(p) as id,labels(p)
    as label");
    $resrel = $neo->run("MATCH (a)-[r:Db]->(b) where a.name contains '".$_POST['srv']."'
    return id(r) as id,id(a) as start,id(b) as end, type(r) as type UNION MATCH
    (b)-[r:Sch]->(c) where b.name contains '".$_POST['db']."' return id(r) as id,id(b) as
    start,id(c) as end, type(r) as type UNION MATCH (c)-[r:Tbl]->(d) where c.name contains
    '".$_POST['sch']."' return id(r) as id,id(c) as start,id(d) as end, type(r) as type");
```

Gambar 5.27 Kode filtering II

```
//jika input Column tercentang tapi tidak terisi
if(isset($_POST['colcheck']) && empty($_POST['col'])){
    $resnode = $neo->run("match (n:Server) where n.name contains '".$_POST['srv']."'
    return n.name as name,id(n) as id,labels(n) as label union match (n:Server)-[r]->(m)
    where n.name contains '".$_POST['srv']."' return m.name as name,id(m) as id,labels(m)
    as label union match (n:Server)-[r]->(m) where n.name contains '".$_POST['srv']."' AND
    m.name contains '".$_POST['db']."' return m.name as name,id(m) as id,labels(m) as
    label UNION match (m:Database)-[r]->(o) where m.name contains '".$_POST['db']."'
    return id(o) as id,o.name as name,labels(o) as label union match (o:Schema)-[r]->(p)
    where o.name contains '".$_POST['sch']."' return p.name as name,id(p) as id,labels(p)
    as label union match (o:Schema)-[:Tbl]->(p:Table)-[:Col]->(q) where o.name contains '".
    $_POST['sch']."' and p.name contains '".$_POST['tbl']."' return q.name as name,id(q)
    as id,labels(q) as label");
    $resrel = $neo->run("MATCH (a)-[r:Db]->(b) where a.name contains '".$_POST['srv']."'
    return id(r) as id,id(a) as start,id(b) as end, type(r) as type UNION MATCH
    (b)-[r:Sch]->(c) where b.name contains '".$_POST['db']."' return id(r) as id,id(b) as
    start,id(c) as end, type(r) as type UNION MATCH (c)-[r:Tbl]->(d) where c.name contains
    '".$_POST['sch']."' return id(r) as id,id(c) as start,id(d) as end, type(r) as type
    UNION MATCH (c)-[:Tbl]->(d)-[r:Col]->(e) where c.name contains '".$_POST['sch']."' and
    d.name contains '".$_POST['tbl']."' return id(r) as id,id(d) as start,id(e) as end,
    type(r) as type");
}
```

Gambar 5.28 Kode filtering III

Seperti yang terlihat pada 3 gambar di atas, terdapat 2 variable yaitu $resnode dan $resrel yang menampung hasil dari query yang dijalankan. Variabel $resnode untuk menampung data node dan variabel $resrel untuk menampung data relationship. Hasil dari input tersebut akan dieksekusi oleh library apabila tombol GENERATE ditekan.

5.4.6. Pembuatan script PHP generate JSON

Script ini diawali dengan menjalankan cypher query untuk mendapatkan data id, nama untuk node, dan data ide, type, end_id dan start_id untuk relationship. Diawali dengan mengakses data seluruh node. Data node yang didapat langsung diiterasi dan dimasukkan ke dalam array $nodes[]. Kode untuk menyimpan data node ke dalam array tertera sebagaimana pada gambar 5.17.

```
//Generate JSON from all Nodes
foreach ($resnode->getRecords() as $record){
    $nodes[] = ["id"=>$record->value('id'),
                "labels"=>$record->value('label'),
                "properties"=>[
                    "name"=> $record->value('name')
                    //taruh value lain di sini (jika ada)
                ]
    ];
}
//Generate JSON from all Relationships
if(isset($resrel)){
    if($resrel->getRecords()==NULL){
        $rel = [];
    }else{
        foreach ($resrel->getRecords() as $record) {
            $rel[] = ["id"=>$record->value('id'),
                "type"=>$record->value('type'),
                "startNode"=>$record->value('start'),
                "endNode"=>$record->value('end'),
                "properties"=>array()
            ];
        }
    }
```

Gambar 5.29 Kode untuk menyimpan data nodes

Setelah mendapatkan data node dan memasukkannya ke dalam array, langkah selanjutnya adalah mendapatkan data relationship. Dan sebagaimana data node, data relationship juga diiterasi dan disimpan di dalam array tersendiri yaitu $rel[]. Kode untuk menjalankan langkah ini tertera pada gambar 5.18.

Array $node[] dan $rel[] selanjutnya disatukan dengan posisi berurutan di dalam subarray $json[‘results’][‘data’][‘graph’] sesuai ketentuan dari library Neo4jD3.js. Selanjutnya dilakukan *encode* pada array $json lalu dijadikan file JSON yang bernama **neodata.json**. Kode untuk menjalankan perintah ini ditunjukkan oleh gambar 5.19.

```
//Generate JSON from all Nodes
foreach ($resnode->getRecords() as $record){
    $nodes[] = ["id"=>$record->value('id'),
                "labels"=>$record->value('label'),
                "properties"=>[
                    "name"=> $record->value('name')
                    //taruh value lain di sini (jika ada)
                    ]
                ];
}
//Generate JSON from all Relationships
foreach ($resrel->getRecords() as $record) {
    $rel[]= ["id"=>$record->value('id'),
             "type"=>$record->value('type'),
             "startNode"=>$record->value('start'),
             "endNode"=>$record->value('end'),
             "properties"=>array()
             ];
}
$json = ["results" => array([
             "data" => array([
             "graph" => array(
             "nodes" => $nodes,
             "relationships" => $rel
             )])])];
file_put_contents('neodata.json', json_encode($json));
```

Gambar 5.30 Kode untuk menyatukan array $nodes[] & $rel[] dan menghasilkan file JSON

5.4.7. Pembuatan script Graph Loader berbasis D3.js

Script *graph loader* ini berfungsi sebagai aplikasi penampil data graph Neo4j ke dalam bentuk diagram yang bisa dilihat melalui browser HTML lewat port HTTP. Pada pengerjaan aplikasi, script ini diberi nama **graph.php.** Script ini adalah script yang dirancang untuk dijalankan di browser, tidak seperti beberapa script sebelumnya. Script ini dimulai dengan menyiapkan library yang dibutuhkan seperti yang tertera pada gambar 5.17.

```
<head>
    <title>GRAPH RESULT</title>
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="css/neo4jd3.min.css">
    <link rel="stylesheet" type="text/css" href="css/font-awesome.min.css">
    <script type="text/javascript" src='js/bootstrap.min.js'></script>
    <script type="text/javascript" src='js/d3.min.js'></script>
    <script type="text/javascript" src='js/neo4jd3.min.js'></script>
    <style>
```

Gambar 5.31 Library yang disiapkan pada Graph.php

Selanjutnya didefinisikan icon apa saja yang akan digunakan pada masing-masing jenis node. Icon yang digunakan adalah

icon yang tersedia pada library FontAwesome. Potongan kode program untuk langkah ini ada pada gambar 5.31.

```
icons: {
    'Server': 'server',
    'Database': 'database',
    'Schema': 'gear',
    'Table': 'table',
    'Column': 'columns'
},
```

Gambar 5.32 Pendefinisian Icon yang digunakan

5.4.8. Otomatisasi dan Penjadwalan

Script **dmv.php** dan **syscat.php** dijalankan lewat command prompt. File batch (.bat) dibuat untuk menjalankan masing-masing script. File **dmv.bat** ditulis dengan perintah berikut:

```
C:\xampp\php\php.exe -f
C:\xampp\htdocs\huf\dmv.php
```

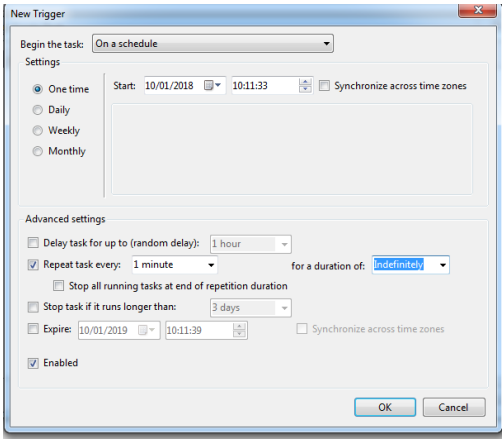
Sedangkan file **syscat.bat** ditulis dengan perintah berikut:

```
C:\xampp\php\php.exe -f
C:\xampp\htdocs\huf\syscat.php
```

Sedangkan file **parse.bat** ditulis dengan perintah berikut:

```
C:\xampp\php\php.exe -f
C:\xampp\htdocs\huf\parse.php
```

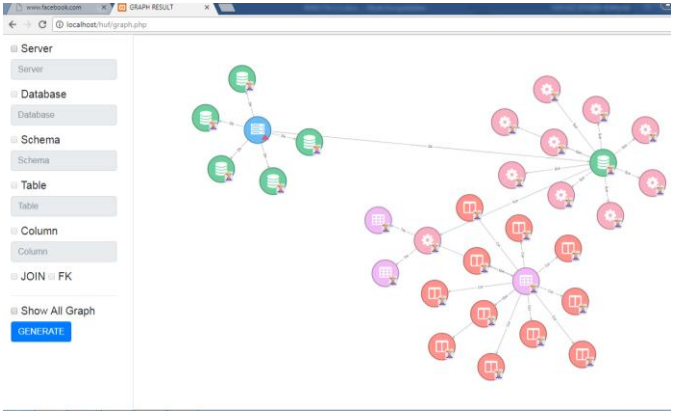
Selanjutnya kedua file .bat ini diatur agar berjalan otomatis dengan frekuensi tiap menit sekali untuk **dmv.bat** dan tiap hari sekali untuk **syscat.bat**. Agar kedua batch file ini bisa berjalan otomatis sesuai jadwal yang diinginkan, kedua file ini perlu didaftarkan pada Windows Task Scheduler dan perlu dibuat trigger baru. Isikan di bagian ‘Repeat task every’ dengan 1 minute untuk **dmv.bat** dan 1 day untuk **syscat.bat**. Pengaturan lengkap trigger bisa dilihat pada gambar 5.32.



Gambar 5.33 Membuat trigger baru pada Task Scheduler

5.5. Implementasi tampilan antarmuka

Berikut adalah hasil implementasi antarmuka aplikasi visualisasi database SQL Server dengan Dynamic Management View berbasis graph Neo4j.



Gambar 5.34 Hasil implementasi antarmuka aplikasi

Pada tampilan antarmuka aplikasi ada 3 buah input teks yang nampak *disable* dan 3 buah *checkbox*. Apabila checkbox dengan nama yang sama dengan input teks dicentang, maka

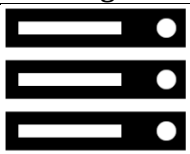
input teks akan menjadi *enabled* sehingga bisa diisi sesuatu. Selain itu terdapat *checkbox* Show All Graph yang berada di samping tombol biru bertulisan generate. Apabila *checkbox* Show All Graph dicentang, maka seluruh input teks di atasnya akan menjadi *disable.*

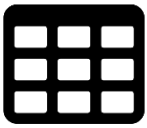
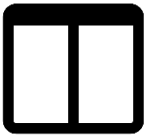
Beberapa jenis input di atas digunakan untuk melakukan filtering data graph yang hendak ditampilkan. *Checkbox* Show All Graph dicentang apabila ingin menampilkan seluruh nodes dan relationships yang ada di dalam data graph Neo4j. Sedangkan input teks di atasnya digunakan untuk menyaring data pada hirarki database tertentu.

5.6. Cara membaca tampilan graph diagram

Graph diagram yang dihasilkan memiliki beberapa simbol yang mewakili masing-masing hirarki database. Penjelasan masing-masing logo yang digunakan akan ditampilkan pada tabel 5.3.

Tabel 5.3 Penjelasan simbol Aplikasi

Logo	Arti
	Instance/Server
	Database
	Schema

	Table
	Column

Antar node dihubungkan oleh relasi berbentuk garis. Hubungan antar Server dan Database disebut :Db. Hubungan antar Database dan Schema disebut :Sch. Hubungan antar Schema dan Table disebut :Tbl. Hubungan antar Table dan Column disebut :Col. Hubungan antar kolom bisa berupa :FK_xxx untuk foreign key relationship atau :JOIN untuk JOIN relationship.

Halaman sengaja dikosongi

BAB VI

PENGUJIAN

Bab ini akan membahas pengujian aplikasi yang telah selesai diimplementasi. Pengujian aplikasi mengacu pada beberapa poin: kebutuhan fungsional dan tingkat keberhasilan dalam mendeteksi struktur database SQL Server dari dynamic management view.

6.1. Pengujian Kebutuhan Fungsional

Pengujian tahap pertama ini memastikan seluruh kebutuhan fungsional terpenuhi dan berjalan tanpa *error*. Pengujian dilakukan pada 2 database yang berbeda yaitu: *sample* database Sales dan database RESITS. Database Sales adalah contoh database SQL Server berskala kecil. Digunakan untuk menguji kebutuhan fungsional aplikasi. Sedangkan database RESITS digunakan untuk menguji kemampuan aplikasi untuk memvisualisasikan database berukuran besar.

