Rancang Bangun Aplikasi Visualisasi Database SQL Server dengan Dynamic Management View Berbasis Graph Neo4j untuk Memetakan Relasi Implisit pada Database

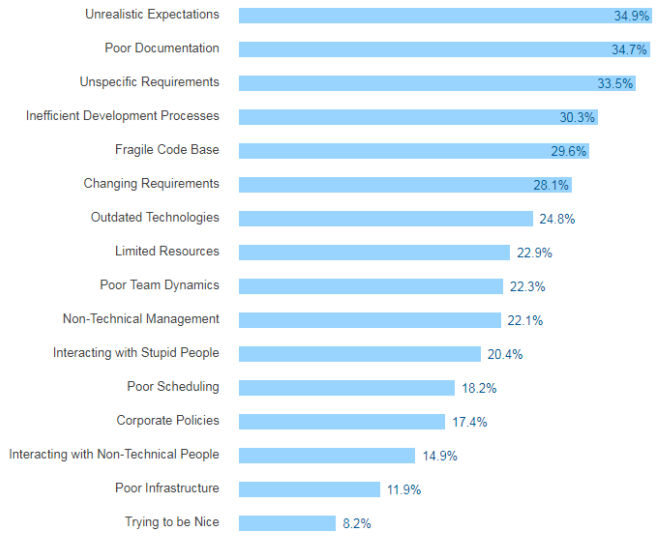
Hufadz Izzudin Robbani dan Radityo Prasetianto Wibowo
Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: radityo_pw@is.its.ac.id

***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*.**

I. PENDAHULUAN

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 [1]. Ermine dalam Introduction to Knowledge Management menjabarkan permasalahan *knowledge management* pada kasus Departemen Kesehatan Filipina disebabkan oleh lemahnya *monitoring* dan evaluasi, serta terbatasnya dokumentasi [2].



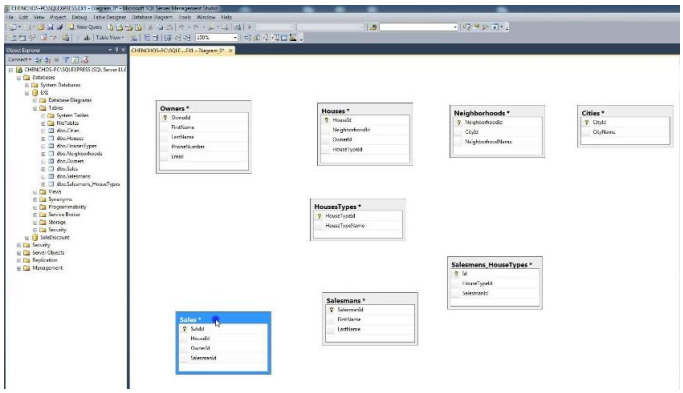
Gambar 1. Survei StackOverflow 2016.

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.

Gambar 2. Survey database terpopuler oleh Db-Engine.

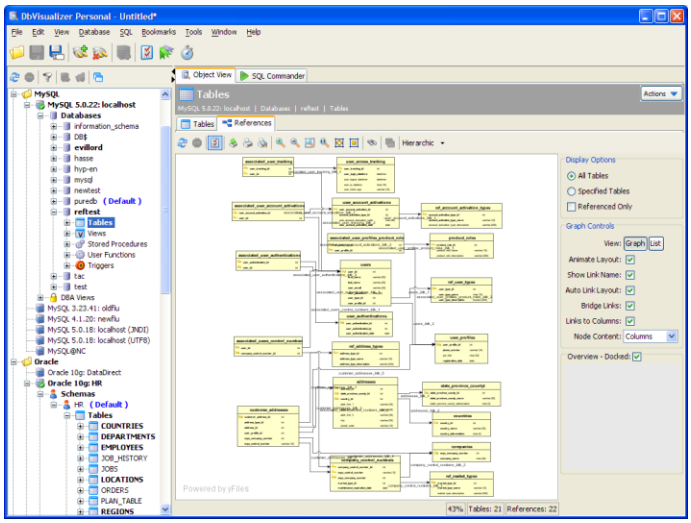
Kurangnya dokumentasi pada *platform* database SQL Server bisa diatasi dengan menggunakan bantuan aplikasi SQL Server Management Studio, sebuah aplikasi *database management*

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 [3].