Tabel 6.1 Hasil pengujian kebutuhan fungsional

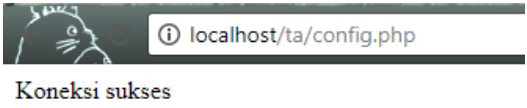
No.	Kebutuhan Fungsional	Status
1	Aplikasi dapat melakukan koneksi ke SQL Server 2016	**Sukses**
2	Aplikasi dapat mengambil informasi query dari fitur Dynamic Management View.	**Sukses**
3	Aplikasi dapat mengambil informasi instance (linked server, remote server, local server) dan daftar database yang tersedia dari fitur System Catalog.	**Sukses**
4	Aplikasi dapat menyimpan informasi query DMV pada kebutuhan No.2 ke dalam database MariaDB.	**Sukses**

5	Aplikasi dapat mengambil informasi schema, table, dan column pada suatu database dengan menggunakan fitur System Catalog.	**Sukses**
6	Aplikasi dapat mengambil informasi instance dan database yang berasosiasi dengan schema, table, dan column yang didapat dari kebutuhan No. 5 dengan menggunakan fitur metadata dan configuration pada SQL Server 2016.	**Sukses**
7	Aplikasi dapat mengambil informasi relasi berbasis Foreign Key pada seluruh tabel yang didapat pada kebutuhan No. 5.	**Sukses**
8	Aplikasi dapat menampilkan informasi relasi berbasis *foreign key* pada database.	**Sukses**
9	Aplikasi dapat menjalankan kebutuhan No. 5,6,7,8 pada semua database yang didapat dari kebutuhan No. 3.	**Sukses**
10	Aplikasi dapat mengambil query DMV yang telah disimpan pada database MariaDB.	**Sukses**
11	Aplikasi dapat mengambil informasi schema, table, column, yang memiliki relasi JOIN dari hasil parsing query DMV yang didapat dari kebutuhan No. 10.	**Sukses**
12	Aplikasi dapat menyimpan data server, database, schema, table, column, foreign key realtionship, dan	**Sukses**

	JOIN relationship ke dalam graph database Neo4j	
13	Aplikasi dapat melakukan filtering untuk menampilkan informasi pada pilihan database, schema, table, atau column tertentu.	**Sukses**
14	Aplikasi dapat menghasilkan array sesuai standar data output library Neo4jD3.js dan menyimpannya ke dalam sebuah file JSON.	**Sukses**
15	Aplikasi dapat mengolah file JSON yang telah dibuat pada kebutuhan No. 13 menjadi bentuk graph diagram lengkap dengan seluruh informasi node dan relationship.	**Sukses**

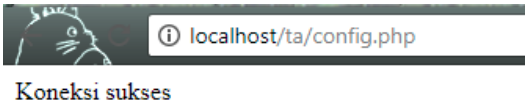
6.1.1. Aplikasi dapat melakukan koneksi ke SQL Server

Pengujian koneksi SQL Server pada database Sales yang diinstal di *local server*. Pengujian dilakukan hanya pada script **config.php**. Pada baris ke-16 script config.php ditambahkan echo 'Koneksi sukses' untuk mencetak tanda pada layar apabila koneksi sukses. Apabila koneksi gagal, maka hasil dari *catch* akan mencetak *error* apa yang terjadi. Pengujian pertama dilakukan pada database lokal. Detil nama database, username, dan password dimasukkan ke dalam variable yang telah disediakan pada script **config.php.** Nampak pada gambar 6.1 koneksi ke database lokal berhasil dilakukan.



Gambar 6.1 Pengujian koneksi database lokal Sales

Pengujian koneksi pada database RESITS membutuhkan koneksi intranet lingkungan Institut Teknologi Sepuluh Nopember (ITS). Kondisi ini bisa didapat dengan melakukan uji coba di lingkungan kampus ITS atau menggunakan akses VPN dari ITS yang bisa didapat dari akses INTEGRA (http://integra.its.ac.id/app.php). Nampak pada gambar 6.2 koneksi ke database RESITS juga berhasil dilakukan.

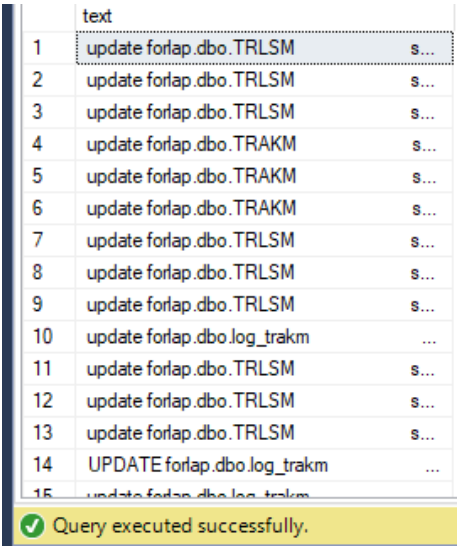


Gambar 6.2 Hasil pengujian koneksi ke database RESITS

Dari kedua pengujian di atas dapat dibuktikan bahwa file **config.php** yang bertanggung jawab untuk melakukan koneksi dan konfigurasi kebutuhan koneksi tidak ada masalah.

6.1.2. Aplikasi dapat mengambil informasi query dari fitur Dynamic Management View.

Pengujian pada tahap ini dilakukan dengan menguji query pada SQL Management Studio. Gambar 6.3 menunjukkan bahwa query DMV yang dijalankan pada SQL Management Studio berhasil mengambil teks query yang pernah dijalankan pada system RESITS. Pengujian pada query menunjukkan pengambilan informasi query dengan DMV sukses dilakukan. Sedangkan pengujian pada level aplikasi diikutkan pada pengujian tahap-tahap setelahnya.



Gambar 6.3 Pengujian DMV pada SQL Server Management Studio

6.1.3. Aplikasi dapat mengambil informasi instance (linked server, remote server, local server) dan daftar database yang tersedia dari fitur System Catalog.

Pengujian ini dilakukan dengan mengeksekusi query melalui SQL Server Management Studio. Apabila koneksi ke SQL Server pada pengujian 6.1.1 berhasil dilakukan maka bisa dipastikan query yang digunakan juga berhasil dijalankan. Query pada tahapan ini dijalankan pada database master yang merupakan database induk dari suatu *instance*.

Hasil pengujian pada database RESITS ditunjukkan pada gambar 6.4 untuk daftar instance dan gambar 6.5 untuk daftar database. Dari hasil pengujian pada database Sales dan RESITS bisa disimpulkan bahwa aplikasi berhasil mendapatkan informasi server dan database.

	srv
1	AKADEMIK
2	AKADEMIK-1
3	AKADEMIK-2
4	AKADEMIK-3
5	INTEGRA
6	REPORTING_AKADEMIK
7	SILACAK
8	SIMPEG
9	SIMPEL
10	SIPMABA
11	SKEM
12	SPMI
13	WINDOWS-1K4UUD7

Gambar 6.4 Daftar server yang didapat dari sys.servers

	db
1	data_api
2	forlap
3	its-dw
4	its-report
5	resits
6	spmi_v...
7	TEST

Gambar 6.5 Daftar server yang didapat dari sys.databases

6.1.4. Aplikasi dapat menyimpan informasi query DMV pada kebutuhan No.2 ke dalam database MariaDB.

Pengujian ini dilakukan dengan memeriksa database Query pada table text. Apabila setelah dijalankan **dmv.bat** kolom text pada table text terisi oleh query maka ini membuktikan bahwa aplikasi telah berhasil menyimpan query DMV.

Gambar 6.6 Query DMV yang disimpan di dalam MariaDB

Untuk memastikan database berhasil memasukkan semua hasil DMV ke dalam database MariaDB perlu dilakukan perbandingan

6.1.5. Aplikasi dapat mengambil informasi schema, table, dan column pada suatu database dengan menggunakan fitur System Catalog.

Pengujian pada bagian ini dilakukan dengan mengeksekusi query system catalog yang telah dijelaskan pada bab 4 lewat SQL Server Management Studio.

6.1.6. Aplikasi dapat mengambil informasi instance dan database yang berasosiasi dengan schema, table, dan column yang didapat dari kebutuhan No. 5 dengan menggunakan fitur metadata dan configuration pada SQL Server 2016.

Pengujian pada bagian ini hanya dengan menggunakan SQL Server Management Studio dengan mengeksekusi query system catalog, metadata, dan configuration seperti yang sudah dibahas pada bab 4. Hasil pengujian pada database Sales ditunjukkan oleh gambar 6.6.

	srv	db	sch	tbl	col
1	DESKTOP-EUD5PCH	salesdb	dbo	Sales	CustomerID
2	DESKTOP-EUD5PCH	salesdb	dbo	Sales	ProductID
3	DESKTOP-EUD5PCH	salesdb	dbo	Sales	Quantity
4	DESKTOP-EUD5PCH	salesdb	dbo	Sales	SalesID
5	DESKTOP-EUD5PCH	salesdb	dbo	Sales	SalesPersonID
6	DESKTOP-EUD5PCH	salesdb	dbo	Products	Name
7	DESKTOP-EUD5PCH	salesdb	dbo	Products	Price
8	DESKTOP-EUD5PCH	salesdb	dbo	Products	ProductID
9	DESKTOP-EUD5PCH	salesdb	dbo	Customers	CustomerID
10	DESKTOP-EUD5PCH	salesdb	dbo	Customers	FirstName
11	DESKTOP-EUD5PCH	salesdb	dbo	Customers	LastName
12	DESKTOP-EUD5PCH	salesdb	dbo	Customers	MiddleInitial
13	DESKTOP-EUD5PCH	salesdb	dbo	Employees	EmployeeID
14	DESKTOP-EUD5PCH	salesdb	dbo	Employees	FirstName
15	DESKTOP-EUD5PCH	salesdb	dbo	Employees	LastName
16	DESKTOP-EUD5PCH	salesdb	dbo	Employees	MiddleInitial

Gambar 6.7 Informasi server, database, schema, table, column pada database Sales

Pengujian pada database forlap sebagaimana ditunjukkan oleh gambar 6.7 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	bobotnilai_tbbnl
2	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	kode_jenjang
3	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	kode_prodi
4	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	kode_pt
5	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	mode_update
6	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	nilaiangkamax_tbbnl
7	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	nilaiangkamin_tbbnl
8	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	nilaihuruf_tbbnl
9	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	prodi_its
10	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	prodi_its_baru
11	WINDOWS-1K4UUD7	forlap	dbo	antrian_tbbnl	tanggal_update

Gambar 6.8 Informasi server, database, schema, table, dan column pada database forlap

Pengujian pada database data_api sebagaimana ditunjukkan oleh gambar 6.8 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	data_api	dbo	jumlahtendikbelajar	id
2	WINDOWS-1K4UUD7	data_api	dbo	jumlahtendikbelajar	jumlahtendikbelajar
3	WINDOWS-1K4UUD7	data_api	dbo	jumlahtendikbelajar	satker
4	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	kontinuitas
5	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	nama_dept
6	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	nama_dosen
7	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	nama_mk
8	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	nip
9	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	sks_kuliah
10	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	smt
11	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	thn_ajaran
12	WINDOWS-1K4UUD7	data_api	dbo	history_sks_mengajar	tingkat

Gambar 6.9 Informasi server, database, schema, table, dan column pada database data_api

Pengujian pada database its-dw sebagaimana ditunjukkan oleh gambar 6.9 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	its-dw	akademik	jalur_diterima	id
2	WINDOWS-1K4UUD7	its-dw	akademik	jalur_diterima	jalur
3	WINDOWS-1K4UUD7	its-dw	akademik	jalur_diterima	jalur_kode_lama
4	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	agama_id
5	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	id
6	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	jalur_diterima_id
7	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	jk_id
8	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	kota_lahir_id
9	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	nama
10	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	nama_ayah
11	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	nama_ibu
12	WINDOWS-1K4UUD7	its-dw	akademik	mahasiswa	nik

Gambar 6.10 Informasi server, database, schema, table, dan column pada database its-dw

Pengujian pada database its-report sebagaimana ditunjukkan oleh gambar 6.10 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	jenis kelamin
2	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	jumlah tendik
3	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kategori kepuasan
4	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepentingan <3
5	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepentingan >3
6	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepentingan 3
7	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepuasan <3
8	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepuasan >3
9	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	kepuasan 3
10	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	nama kepuasan
11	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	pangkat
12	WINDOWS-1K4UUD7	its-report	kepegawaian	rep_kepuasan_tendik	rata-rata skala kepentingan

Gambar 6.11 Informasi server, database, schema, table, dan column pada database its-report

Pengujian pada database resits sebagaimana ditunjukkan oleh gambar 6.11 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	resits	dbo	tran_json_node	id_node
2	WINDOWS-1K4UUD7	resits	dbo	tran_json_node	kode_pegawai
3	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_dosen_peneliti	bobot_peneliti
4	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_dosen_peneliti	bobot_topik
5	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_dosen_peneliti	final
6	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_dosen_peneliti	kode_pegawai1
7	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_dosen_peneliti	kode_pegawai2
8	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_kata_dokumen	kata
9	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_kata_dokumen	kode_pegawai
10	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_kata_dokumen	tf
11	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_kata_dokumen	tf_idf_norm
12	WINDOWS-1K4UUD7	resits	dbo	tran_bobot_kata_dokumen	tf_norm

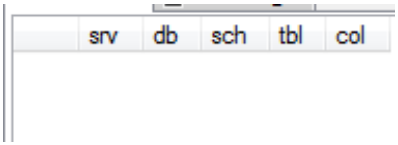
Gambar 6.12 Informasi server, database, schema, table, dan column pada database resits

Pengujian pada database spmi_view sebagaimana ditunjukkan oleh gambar 6.12 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.

	srv	db	sch	tbl	col
1	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	enabled
2	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	flag_btsi
3	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	flag_reviewer
4	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	flag_user
5	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	indikator
6	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	kriteria1
7	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	kriteria2
8	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	kriteria3
9	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	kriteria4
10	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	kriteria5
11	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	nilai1
12	WINDOWS-1K4UUD7	spmi_view	spmi	penilaian_indikator	nilai2

Gambar 6.13 Informasi server, database, schema, table, dan column pada database spmi_view

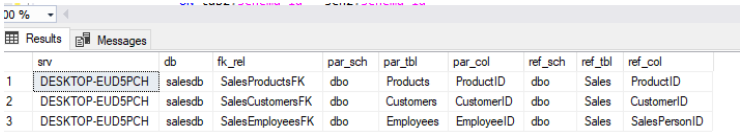
Pengujian pada database TEST sebagaimana ditunjukkan oleh gambar 6.13 berhasil mengambil informasi asosiasi server, database, schema, table, dan column di dalamnya.



Gambar 6.14 Informasi server, database, schema, table, dan column pada database TEST

6.1.7. Aplikasi dapat menampilkan informasi relasi berbasis foreign key pada database.

Pengujian pada bagian ini cukup hanya dengan memanfaatkan SQL Server Management Studio dengan mengeksekusi query system catalog untuk mengambil foreign key relationship seperti yang tertulis pada bab IV. Jika query berhasil dijalankan, maka aplikasi kemungkinan besar akan bisa mengambil data tersebut. Pengujian pertama dilakukan pada database Sales. Gambar 6. berikut menampilkan hasil dari query tersebut.

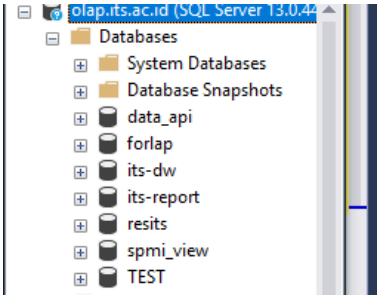

00 %

Results | Messages

	srv	db	fk_rel	par_sch	par_tbl	par_col	ref_sch	ref_tbl	ref_col
1	DESKTOP-EUD5PCH	salesdb	SalesProductsFK	dbo	Products	ProductID	dbo	Sales	ProductID
2	DESKTOP-EUD5PCH	salesdb	SalesCustomersFK	dbo	Customers	CustomerID	dbo	Sales	CustomerID
3	DESKTOP-EUD5PCH	salesdb	SalesEmployeesFK	dbo	Employees	EmployeeID	dbo	Sales	SalesPersonID

Gambar 6.15 Informasi relationship yang dihasilkan dari system catalog

Sedangkan untuk database RESITS, karena pada source code **syscat.php** terdapat eksekusi yang iteratif untuk tiap database pada sistem server dimana database RESITS berada, maka pengujian dilakukan pada semua database selain database system yaitu database: data-api, forlap, resits, its-dw, its-report, spmi-view, dan TEST.



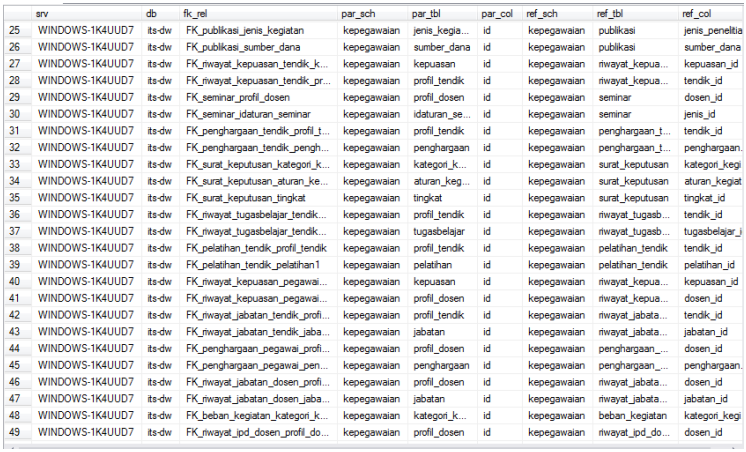
Gambar 6.16 Daftar database pada server lokasi RESITS

Hasil pengujian database pada sistem RESITS ditampilkan pada gambar-gambar di bawah. Pengujian pertama adalah pada database its-dw. Hasil pengujian ditampilkan pada gambar 6.7, 6.8, 6.9, 6.10 di bawah ini.

Results | Messages

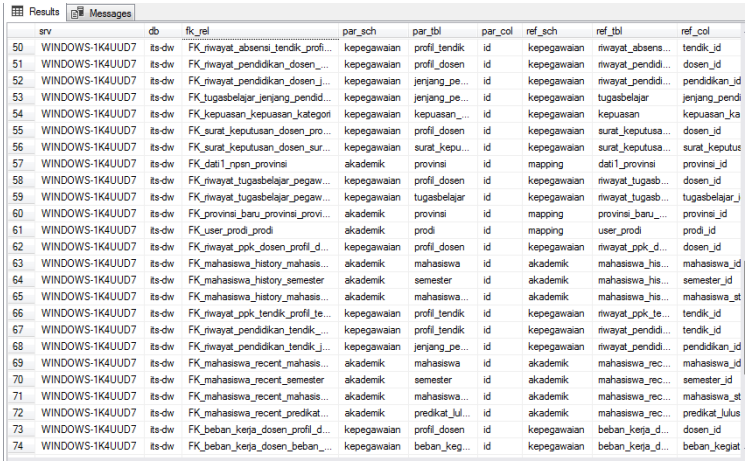
	srv	db	fk_rel	par_sch	par_tbl	par_col	ref_sch	ref_tbl	ref_col
1	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_prodi	akademik	prodi	id	akademik	mahasiswa	prodi_id
2	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_jenis_kelamin	akademik	jenis_kelamin	id	akademik	mahasiswa	jk_id
3	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_kota	akademik	kota	id	akademik	mahasiswa	kota_lahir_id
4	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_provinsi	akademik	provinsi	id	akademik	mahasiswa	provinsi_lahir_
5	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_jalur_diterima	akademik	jalur_diterima	id	akademik	mahasiswa	jalur_diterima_
6	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_jalur_diterima1	akademik	jalur_diterima	id	akademik	mahasiswa	jalur_diterima_
7	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_agama	master	agama	id	akademik	mahasiswa	agama_id
8	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_kota1	akademik	kota	id	akademik	mahasiswa	slta_kota_id
9	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_provinsi1	akademik	provinsi	id	akademik	mahasiswa	slta_provinsi_i
10	WINDOWS-1K4UUD7	its-dw	FK_kota_lahir_tendik_profil_tendik	kepegawaian	profil_tendik	id	kepegawaian	kota_lahir_tendik	tendik_id
11	WINDOWS-1K4UUD7	its-dw	FK_kota_provinsi	akademik	provinsi	id	akademik	kota	provinsi_id
12	WINDOWS-1K4UUD7	its-dw	FK_dosen_profil_dosen	kepegawaian	profil_dosen	id	master	dosen	dosen_id
13	WINDOWS-1K4UUD7	its-dw	FK_dosen_status_kawin	kepegawaian	status_kawin	id	master	dosen	status_kawin_
14	WINDOWS-1K4UUD7	its-dw	FK_dosen_satuan_kerja	kepegawaian	satuan_kerja	id	master	dosen	satuankerja_i
15	WINDOWS-1K4UUD7	its-dw	FK_dosen_pangkat	kepegawaian	pangkat	id	master	dosen	pangkat_id
16	WINDOWS-1K4UUD7	its-dw	FK_dosen_status	kepegawaian	status	id	master	dosen	status_id
17	WINDOWS-1K4UUD7	its-dw	FK_dosen_status_aktif	kepegawaian	status_aktif	id	master	dosen	statusaktif_id
18	WINDOWS-1K4UUD7	its-dw	FK_tendik_profil_tendik	kepegawaian	profil_tendik	id	master	tendik	tendik_id
19	WINDOWS-1K4UUD7	its-dw	FK_tendik_status_kawin	kepegawaian	status_kawin	id	master	tendik	status_kawin_
20	WINDOWS-1K4UUD7	its-dw	FK_tendik_satuan_kerja	kepegawaian	satuan_kerja	id	master	tendik	satuankerja_i
21	WINDOWS-1K4UUD7	its-dw	FK_tendik_pangkat	kepegawaian	pangkat	id	master	tendik	pangkat_id
22	WINDOWS-1K4UUD7	its-dw	FK_tendik_status	kepegawaian	status	id	master	tendik	status_id
23	WINDOWS-1K4UUD7	its-dw	FK_tendik_status_aktif	kepegawaian	status_aktif	id	master	tendik	statusaktif_id
24	WINDOWS-1K4UUD7	its-dw	FK_publikasi_aturan_kegiatan	kepegawaian	aturan_keg...	id	kepegawaian	publikasi	aturan_kegiat
25	WINDOWS-1K4UUD7	its-dw	FK_publikasi_jenis_kegiatan	kepegawaian	jenis_kegia...	id	kepegawaian	publikasi	jenis_penelitia

Gambar 6.17 Informasi foreign key relationship pada database its-dw



	srv	db	fk_rel	par_sch	par_tbl	par_col	ref_sch	ref_tbl	ref_col
25	WINDOWS-1K4UUD7	its-dw	FK_publikasi_jenis_kegiatan	kepegawaian	jenis_kegia...	id	kepegawaian	publikasi	jenis_penelitia
26	WINDOWS-1K4UUD7	its-dw	FK_publikasi_sumber_dana	kepegawaian	sumber_dana	id	kepegawaian	publikasi	sumber_dana
27	WINDOWS-1K4UUD7	its-dw	FK_riwayat_kepuasan_tendik_k...	kepegawaian	kepuasan	id	kepegawaian	riwayat_kepua...	kepuasan_id
28	WINDOWS-1K4UUD7	its-dw	FK_riwayat_kepuasan_tendik_pr...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_kepua...	tendik_id
29	WINDOWS-1K4UUD7	its-dw	FK_seminar_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	seminar	dosen_id
30	WINDOWS-1K4UUD7	its-dw	FK_seminar_idaturan_seminar	kepegawaian	idaturan_se...	id	kepegawaian	seminar	jenis_id
31	WINDOWS-1K4UUD7	its-dw	FK_penghargaan_tendik_profil_t...	kepegawaian	profil_tendik	id	kepegawaian	penghargaan_t...	tendik_id
32	WINDOWS-1K4UUD7	its-dw	FK_penghargaan_tendik_pengh...	kepegawaian	penghargaan	id	kepegawaian	penghargaan_t...	penghargaan_
33	WINDOWS-1K4UUD7	its-dw	FK_surat_keputusan_kategori_k...	kepegawaian	kategori_k...	id	kepegawaian	surat_keputusan	kategori_kegi
34	WINDOWS-1K4UUD7	its-dw	FK_surat_keputusan_aturan_ke...	kepegawaian	aturan_keg...	id	kepegawaian	surat_keputusan	aturan_kegiat
35	WINDOWS-1K4UUD7	its-dw	FK_surat_keputusan_tingkat	kepegawaian	tingkat	id	kepegawaian	surat_keputusan	tingkat_id
36	WINDOWS-1K4UUD7	its-dw	FK_riwayat_tugasbelajar_tendik...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_tugasb...	tendik_id
37	WINDOWS-1K4UUD7	its-dw	FK_riwayat_tugasbelajar_tendik...	kepegawaian	tugasbelajar	id	kepegawaian	riwayat_tugasb...	tugasbelajar_i
38	WINDOWS-1K4UUD7	its-dw	FK_pelatihan_tendik_profil_tendik	kepegawaian	profil_tendik	id	kepegawaian	pelatihan_tendik	tendik_id
39	WINDOWS-1K4UUD7	its-dw	FK_pelatihan_tendik_pelatihan1	kepegawaian	pelatihan	id	kepegawaian	pelatihan_tendik	pelatihan_id
40	WINDOWS-1K4UUD7	its-dw	FK_riwayat_kepuasan_pegawai...	kepegawaian	kepuasan	id	kepegawaian	riwayat_kepua...	kepuasan_id
41	WINDOWS-1K4UUD7	its-dw	FK_riwayat_kepuasan_pegawai...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_kepua...	dosen_id
42	WINDOWS-1K4UUD7	its-dw	FK_riwayat_jabatan_tendik_profi...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_jabata...	tendik_id
43	WINDOWS-1K4UUD7	its-dw	FK_riwayat_jabatan_tendik_jaba...	kepegawaian	jabatan	id	kepegawaian	riwayat_jabata...	jabatan_id
44	WINDOWS-1K4UUD7	its-dw	FK_penghargaan_pegawai_profi...	kepegawaian	profil_dosen	id	kepegawaian	penghargaan_...	dosen_id
45	WINDOWS-1K4UUD7	its-dw	FK_penghargaan_pegawai_pen...	kepegawaian	penghargaan	id	kepegawaian	penghargaan_...	penghargaan_
46	WINDOWS-1K4UUD7	its-dw	FK_riwayat_jabatan_dosen_profi...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_jabata...	dosen_id
47	WINDOWS-1K4UUD7	its-dw	FK_riwayat_jabatan_dosen_jaba...	kepegawaian	jabatan	id	kepegawaian	riwayat_jabata...	jabatan_id
48	WINDOWS-1K4UUD7	its-dw	FK_beban_kegiatan_kategori_k...	kepegawaian	kategori_k...	id	kepegawaian	beban_kegiatan	kategori_kegi
49	WINDOWS-1K4UUD7	its-dw	FK_riwayat_jpd_dosen_profil_do...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_jpd_do...	dosen_id