Gambar 3. 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 [4]. 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 4. 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.

II. STUDI LITERATUR

A. *Penelitian Sebelumnya*

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*.

B. *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 [5].

C. *Implicit Relationship*

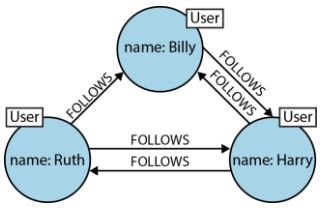
Untuk membuat relasi antar entitas data dalam database, *primary key* dan *foreign* didefinisikan. Namun dalam kasus nyata adanya *primary key* dan *foreign key* hanya menjelaskan sebagian dari relasi keseluruhan. Banyak relasi dimodelkan dengan view dan melakukan *selection query* pada tabel dan view. 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 [6].

D. *Parsing & syntax analysys*

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* [7].

E. *Graph*



Gambar 5. Contoh graph sederhana.

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 [8].

F. *Neo4j graph DBMS*

Neo4J adalah sistem manajemen basis data graph yang dikembangkan oleh Neo4J, Inc., sebuah perusahaan yang berbasis di San Francisco Bay Area, Amerika Serikat [9]. 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 [10]. Neo4J menggunakan bahasa query bernama Cypher. Cypher adalah sebuah bahasa deklaratif yang terinspirasi dari SQL untuk mendeskripsikan pola graph [11]. Cypher pada awalnya dikembangkan eksklusif untuk Neo4J tapi kemudian dengan dirilisnya openCypher [12] kini Cypher bisa diterapkan pada SAP Hana [13] dan AgensGraph [14]. Cypher Query mendefinisikan node relationship dengan syntax CREATE dan MATCH.

G. *D3.js*

D3.js adalah library javascript berlisensi BSD untuk manipulasi dokumen berbasis data. D3 adalah kepanjangan dari Data-Driven Documents [15]. D3.js memungkinkan untuk mengikat *arbitrary data* ke sebuah Document Object Model (DOM) yang selanjutnya dapat diterapkan *data-driven transformation*.

III. RANCANGAN APLIKASI

A. *Analisis Kebutuhan*

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.

B. *Query*

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 adalah system catalog.

```
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
```

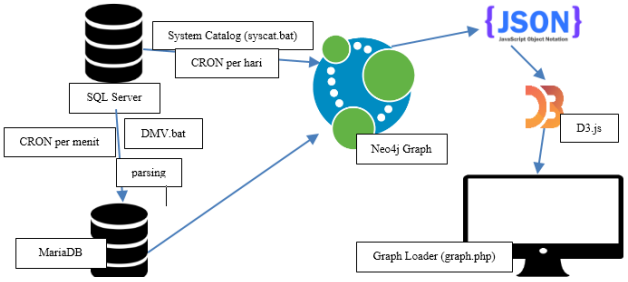
Query ketiga yang digunakan adalah query sys.databases yaitu query yang digunakan untuk mendapatkan informasi database yang tersedia pada server. Informasi yang dibutuhkan hanya nama database. Database yang didapat haruslah database yang dibuat oleh user dan bukan database sistem atau database reporting.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

C. *Desain Arsitektur*

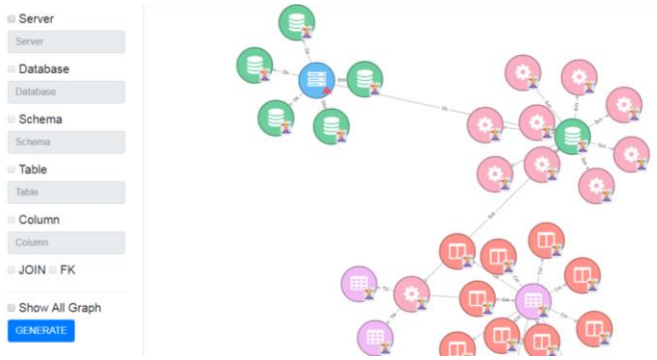


Gambar 6. Desain Arsitektur

Proses pertama adalah proses mendapatkan seluruh informasi server, database, schema, table, column, dan foreign key realtionship pada database SQL Server dengan query pertama dan kedua. Informasi yang diperoleh dari proses ini selanjutnya dimasukkan ke graph Neo4j menggunakan bahasa query cypher. Proses pertama diotomatisasi untuk berjalan selama setiap hari sekali. Proses kedua adalah proses mendapatkan daftar query yang pernah dijalankan pada database SQL Server. Pada proses inilah query ketiga dijalankan. Singkatnya proses ini adalah melakukan query ke SQL Server untuk mendapatkan history query pada sistem. 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.