Gambar 6.18 Informasi foreign key relationship pada database its-dw



Results Messages

	srv	db	fk_rel	par_sch	par_tbl	par_col	ref_sch	ref_tbl	ref_col
50	WINDOWS-1K4UUD7	its-dw	FK_riwayat_absensi_tendik_profi...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_absens...	tendik_id
51	WINDOWS-1K4UUD7	its-dw	FK_riwayat_pendidikan_dosen_...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_pendidi...	dosen_id
52	WINDOWS-1K4UUD7	its-dw	FK_riwayat_pendidikan_dosen_j...	kepegawaian	jenjang_pe...	id	kepegawaian	riwayat_pendidi...	pendidikan_id
53	WINDOWS-1K4UUD7	its-dw	FK_tugasbelajar_jenjang_pendid...	kepegawaian	jenjang_pe...	id	kepegawaian	tugasbelajar	jenjang_pendi
54	WINDOWS-1K4UUD7	its-dw	FK_kepuasan_kepuasan_kategori	kepegawaian	kepuasan_...	id	kepegawaian	kepuasan	kepuasan_ka
55	WINDOWS-1K4UUD7	its-dw	FK_surat_keputusan_dosen_pro...	kepegawaian	profil_dosen	id	kepegawaian	surat_keputusa...	dosen_id
56	WINDOWS-1K4UUD7	its-dw	FK_surat_keputusan_dosen_sur...	kepegawaian	surat_kepu...	id	kepegawaian	surat_keputusa...	surat_keputus
57	WINDOWS-1K4UUD7	its-dw	FK_dati1_npsn_provinsi	akademik	provinsi	id	mapping	dati1_provinsi	provinsi_id
58	WINDOWS-1K4UUD7	its-dw	FK_riwayat_tugasbelajar_pegaw...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_tugasb...	dosen_id
59	WINDOWS-1K4UUD7	its-dw	FK_riwayat_tugasbelajar_pegaw...	kepegawaian	tugasbelajar	id	kepegawaian	riwayat_tugasb...	tugasbelajar_i
60	WINDOWS-1K4UUD7	its-dw	FK_provinsi_baru_provinsi_provi...	akademik	provinsi	id	mapping	provinsi_baru_...	provinsi_id
61	WINDOWS-1K4UUD7	its-dw	FK_user_prodi_prodi	akademik	prodi	id	mapping	user_prodi	prodi_id
62	WINDOWS-1K4UUD7	its-dw	FK_riwayat_ppk_dosen_profil_d...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_ppk_d...	dosen_id
63	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_history_mahasis...	akademik	mahasiswa	id	akademik	mahasiswa_his...	mahasiswa_id
64	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_history_semester	akademik	semester	id	akademik	mahasiswa_his...	semester_id
65	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_history_mahasis...	akademik	mahasiswa...	id	akademik	mahasiswa_his...	mahasiswa_st
66	WINDOWS-1K4UUD7	its-dw	FK_riwayat_ppk_tendik_profil_te...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_ppk_te...	tendik_id
67	WINDOWS-1K4UUD7	its-dw	FK_riwayat_pendidikan_tendik_...	kepegawaian	profil_tendik	id	kepegawaian	riwayat_pendidi...	tendik_id
68	WINDOWS-1K4UUD7	its-dw	FK_riwayat_pendidikan_tendik_j...	kepegawaian	jenjang_pe...	id	kepegawaian	riwayat_pendidi...	pendidikan_id
69	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_recent_mahasis...	akademik	mahasiswa	id	akademik	mahasiswa_rec...	mahasiswa_id
70	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_recent_semester	akademik	semester	id	akademik	mahasiswa_rec...	semester_id
71	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_recent_mahasis...	akademik	mahasiswa...	id	akademik	mahasiswa_rec...	mahasiswa_st
72	WINDOWS-1K4UUD7	its-dw	FK_mahasiswa_recent_predikat...	akademik	predikat_lul...	id	akademik	mahasiswa_rec...	predikat_lulus
73	WINDOWS-1K4UUD7	its-dw	FK_beban_kerja_dosen_profil_d...	kepegawaian	profil_dosen	id	kepegawaian	beban_kerja_d...	dosen_id
74	WINDOWS-1K4UUD7	its-dw	FK_beban_kerja_dosen_beban_...	kepegawaian	beban_keg...	id	kepegawaian	beban_kerja_d...	beban_kegiat

Gambar 6.19 Informasi foreign key relationship pada database its-dw

75	WINDOWS-1K4UUD7	its-dw	FK_pelatihan_pegawai_profil_do...	kepegawaian	profil_dosen	id	kepegawaian	pelatihan_dosen	dosen_id
76	WINDOWS-1K4UUD7	its-dw	FK_pelatihan_pegawai_pelatihan	kepegawaian	pelatihan	id	kepegawaian	pelatihan_dosen	pelatihan_id
77	WINDOWS-1K4UUD7	its-dw	FK_pelatihan_pelatihan_tingkat	kepegawaian	pelatihan_ti...	id	kepegawaian	pelatihan	pelatihan_ting
78	WINDOWS-1K4UUD7	its-dw	FK_aturan_kegiatan_kategori_k...	kepegawaian	kategori_k...	id	kepegawaian	aturan_kegiatan	kategori_kegi
79	WINDOWS-1K4UUD7	its-dw	FK_jurnal_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	jurnal	dosen_id
80	WINDOWS-1K4UUD7	its-dw	FK_jurnal_idaturan_jurnal	kepegawaian	idaturan_jur...	id	kepegawaian	jurnal	jenis_id
81	WINDOWS-1K4UUD7	its-dw	FK_paten_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	paten	dosen_id
82	WINDOWS-1K4UUD7	its-dw	FK_paten_idaturan_paten	kepegawaian	idaturan_p...	id	kepegawaian	paten	jenis_id
83	WINDOWS-1K4UUD7	its-dw	FK_publikasi_dosen_publikasi	kepegawaian	publikasi	id	kepegawaian	publikasi_dosen	publikasi_id
84	WINDOWS-1K4UUD7	its-dw	FK_penelitian_dosen_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	publikasi_dosen	dosen_id
85	WINDOWS-1K4UUD7	its-dw	FK_penelitian_dosen_jenispenelti	kepegawaian	jenispeneliti	id	kepegawaian	publikasi_dosen	jenispeneliti_id
86	WINDOWS-1K4UUD7	its-dw	FK_penelitian_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	penelitian	dosen_id
87	WINDOWS-1K4UUD7	its-dw	FK_riwayat_absensi_dosen_profi...	kepegawaian	profil_dosen	id	kepegawaian	riwayat_absens...	dosen_id
88	WINDOWS-1K4UUD7	its-dw	FK_kota_lahir_dosen_profil_dosen	kepegawaian	profil_dosen	id	kepegawaian	kota_lahir_dosen	dosen_id
89	WINDOWS-1K4UUD7	its-dw	FK_prodi_jurusan	akademik	jurusan	id	akademik	prodi	jurusan_id

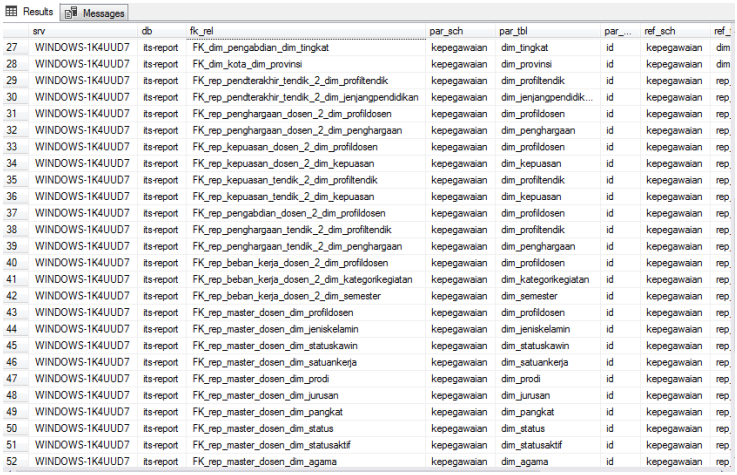
Query executed successfully. olap.its.ac.id (13.0 SP1) | monitoring (76) | its-dw | 00:00:01 | 89 rows

Gambar 6.20 Informasi foreign key relationship pada database its-dw

Selanjutnya pengujian dilakukan pada database its-report. Hasil pengujian pada database its-report ditampilkan pada gambar 6.11 sampai 6.13.

	srv	db	fk_rel	par_sch	par_tbl	par_...	ref_sch	ref_...
1	WINDOWS-1K4UUD7	its-report	FK_rep_jabatan_fungsional_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep_
2	WINDOWS-1K4UUD7	its-report	FK_rep_jabatan_fungsional_dosen_2_dim_jabatan	kepegawaian	dim_jabatan	id	kepegawaian	rep_
3	WINDOWS-1K4UUD7	its-report	FK_rep_ppk_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep_
4	WINDOWS-1K4UUD7	its-report	FK_rep_pelatihan_dosen_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep_
5	WINDOWS-1K4UUD7	its-report	FK_rep_pelatihan_dosen_dim_pelatihan	kepegawaian	dim_pelatihan	id	kepegawaian	rep_
6	WINDOWS-1K4UUD7	its-report	FK_rep_persebaran_provlahir_dosen_2_dim_profildo...	kepegawaian	dim_profildosen	id	kepegawaian	rep_
7	WINDOWS-1K4UUD7	its-report	FK_rep_persebaran_provlahir_dosen_2_dim_kota	kepegawaian	dim_kota	id	kepegawaian	rep_
8	WINDOWS-1K4UUD7	its-report	FK_dim_pelatihan_dim_pelatihan_tingkat	kepegawaian	dim_pelatihan_tingkat	id	kepegawaian	dim
9	WINDOWS-1K4UUD7	its-report	FK_rep_jabatan_fungsional_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep_
10	WINDOWS-1K4UUD7	its-report	FK_rep_jabatan_fungsional_tendik_2_dim_jabatan	kepegawaian	dim_jabatan	id	kepegawaian	rep_
11	WINDOWS-1K4UUD7	its-report	FK_dim_penelitian_dim_jeniskegiatan	kepegawaian	dim_jeniskegiatan	id	kepegawaian	dim
12	WINDOWS-1K4UUD7	its-report	FK_dim_penelitian_dim_sumberdana	kepegawaian	dim_sumberdana	id	kepegawaian	dim
13	WINDOWS-1K4UUD7	its-report	FK_dim_kepuasan_dim_kepuasankategori	kepegawaian	dim_kepuasankateg...	id	kepegawaian	dim
14	WINDOWS-1K4UUD7	its-report	FK_rep_ppk_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep_
15	WINDOWS-1K4UUD7	its-report	FK_rep_penelitian_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep_
16	WINDOWS-1K4UUD7	its-report	FK_rep_penelitian_dosen_2_dim_penelitian	kepegawaian	dim_penelitian	id	kepegawaian	rep_
17	WINDOWS-1K4UUD7	its-report	FK_rep_penelitian_dosen_2_dim_jenispeneliti	kepegawaian	dim_jenispeneliti	id	kepegawaian	rep_
18	WINDOWS-1K4UUD7	its-report	FK_rep_persebaran_provlahir_tendik_2_dim_profilten...	kepegawaian	dim_profiltendik	id	kepegawaian	rep_
19	WINDOWS-1K4UUD7	its-report	FK_rep_persebaran_provlahir_tendik_2_dim_kota	kepegawaian	dim_kota	id	kepegawaian	rep_
20	WINDOWS-1K4UUD7	its-report	FK_rep_pelatihan_tendik_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep_
21	WINDOWS-1K4UUD7	its-report	FK_rep_pelatihan_tendik_dim_pelatihan	kepegawaian	dim_pelatihan	id	kepegawaian	rep_
22	WINDOWS-1K4UUD7	its-report	FK_rep_tugasbelajar_dosen_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep_
23	WINDOWS-1K4UUD7	its-report	FK_rep_tugasbelajar_dosen_dim_tugasbelajar	kepegawaian	dim_tugasbelajar	id	kepegawaian	rep_
24	WINDOWS-1K4UUD7	its-report	FK_rep_tugasbelajar_tendik_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep_
25	WINDOWS-1K4UUD7	its-report	FK_rep_tugasbelajar_tendik_dim_tugasbelajar	kepegawaian	dim_tugasbelajar	id	kepegawaian	rep_
26	WINDOWS-1K4UUD7	its-report	FK_dim_pengabdian_dim_semester	kepegawaian	dim_semester	id	kepegawaian	dim

Gambar 6.21 Informasi foreign key relationship database its-report



Results | Messages

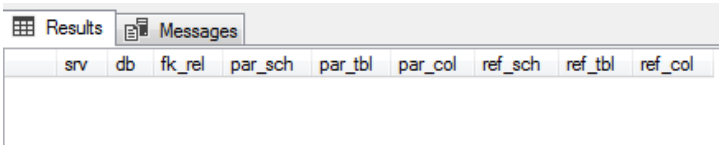
	srv	db	fk_rel	par_sch	par_tbl	par_...	ref_sch	ref_...
27	WINDOWS-1K4UUD7	its-report	FK_dim_pengabdian_dim_tingkat	kepegawaian	dim_tingkat	id	kepegawaian	dim
28	WINDOWS-1K4UUD7	its-report	FK_dim_kota_dim_provinsi	kepegawaian	dim_provinsi	id	kepegawaian	dim
29	WINDOWS-1K4UUD7	its-report	FK_rep_pendterakhir_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep
30	WINDOWS-1K4UUD7	its-report	FK_rep_pendterakhir_tendik_2_dim_jenjangpendidikan	kepegawaian	dim_jenjangpendidik...	id	kepegawaian	rep
31	WINDOWS-1K4UUD7	its-report	FK_rep_penghargaan_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
32	WINDOWS-1K4UUD7	its-report	FK_rep_penghargaan_dosen_2_dim_penghargaan	kepegawaian	dim_penghargaan	id	kepegawaian	rep
33	WINDOWS-1K4UUD7	its-report	FK_rep_kepuasan_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
34	WINDOWS-1K4UUD7	its-report	FK_rep_kepuasan_dosen_2_dim_kepuasan	kepegawaian	dim_kepuasan	id	kepegawaian	rep
35	WINDOWS-1K4UUD7	its-report	FK_rep_kepuasan_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep
36	WINDOWS-1K4UUD7	its-report	FK_rep_kepuasan_tendik_2_dim_kepuasan	kepegawaian	dim_kepuasan	id	kepegawaian	rep
37	WINDOWS-1K4UUD7	its-report	FK_rep_pengabdian_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
38	WINDOWS-1K4UUD7	its-report	FK_rep_penghargaan_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep
39	WINDOWS-1K4UUD7	its-report	FK_rep_penghargaan_tendik_2_dim_penghargaan	kepegawaian	dim_penghargaan	id	kepegawaian	rep
40	WINDOWS-1K4UUD7	its-report	FK_rep_beban_kerja_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
41	WINDOWS-1K4UUD7	its-report	FK_rep_beban_kerja_dosen_2_dim_kategorikegiatan	kepegawaian	dim_kategorikegiatan	id	kepegawaian	rep
42	WINDOWS-1K4UUD7	its-report	FK_rep_beban_kerja_dosen_2_dim_semester	kepegawaian	dim_semester	id	kepegawaian	rep
43	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
44	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_jeniskelamin	kepegawaian	dim_jeniskelamin	id	kepegawaian	rep
45	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_statuskawin	kepegawaian	dim_statuskawin	id	kepegawaian	rep
46	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_satuankerja	kepegawaian	dim_satuankerja	id	kepegawaian	rep
47	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_prodi	kepegawaian	dim_prodi	id	kepegawaian	rep
48	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_jurusan	kepegawaian	dim_jurusan	id	kepegawaian	rep
49	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_pangkat	kepegawaian	dim_pangkat	id	kepegawaian	rep
50	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_status	kepegawaian	dim_status	id	kepegawaian	rep
51	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_statusaktif	kepegawaian	dim_statusaktif	id	kepegawaian	rep
52	WINDOWS-1K4UUD7	its-report	FK_rep_master_dosen_dim_agama	kepegawaian	dim_agama	id	kepegawaian	rep

Gambar 6.22 Informasi foreign key relationship database its-report

53	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep
54	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_statuskawin	kepegawaian	dim_statuskawin	id	kepegawaian	rep
55	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_jeniskelamin	kepegawaian	dim_jeniskelamin	id	kepegawaian	rep
56	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_satuankerja	kepegawaian	dim_satuankerja	id	kepegawaian	rep
57	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_pangkat	kepegawaian	dim_pangkat	id	kepegawaian	rep
58	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_status	kepegawaian	dim_status	id	kepegawaian	rep
59	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_statusaktif	kepegawaian	dim_statusaktif	id	kepegawaian	rep
60	WINDOWS-1K4UUD7	its-report	FK_rep_master_tendik_dim_agama1	kepegawaian	dim_agama	id	kepegawaian	rep
61	WINDOWS-1K4UUD7	its-report	FK_dim_jurusan_dim_fakultas	kepegawaian	dim_fakultas	id	kepegawaian	dim
62	WINDOWS-1K4UUD7	its-report	FK_rep_ipd_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
63	WINDOWS-1K4UUD7	its-report	FK_rep_ipd_dosen_2_dim_semester	kepegawaian	dim_semester	id	kepegawaian	rep
64	WINDOWS-1K4UUD7	its-report	FK_rep_absensi_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
65	WINDOWS-1K4UUD7	its-report	FK_rep_absensi_tendik_2_dim_profiltendik	kepegawaian	dim_profiltendik	id	kepegawaian	rep
66	WINDOWS-1K4UUD7	its-report	FK_rep_pendterakhir_dosen_2_dim_profildosen	kepegawaian	dim_profildosen	id	kepegawaian	rep
67	WINDOWS-1K4UUD7	its-report	FK_rep_pendterakhir_dosen_2_dim_jenjangpendidikan	kepegawaian	dim_jenjangpendidik...	id	kepegawaian	rep

Gambar 6.23 Informasi foreign key relationship database its-report

Selanjutnya hasil pengujian pada database resits ditampilkan pada gambar 6.14.

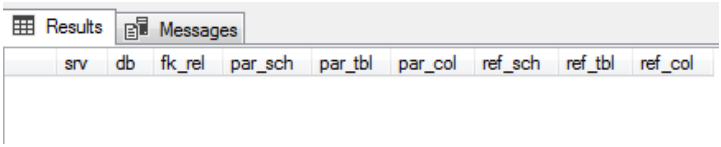


Results | Messages

srv	db	fk_rel	par_sch	par_tbl	par_col	ref_sch	ref_tbl	ref_col

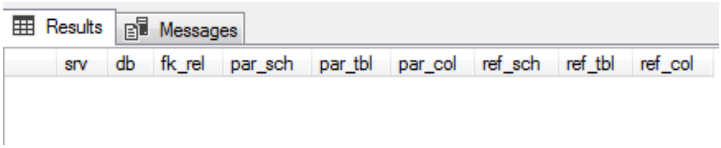
Gambar 6.24 Informasi foreign key relationship database resits

Selanjutnya hasil pengujian pada database spmi_view ditampilkan pada gambar 6.15.



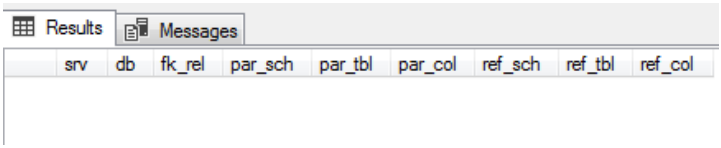
Gambar 6.25 Informasi foreign key relationship database spmi_view

Selanjutnya hasil pengujian pada database TEST ditampilkan pada gambar 6.16.



Gambar 6.26 Informasi foreign key relationship database TEST

Selanjutnya hasil pengujian pada database forlap ditampilkan pada gambar 6.17.



Gambar 6.27 Informasi foreign key relationship database forlap

Jika pengujian melalui SQL Server Management Studio berhasil, maka tinggal memeriksa apakah koneksi PDO dan iterasi hasil *fetch* yang dihasilkan dari eksekusi database menghasilkan hasil yang dimaksud.

6.1.8. Aplikasi dapat menjalankan kebutuhan No. 5,6,7,8 pada semua database yang didapat dari kebutuhan No. 3.

Dari pengujian poin kebutuhan fungsional No.5, 6, 7, 8 di atas dapat disimpulkan pengujian pada masing-masing database berhasil mendapatkan data yang dibutuhkan. Karenanya pada pengujian tahap ini aplikasi dinyatakan sukses.

6.1.9. Aplikasi dapat mengambil query DMV yang telah disimpan pada database MariaDB

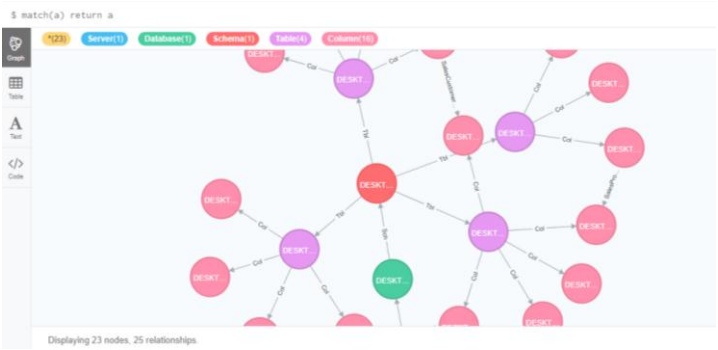
Pengujian ini dilakukan dengan

6.1.10. Aplikasi dapat mengambil informasi schema, table, column, yang memiliki relasi JOIN dari hasil parsing query DMV yang didapat dari kebutuhan No. 10.

Sebagaimana pada kode yang tertera di bab V, aplikasi mengambil data query dari MariaDB yang kemudian diparsing dan langsung disimpan per elemen informasi ke dalam graph Neo4j.

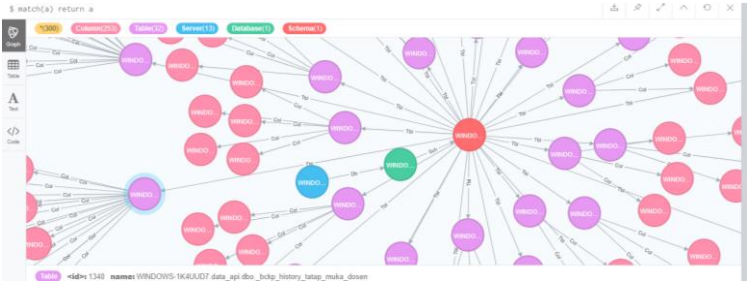
6.1.11. Aplikasi dapat menyimpan data server, database, schema, table, column, foreign key realtionship, dan JOIN relationship ke dalam graph database Neo4j

Pengujian dilakukan dengan menjalankan **dmv.bat**, **syscat.bat**, dan **parse.bat** pada database RESITS dan pada database Sales. Setelah masing-masing file dijalankan, buka browser di alamat http://localhost:7474. Lalu ketikkan `match(a) return a` pada input query cypher. Jika graph berhasil ditampilkan, maka aplikasi terhitung berhasil menyimpan data ke dalam Neo4j. Hasil pengujian untuk database Sales ditunjukkan pada gambar 6..



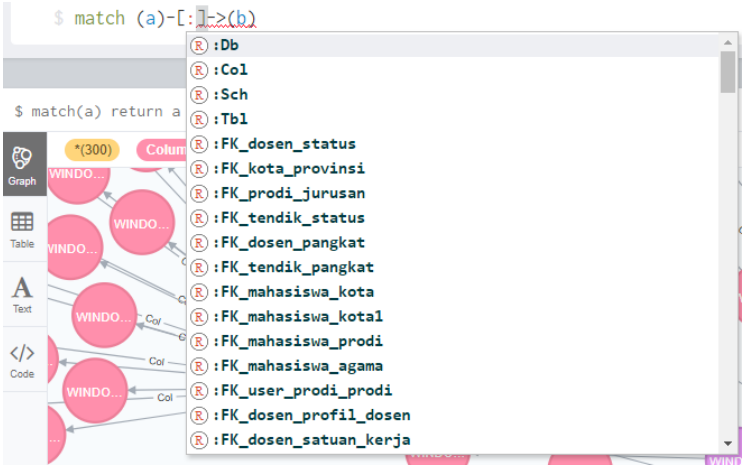
Gambar 6.28 Graph database hasil visualisasi database Sales

Sedangkan hasil pengujian database RESITS ditunjukkan pada gambar 6..



Gambar 6.29 Graph database hasil visualisasi database RESITS

Pada gambar 6.2. di atas nampak bahwa jumlah node yang ditampilkan hanya berjumlah 300 node yang merupakan limitasi Neo4j browser, sehingga belum nampak relasi *foreign key* pada graph visualisasi database RESITS. Karena itu perlu dilakukan penelusuran lebih lanjut untuk mengungkap apakah relasi foreign key berhasil disimpan di dalam graph database. Untuk melakukannya masukkan cypher sebagai berikut: match (a)-[:]->(b), lalu arahkan pointer (*hover*) pada bagain titik dua cypher tersebut, maka akan muncul *suggestion* relationship apa saja yang telah tersimpan di dalam graph database seperti pada gambar 6.. berikut.