D. *Desain Antarmuka*

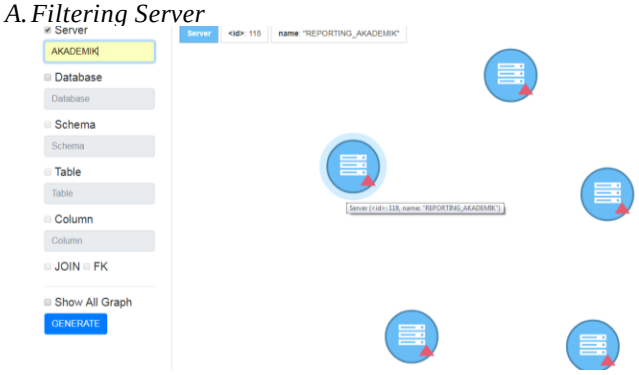
Antarmuka yang tersedia hanyalah pada script *graph loader*. Antarmuka pada script ini berfungsi menyajikan graph dalam bentuk diagram yang bisa diakses pada browser HTTP biasa. Tampilan secara umum terbagi menjadi ruas kanan dan kiri. Ruas kiri berfungsi untuk memasukkan kriteria *filtering*. Pada ruas 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. Pada 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 7.



Gambar 7. Desain antarmuka keseluruhan.

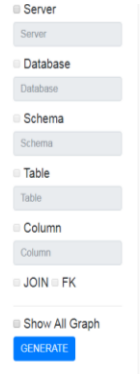
IV. PENGUJIAN

A. *Filtering Server*



Gambar 8. 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. 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 9 yaitu tidak ada graph yang dihasilkan.



Gambar 9 Hasil filtering Server dengan keyword 'akademik'

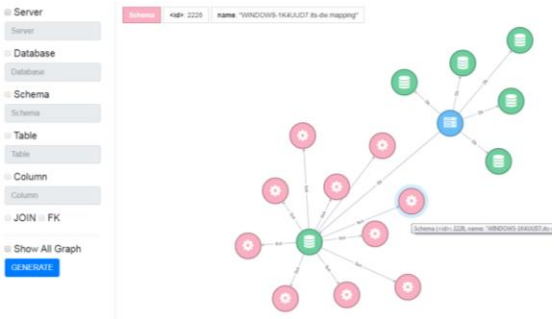
B. Filtering Server dan Database

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 10. Hasil pengujian filtering Server dan Database.

C. Filtering Server, Database, dan Schema

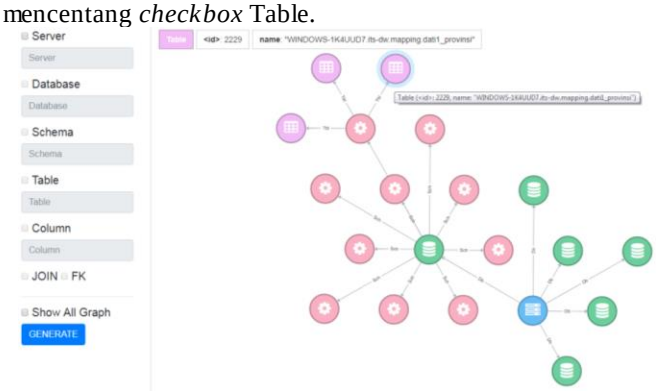
Aktivitas pengujian sama seperti pada pengujian B dengan ditambahkan aktivitas mencentang *checkbox* Schema.



Gambar 11. Hasil pengujian filtering Server, Database, Schema.