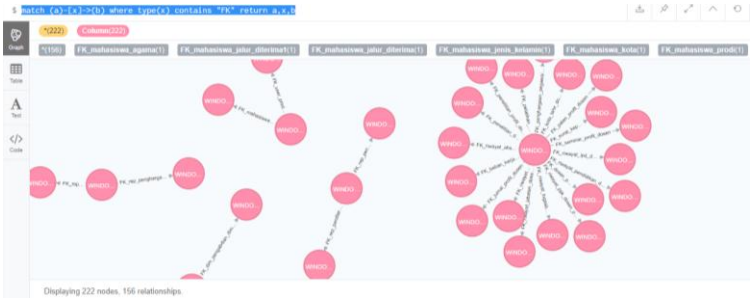


Gambar 6.30 Relationship yang telah tersimpan di dalam graph Neo4j

Untuk lebih meyakinkan bahwa relationship foreign key telah berhasil disimpan ke dalam Neo4j gunakan cypher berikut:

```
match (a)-[x]->(b) where type(x) contains "FK" return a,x,b
```

Query di atas berfungsi untuk mengambil data 2 node yang direpresentasikan dengan **a** dan **b** yang di antara keduanya terdapat *relationship* yang direpresentasikan dengan **x**. Hasil yang diambil haruslah merupakan yang relationship-nya mengandung string “FK”. Karena pada bagian ini yang diuji adalah keberadaan foreign key relationship. Return berfungsi untuk menampilkan seluruh node dan relationship yang telah didefinisikan pada **match**. Sehingga hasilnya seperti yang ditampilkan gambar 6.7.



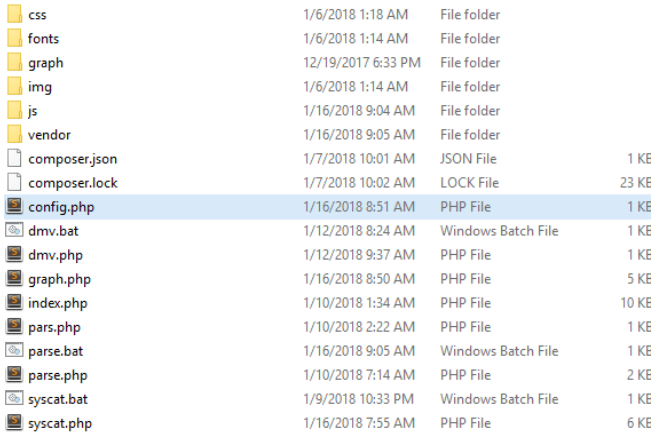
Gambar 6.31 Node yang dihubungkan relationship berbasis foreign key

Dari kedua hasil di atas, terbukti aplikasi telah berhasil menyimpan hasil visualisasi hirarki database SQL Server ke dalam bentuk graph Neo4j.

Untuk memastikan aplikasi benar-benar berhasil menyimpan seluruh data yang didapat dari query ke database SQL Server, perlu dilakukan pengujian tambahan yaitu dengan membandingkan data yang didapat pada SQL Server dengan data yang tersimpan pada graph database Neo4j.

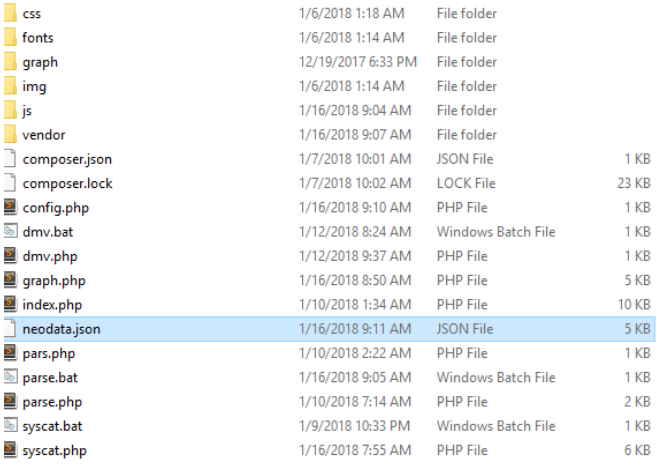
6.1.12. Aplikasi dapat menghasilkan array sesuai standar data output library Neo4jD3.js dan menyimpannya ke dalam sebuah file JSON

Pengujian tahap ini dilakukan dengan mengamati perbedaan direktori lokasi aplikasi diletakkan pada sebelum dan setelah menjalankan **syscat.bat, dmv.bat,** dan **parse.bat**. Pada kode aplikasi, nampak bahwa array disimpan ke dalam file JSON pada direktori root bernama **neodata.json**. Berikut adalah file direktori root sebelum 3 script tersebut dijalankan.



Gambar 6.32 Direktori aplikasi

Pengujian pertama dilakukan pada database Sales. Setelah 3 script dijalankan dengan memakan waktu yang cukup singkat, nampak ada file baru yaitu neodata.json yang memiliki ukuran 5 KB seperti yang ditampilkan pada gambar 6.11.



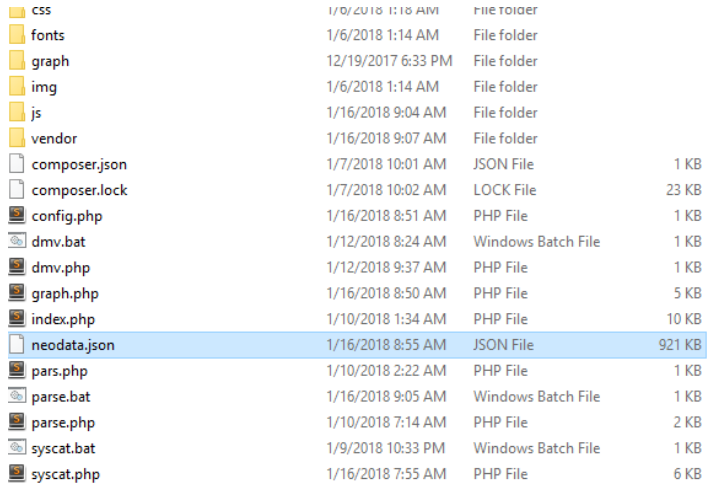
Gambar 6.33 Direktori aplikasi setelah pengujian database Sales

Untuk membuktikan bahwa file **neodata.json** benar mengandung informasi node dan relationship, buka file

neodata.json dengan notepad atau text editor lainnya. Atau bisa juga dengan layanan JSON formatter yang tersedia gratis seperti http://freeformatter.com. Hasilnya seperti yang ditampilkan pada gambar 6.1x.

Gambar 6.34 Isi file JSON setelah dilakukan *formatting*

Pengujian kedua dilakukan pada database RESITS. Setelah dijalankan 3 script tersebut untuk database RESITS, akan nampak perbedaan yaitu file neodata.json berubah ukuran menjadi lebih besar yaitu 921 KB seperti yang ditampilkan pada gambar 6.12.



css	1/6/2018 1:18 AM	File folder	
fonts	1/6/2018 1:14 AM	File folder	
graph	12/19/2017 6:33 PM	File folder	
img	1/6/2018 1:14 AM	File folder	
js	1/16/2018 9:04 AM	File folder	
vendor	1/16/2018 9:07 AM	File folder	
composer.json	1/7/2018 10:01 AM	JSON File	1 KB
composer.lock	1/7/2018 10:02 AM	LOCK File	23 KB
config.php	1/16/2018 8:51 AM	PHP File	1 KB
dmv.bat	1/12/2018 8:24 AM	Windows Batch File	1 KB
dmv.php	1/12/2018 9:37 AM	PHP File	1 KB
graph.php	1/16/2018 8:50 AM	PHP File	5 KB
index.php	1/10/2018 1:34 AM	PHP File	10 KB
neodata.json	1/16/2018 8:55 AM	JSON File	921 KB
pars.php	1/10/2018 2:22 AM	PHP File	1 KB
parse.bat	1/16/2018 9:05 AM	Windows Batch File	1 KB
parse.php	1/10/2018 7:14 AM	PHP File	2 KB
syscat.bat	1/9/2018 10:33 PM	Windows Batch File	1 KB
syscat.php	1/16/2018 7:55 AM	PHP File	6 KB

Gambar 6.35 Direktori aplikasi setelah pengujian database RESITS

Apabila dibuka dengan text editor dan dilakukan formatting maka hasilnya seperti yang ditampilkan pada gambar.

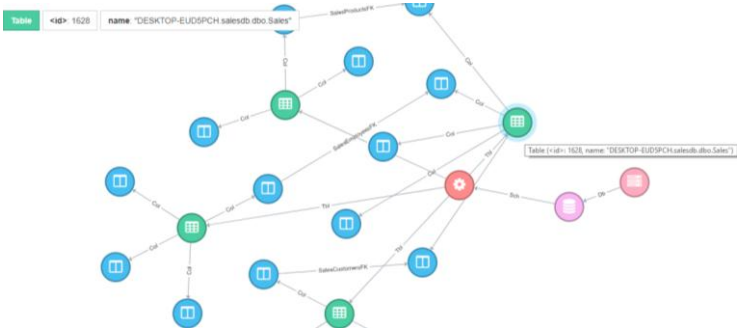

```
{"results":[{"data":[{"graph":{"nodes":[{"id":113,"labels":["Server"],"properties":{"name":
"AKADEMIK"}},{"id":114,"labels":["Server"],"properties":{"name":"AKADEMIK-1"}},{"id":115,"labels":[
"Server"],"properties":{"name":"AKADEMIK-2"}},{"id":116,"labels":["Server"],"properties":{"name":
"AKADEMIK-3"}},{"id":117,"labels":["Server"],"properties":{"name":"INTEGRA"}},{"id":118,"labels":[
"Server"],"properties":{"name":"REPORTING_AKADEMIK"}},{"id":119,"labels":["Server"],"properties":{
"name":"SILACAK"}},{"id":120,"labels":["Server"],"properties":{"name":"SIMPEG"}},{"id":127,"labels"
:["Server"],"properties":{"name":"SIMPEL"}},{"id":128,"labels":["Server"],"properties":{"name":
"SIPMABA"}},{"id":129,"labels":["Server"],"properties":{"name":"SKEM"}},{"id":130,"labels":[
"Server"],"properties":{"name":"SPMI"}},{"id":131,"labels":["Server"],"properties":{"name":
"WINDOWS-1K4UUD7"}},{"id":132,"labels":["Database"],"properties":{"name":"WINDOWS-1K4UUD7.data_api"
}},{"id":133,"labels":["Schema"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo"}},{"id":134,
"labels":["Table"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.jumlahtendikbelajar"}},{"id":
135,"labels":["Column"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.jumlahtendikbelajar.id"
}},{"id":136,"labels":["Column"],"properties":{"name":
"WINDOWS-1K4UUD7.data_api.dbo.jumlahtendikbelajar.jumlahtendikbelajar"}},{"id":137,"labels":[
"Column"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.jumlahtendikbelajar.satker"}},{"id":
138,"labels":["Table"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar"}},{
"id":139,"labels":["Column"],"properties":{"name":
"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.kontinuitas"}},{"id":140,"labels":["Column"],
"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.nama_dept"}},{"id":141,
"labels":["Column"],"properties":{"name":
"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.nama_dosen"}},{"id":142,"labels":["Column"],
"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.nama_mk"}},{"id":143,
"labels":["Column"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.nip"
}},{"id":144,"labels":["Column"],"properties":{"name":
"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.sks_kuliah"}},{"id":145,"labels":["Column"],
"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.smt"}},{"id":146,"labels":[
"Column"],"properties":{"name":"WINDOWS-1K4UUD7.data_api.dbo.history_sks_mengajar.thn_ajaran"}},{
"id":147,"labels":["Column"],"properties":{"name":
```


Gambar 6.36 Isi file JSON dibuka lewat text editor

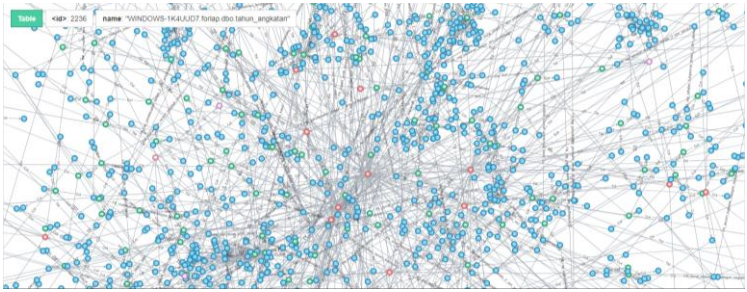
Dari hasil pengujian di atas terbukti bahwa aplikasi berhasil menghasilkan file JSON yang menyimpan informasi nodes dan relationship dari graph Neo4j.

6.1.13. Aplikasi dapat mengolah file JSON yang telah dibuat pada kebutuhan No. 13 menjadi bentuk graph diagram lengkap dengan seluruh informasi node dan relationship.

Pengujian pada tahap terakhir ini dilakukan dengan cara menjalankan graph.php melalui browser. Apabila pada browser berhasil ditampilkan *graph diagram* seperti di bawah ini, maka aplikasi terbukti berhasil mengolah file JSON menjadi diagram. Hasil pengolahan database RESITS ditampilkan pada gambar 6.10. Sedangkan hasil pengolahan database Sales ditampilkan pada gambar 6.11.



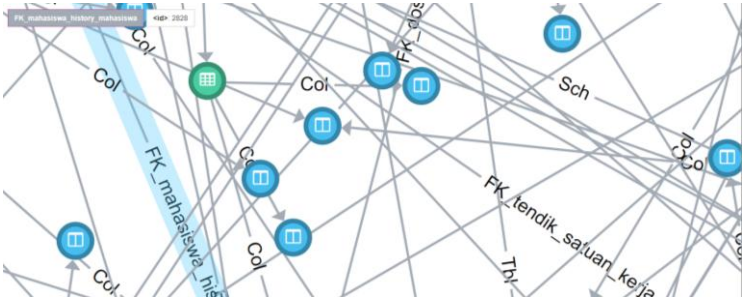
Gambar 6.37 Hasil pengolahan JSON menjadi graph diagram untuk database Sales



Gambar 6.38 Hasil pengolahan JSON menjadi graph diagram untuk database RESITS sebelium dilakukan *zooming*

Nampak pada hasil pengujian di atas, diagram yang dihasilkan dari database RESITS relatif besar dan susah untuk diamati. Selain itu diagram juga berat untuk ditampilkan pada *client-*

side. Pada pengujian ini apabila diagram dizoom dengan menggunakan scroll pada mouse maka hasilnya akan nampak lebih jelas database, schema, table, column, dan beberapa foreign key relationship seperti pada gambar 6.12.