D. Filtering Server, Database, Schema dan Table

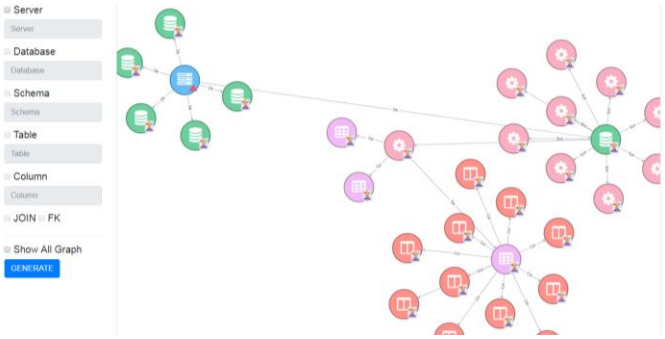
Aktivitas pengujian sama seperti pada pengujian C dengan ditambahkan aktivitas mengisi input *text* Schema dan mencentang *checkbox* Table.



Gambar 12. Hasil filtering Server, Database, Schema, Table.

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.

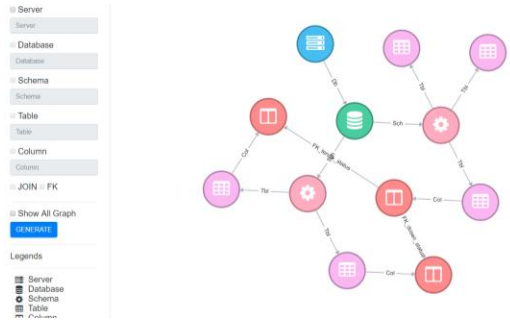


Gambar 13. Hasil filtering Server, Database, Schema, Table, Column.

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.

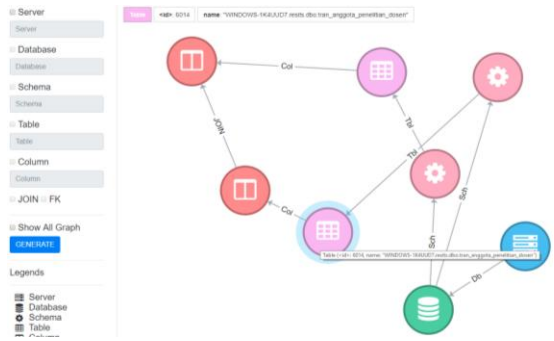


Gambar 14. Hasil filtering Server, Database, Schema, Table, Column, dan FK relationship.

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.



Gambar 15. Hasil filtering Server, Database, Schema, Table, Column, dan JOIN relationship.

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. Hasil pengujian fitur Show All Graph.

V. KESIMPULAN & SARAN

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 berhasil memetakan hirarki SQL Server dan mendeteksi relationship yang bersifat eksplisit berbasis foreign key dan implisit berbasis JOIN.

Berikut adalah beberapa saran dan kemungkinan pengembangan dari aplikasi ini:

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

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

[3] "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].

[4] "MS SQL Server Database Tool for Windows, macOS, Linux - DbVisualizer." [Online]. Available: https://www.dbvis.com/doc/sqlserver-database-support/. [Accessed: 06-Oct-2017].

[5] "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].

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

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

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

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

[10] "DB-Engines Ranking - popularity ranking of graph DBMS."

[Online]. Available: https://db-engines.com/en/ranking/graph+dbms. [Accessed: 28-Sep-2017].
[11] "Neo4j's Graph Query Language: An Introduction to Cypher." [Online]. Available: https://neo4j.com/developer/cypher-query-language/. [Accessed: 06-Oct-2017].
[12] "Meet openCypher: The SQL for Graphs - Neo4j Graph Database." [Online]. Available: https://neo4j.com/blog/open-cypher-sql-for-graphs/. [Accessed: 29-Sep-2017].
[13] "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].
[14] "About AgensGraph and Solution- Bitnine Global Inc." [Online]. Available: http://bitnine.net/solutions/agensgraph/?ckattempt=1. [Accessed: 29-Sep-2017].
[15] "D3.js - Data-Driven Documents." [Online]. Available: https://d3js.org/#introduction. [Accessed: 04-Oct-2017].