Gambar 6.39 Hasil pengolahan JSON menjadi graph diagram untuk database RESITS dengan *zooming*

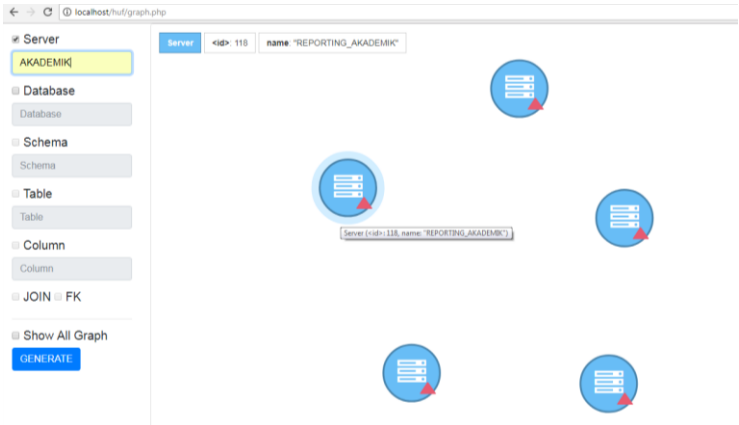
Dan demikianlah pengujian aplikasi dari sisi kebutuhan fungsional. Dari 14 tahapan pengujian tersebut dapat disimpulkan bahwa aplikasi berjalan sesuai dengan yang diharapkan karena telah memenuhi seluruh kebutuhan fungsional yang ditetapkan pada rancangan aplikasi.

6.1.14. Pengujian Fungsi Filtering

Fungsi filtering pada aplikasi diuji dengan menyajikan beberapa skenario pengujian antara lain:

A. Filtering Server

Pengujian diawali dengan mencentang *checkbox* Server. *Checkbox* yang tercentang berhasil membuat input teks Server aktif dan bisa diisi. Pengujian dilanjutkan dengan memasukkan keyword yang berhubungan dengan nama Server pada RESITS semisal AKADEMIK pada input teks Server. Lalu klik tombol GENERATE. Aplikasi berhasil menampilkan informasi tentang Server AKADEMIK, AKADEMIK-1, AKADEMIK-2, AKADEMIK-3, REPORTING_AKADEMIK sesuai kriteria *keyword* yang dimasukkan.

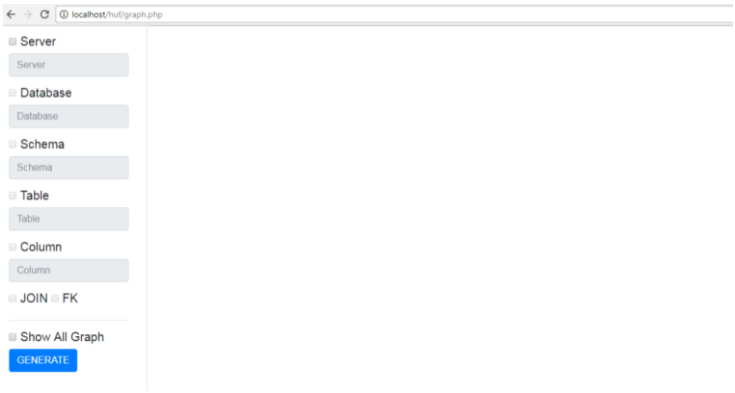


Gambar 6.40 Hasil filtering Server dengan keyword AKADEMIK

Apabila dicoba menggunakan *keyword* lain misal WINDOWS, aplikasi juga berhasil menampilkan informasi server WINDOWS-1K4UUD7 sesuai kriteria *keyword* yang dimasukkan.

Gambar 6.41 Hasil filtering Server dengan keyword WINDOWS

Pengujian dilanjutkan untuk memeriksa kondisi dan syarat input teks apakah *case-sensitive* atau *non case-sensitive.* Input teks server diuji dengan dimasukkan keyword ‘akademik’ menggunakan huruf non-kapital semua. Yang terjadi seperti nampak pada gambar yaitu tidak ada graph yang dihasilkan.

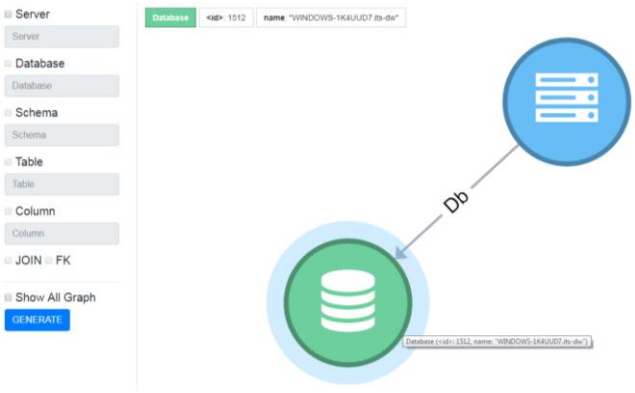


Gambar 6.42 Hasil filtering Server dengan keyword akademik

Dari hasil pengujian di atas dapat disimpulkan aplikasi berhasil melakukan *filtering* pada graph yang ingin ditampilkan dengan batasan bahwa *keyword* yang digunakan untuk *filtering* haruslah tepat sesuai besar kecilnya huruf.

B. Filtering Server dan Database

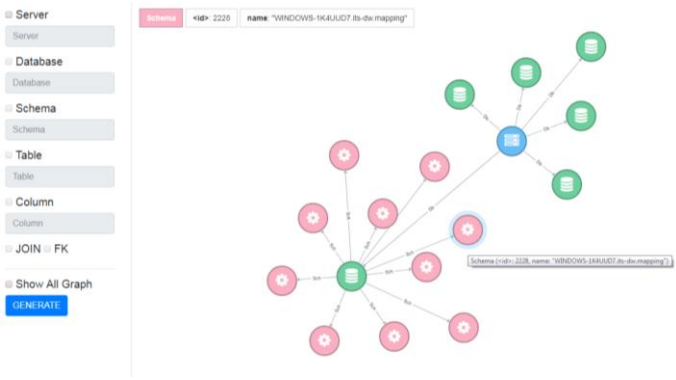
Pengujian diawali seperti sebelumnya yaitu dengan mencentang *checkbox* Server dan mengisi input teks Server yang telah aktif dengan keyword misal WINDOWS. Selanjutnya centang juga *checkbox* Database dan isikan input teks Database yang telah aktif dengan keyword misal its-dw. Karena sudah diketahui bahwa input keyword bersifat *case-sensitive* maka pengujian pada faktor *case-sensitive* tidak lagi dilakukan. Hasil pengujian sebagaimana pada gambar di bawah ini menunjukkan bahwa aplikasi berhasil menampilkan informasi Server WINDOWS-1K4UUD7 dan Database its-dw sesuai kriteria *keyword* yang dimasukkan.



Gambar 6.43 Hasil pengujian filtering Server dan Database

C. Filtering Server, Database, dan Schema

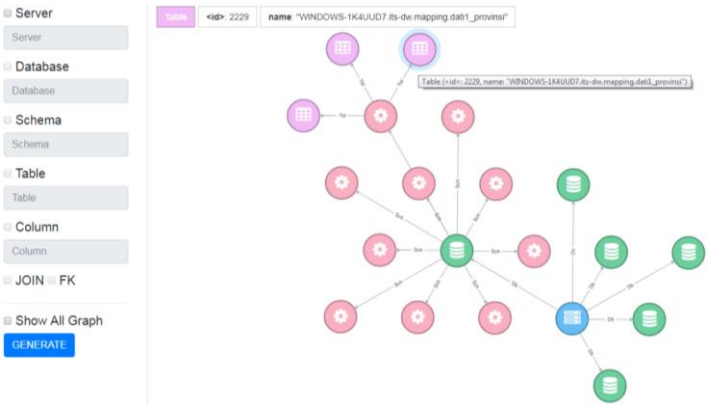
Aktivitas pengujian sama seperti pada pengujian B dengan ditambahkan aktivitas mencentang *checkbox* Schema. Dari hasil pengujian bagian ini sebagaimana ditunjukkan pada gambar 6.44 di bawah ini dapat disimpulkan bahwa aplikasi berhasil mem-*filter* informasi graph untuk Server, Database, dan Schema sesuai dengan kriteria *keyword* yang dimasukkan.



Gambar 6.44 Hasil pengujian filtering Server, Database, dan Schema

D. Filtering Server, Database, Schema dan Table

Aktivitas pengujian sama seperti pada pengujian C dengan ditambahkan aktivitas mengisi input *text* Schema dan mencentang *checkbox* Table.

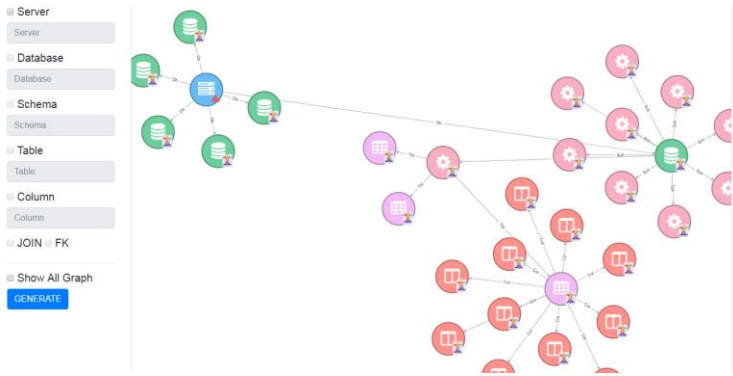


Gambar 6.45 Hasil filtering Server, Database, Schema dan Table

Gambar 6.45 di atas menunjukkan bahwa aplikasi berhasil melakukan *filtering* informasi Server, Database, Schema, dan Table tertentu.

E. Filtering Server, Database, Schema, Table, dan Column

Pengujian yang dilakukan sama seperti pengujian D dengan beberapa tambahan yaitu dengan mencentang input *checkbox* Table dan mengisi pada input *text* Table. Kemudian klik *checkbox* Column. Lalu klik tombol GENERATE. Aplikasi berhasil menampilkan informasi tentang Table dan Column tertentu seperti pada gambar 6.46.

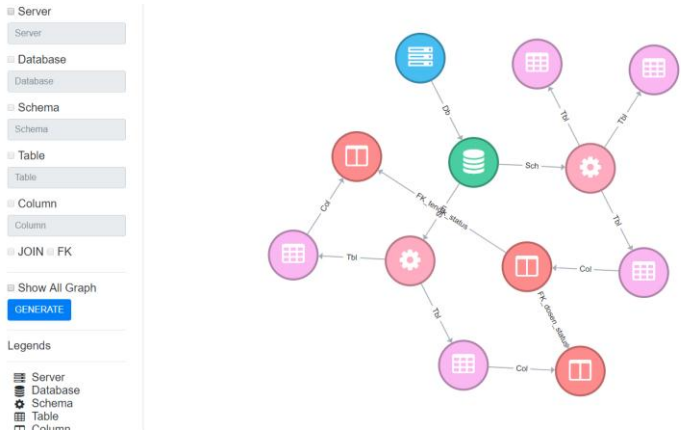


Gambar 6.46 Hasil filtering Server, Database, Schema, Table, Column

Dari hasil pengujian di atas dapat disimpulkan bahwa aplikasi berhasil melakukan *filtering* untuk Server, Database, Schema, Table, dan Column tertentu.

F. Filtering Filtering Server, Database, Schema, Table, Column, dan FK relationship

Pengujian yang dilakukan sama seperti pengujian D dengan beberapa tambahan yaitu dengan mencentang input *checkbox* Column dan mengisi pada input *text* Column. Kemudian klik *checkbox* FK. Lalu klik tombol GENERATE. Aplikasi berhasil menampilkan informasi FK relationship seperti pada gambar 6.47 di bawah ini.

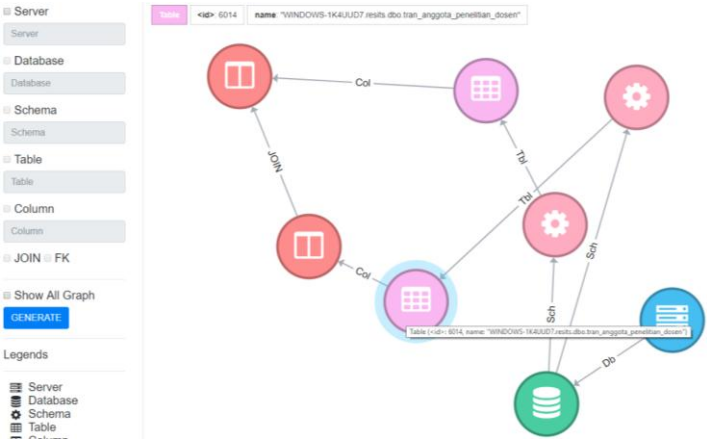


Gambar 6.47 Hasil filtering Server, Database, Schema, Table, Column, dan FK relationship

Dari hasil pengujian di atas dapat disimpulkan bahwa aplikasi berhasil melakukan *filtering* untuk Server, Database, Schema, Table, Column, dan foreign key relationship tertentu.

G. Filtering Filtering Server, Database, Schema, Table, Column, dan JOIN relationship

Pengujian yang dilakukan sama seperti pengujian D dengan beberapa tambahan yaitu dengan mencentang input *checkbox* Column dan mengisi pada input *text* Column. Kemudian klik *checkbox* JOIN. Lalu klik tombol GENERATE. Aplikasi berhasil menampilkan informasi FK relationship seperti pada gambar 6.48 di bawah ini.

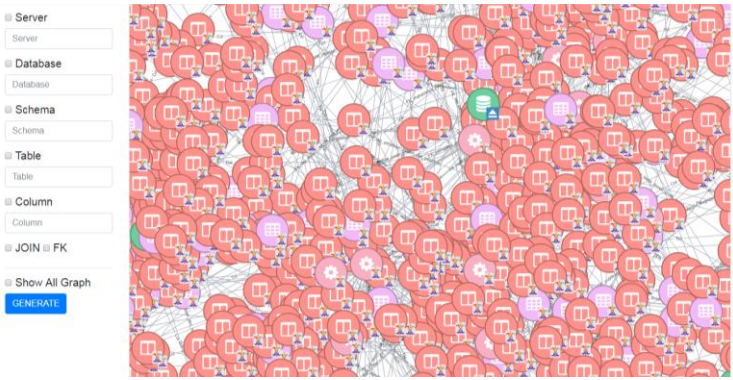


Gambar 6.48 Hasil filtering Server, Database, Schema, Table, Column, dan JOIN relationship

Dari hasil pengujian di atas dapat disimpulkan bahwa aplikasi berhasil melakukan *filtering* untuk Server, Database, Schema, Table, Column, dan JOIN relationship tertentu.

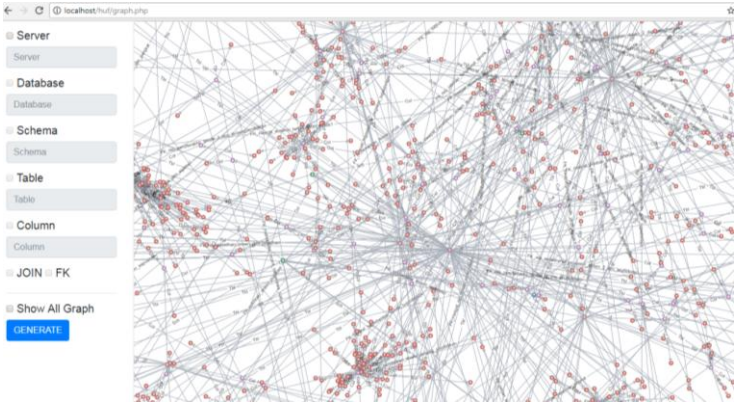
H. Show All Graph

Pengujian diawali dengan mencentang *checkbox* Show All Graph. *Checkbox* yang tercentang berhasil membuat seluruh input *checkbox* dan input teks menjadi nonaktif atau *disabled*. Pengujian dilanjutkan dengan langsung menekan tombol GENERATE. Sebagaimana ditunjukkan pada gambar 6.49, aplikasi berhasil menampilkan seluruh informasi graph yang tersimpan di dalam Neo4j.



Gambar 6.49 Hasil pengujian fitur Show All Graph

Karena graph diagram yang dihasilkan sulit dilihat maka pada source code script **graph.php,** pada bagian javascript, nodeRadius diubah menjadi 3 sehingga graph yang dihasilkan menjadi lebih kecil sebagaimana nampak pada gambar.



Gambar 6.50 Hasil pengujian fitur Show All Graph dan nodeRadius 3

Dari hasil pengujian di atas dapat disimpulkan bahwa aplikasi berhasil mengolah semua informasi node dan relationship yang tersimpan dalam graph database Neo4j.

6.2. Kesimpulan Pengujian

Berikut kesimpulan dari hasil pengujian aplikasi:

1. Aplikasi berhasil memetakan hirarki database SQL Server dalam level instance/server, database, schema, table, column, foreign key relationship, dan JOIN relationship.
2. Aplikasi berhasil membedakan relasi eksplisit berbasis foreign key dan relasi implisit berbasis JOIN.
3. Aplikasi hanya berhasil mengenali 2 jenis JOIN yaitu LEFT JOIN dan RIGHT JOIN. Selain kedua jenis tersebut, aplikasi hanya mengenalinya sebagai JOIN saja.
4. Aplikasi berhasil menyimpan data instance, database, schema, table, column, foreign key relationship, dan JOIN relationship ke dalam *graph database* Neo4j.
5. Aplikasi berhasil membedakan nama-nama table, column, dan schema yang sama di lokasi berbeda.
6. Aplikasi berhasil menghasilkan JSON yang berisi informasi *node* dan *relationship* dalam *graph* Neo4j.
7. Aplikasi berhasil mengolah file JSON dan menampilkan hasil berbentuk graph diagram yang bisa dibuka menggunakan HTTP browser.
8. Aplikasi berhasil memfilter informasi sampai pada level server, database, schema, table, column, foreign key relationship, dan JOIN relationship tertentu dengan batasan kriteria *keyword* pencarian bersifat *case-sensitive*.

Halaman sengaja dikosongkan

BAB VII

KESIMPULAN & SARAN

Bab ini berisi kesimpulan dari keseluruhan proses pengerjaan tugas akhir yang berjudul Rancang Bangung Aplikasi Visualisasi Database SQL Server dengan Dynamic Management View Berbasis Graph Neo4j untuk Memetakan Relasi Implisit pada Database. Selain kesimpulan juga dicantumkan saran dan peluang pengembangan aplikasi ini yang bisa dimanfaatkan untuk pertimbangan topik pengembangan dan penelitian selanjutnya.

7.1. Kesimpulan

Kesimpulan dari pengerjaan rancang bangun aplikasi visualisasi database berasis graph dengan menggunakan Dynamic Management View ini antara lain:

1. Aplikasi berhasil dikembangkan dan diimplementasikan dengan baik.
2. Query yang digunakan di dalam aplikasi yaitu system catalog dan dynamic management view berhasil memetakan hirarki SQL Server dan mendeteksi relationship yang bersifat eksplisit berbasis foreign key dan implisit berbasis JOIN.

7.2. Saran dan Peluang Pengembangan

Berikut adalah beberapa saran dan kemungkinan pengembangan dari aplikasi visualisasi database berbasis graph dengan menggunakan DMV:

1. Menambahkan fitur untuk mengambil dan mengolah data stored procedure dan openquery.
2. Menyempurnakan algoritma untuk melakukan *parsing*
3. Menggunakan bentuk, warna, logo, dan ukuran yang lebih variatif pada elemen hirarki database.
4. Menyempurnakan algoritma *parsing* agar bisa membedakan tipe JOIN selain LEFT dan RIGHT.
5. Mengembangkan algoritma *parsing* agar lebih akurat menentukan posisi kolom JOIN ON yang dibolak-balik.

6. Mengembangkan algoritma *parsing* agar lebih akurat dalam melakukan *parsing* pada query yang menggunakan alias.
7. Menambahkan fitur untuk melakukan crawling query pada *source code* perangkat lunak.
8. Pengembangan fitur *Application Programming Interface* (API) agar luaran yang dihasilkan aplikasi bisa dimanfaatkan oleh aplikasi eksternal.

DAFTAR PUSTAKA

[1] M. Kajko-Mattsson, "A Survey of Documentation Practice within Corrective Maintenance," *Empir. Softw. Eng.*, vol. 10, no. 1, pp. 31–55, 2005.

[2] "Stack Overflow Developer Survey 2016 Results." [Online]. Available: https://insights.stackoverflow.com/survey/2016. [Accessed: 06-Oct-2017].

[3] J. L. Ermine, "Introduction to Knowledge Management," *Trends Enterp. Knowl. Manag.*, pp. 21–43, 2010.

[4] S. Kementerian and S. Sos, *Roadmap Pengembangan Teknologi Informasi Komunikasi 2015-2020 Kementerian Pemuda dan Olahraga*. 2015.

[5] "Survey Shows SQL Dominates NoSQL in the Cloud -- Visual Studio Magazine." [Online]. Available: https://visualstudiomagazine.com/blogs/data-driver/2014/06/sql-cloud-report.aspx. [Accessed: 06-Oct-2017].

[6] B. N. Levine, C. Shields, and N. B. Margolin, "A Survey of Solutions to the Sybil Attack."

[7] W. R. King, "Knowledge Management and Organizational Learning," vol. 4, pp. 3–13, 2009.

[8] "Design Database Diagrams (Visual Database Tools) | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/en-us/sql/ssms/visual-db-tools/design-database-diagrams-visual-database-tools. [Accessed: 18-Jan-2018].

[9] "MS SQL Server Database Tool for Windows, macOS, Linux - DbVisualizer." [Online]. Available: https://www.dbvis.com/doc/sqlserver-database-

support/. [Accessed: 06-Oct-2017].

[10] "Dynamic Management Views (Transact-SQL) | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/system-dynamic-management-views. [Accessed: 06-Oct-2017].

[11] A. Huang and Q. Xue, "Exploring Implicit Relationships In a Relational," pp. 1–7, 2002.

[12] A. Aho, R. Sethi, J. Ullman, and M. S. Lam, *Compilers: Principles, Techniques, and Tools*. London: Pearson, 1986.

[13] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, 1 edition. California: O'Reilly Media, 2014.

[14] "The Competitive Dynamics of the Consumer Web: Five Graphs Deliver a Sustainable Advantage." [Online]. Available: https://www.gartner.com/doc/2081316/competitive-dynamics-consumer-web-graphs. [Accessed: 29-Sep-2017].

[15] "Staff and Leadership - Neo4j Graph Database." [Online]. Available: https://neo4j.com/staff/. [Accessed: 29-Sep-2017].

[16] "Product - Neo4j Graph Database." [Online]. Available: https://neo4j.com/product/. [Accessed: 27-Sep-2017].

[17] "DB-Engines Ranking - popularity ranking of graph DBMS." [Online]. Available: https://db-engines.com/en/ranking/graph+dbms. [Accessed: 28-Sep-2017].

[18] "Compare Neo4j Enterprise Edition vs. Community Edition." [Online]. Available: https://neo4j.com/editions/. [Accessed: 29-Sep-2017].

[19] “Neo4j’s Graph Query Language: An Introduction to Cypher.” [Online]. Available: https://neo4j.com/developer/cypher-query-language/. [Accessed: 06-Oct-2017].

[20] “Meet openCypher: The SQL for Graphs - Neo4j Graph Database.” [Online]. Available: https://neo4j.com/blog/open-cypher-sql-for-graphs/. [Accessed: 29-Sep-2017].

[21] “Graph Processing with SAP HANA 2 | SAP Blogs.” [Online]. Available: https://blogs.sap.com/2016/12/01/graph-processing-with-sap-hana-2/. [Accessed: 29-Sep-2017].

[22] “About AgensGraph and Solution- Bitnine Global Inc.” [Online]. Available: http://bitnine.net/solutions/agensgraph/?ckattempt=1. [Accessed: 29-Sep-2017].

[23] “D3.js - Data-Driven Documents.” [Online]. Available: https://d3js.org/#introduction. [Accessed: 04-Oct-2017].

[24] “Releases . d3/d3 . GitHub.” [Online]. Available: https://github.com/d3/d3/releases. [Accessed: 05-Oct-2017].

BIODATA PENULIS

Penulis lahir di Surabaya pada tanggal 19 Oktober 1993 dan merupakan anak pertama dari tiga bersaudara. Pendidikan formal yang ditempuh penulis yaitu : SD Muhammadiyah 6 Surabaya, SMP Negeri 17 Surabaya, dan SMA Negeri 16 Surabaya. Selepas menyelesaikan wajib belajar 12 tahun, penulis melanjutkan pendidikan ke jenjang perguruan tinggi mengambil S1 Jurusan Sistem Informasi (SI) Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2011.

Selama menempuh perkuliahan di jurusan Sistem Informasi, penulis aktif di beberapa kegiatan minat bakat antara lain: Karate dan Bridge. Penulis mendapat kesempatan pada tahun 2012 untuk mewakili ITS dalam Kejuaraan Nasional Bridge antar-mahasiswa ke-14 di Jakarta. Penulis juga diberi amanah untuk menjadi sekretaris internal Unit Kegiatan Mahasiswa (UKM) Bridge pada tahun 2013. Selain aktif di kegiatan minat bakat, penulis juga aktif di kegiatan Keislaman di lingkup jurusan Sistem Informasi. Penulis diberi amanah menjadi staff media Kajian Islam Sistem Informasi (KISI) pada tahun 2013 dan Kepala Departemen Kaderisasi KISI pada tahun 2014.

Penulis mengambil bidang minat Akuisisi Data dan Diseminasi Informasi (ADDI) dengan topik penelitian *graph database*, SQL Server, dan *data visualization*. Penulis dapat dihubungi lewat email : hufadz.domain@gmail.com.