PEMODELAN GRAPH DATABASE UNTUK MODA TRANSPORTASI BUS RAPID TRANSIT

Panji Wisnu Wirawan[1], Djalal Er Riyanto[2], Khadijah[3]

Departemen Ilmu Komputer/Informatika, Fakultas Sains dan Matematika,Unversitas Diponegoro
Jl. Prof Soedarto, SH, Tembalang, Semarang, (024)747 4754
Email : maspanji@undip.ac.id[1] , erriyanto@undip.ac.id[2] , khadijah0303@gmail.com[3]

Abstrak

Bus Rapid Transit (BRT) merupakan salah satu alternatif transportasi massal. Rute BRT memiliki karakteristik khusus yang dapat dimodelkan dengan graph. Ketika rute dan shelter semakin bertambah, dibutuhkan aplikasi komputer untuk melakukan pencarian rute BRT. Hal tersebut akan memudahkan pencarian dan penjelajahan rute-rute BRT. Namun, ketika rute diimplementasikan menggunakan basis data relasional, performa query dapat menurun karena banyaknya operasi JOIN untuk mencari rute. Artikel ini mengusulkan sebuah model graph database untuk BRT dan implementasinya. Identifikasi kebutuhan data dilakukan, dilanjutkan dengan pemodelan menggunakan entity relationship (ER). Hasil ER tersebut kemudian dipetakan ke dalam property graph untuk kemudian diimplementasikan menggunakan produk graph database Neo4J. Hasil penelitian ini menunjukkan bahwa model yang dibuat bisa diterapkan dalam basis data graph dan graph dapat menunjukkan rute BRT tertentu. Dari sisi performance, basis data graph menunjukkan kinerja perambatan yang lebih baik dibandingkan dengan basis data relasional.

Keyword : BRT, *graphdatabase*

1. PENDAHULUAN

Masyarakat di beberapa kota di Indonesia saat ini telah memiliki alternatif transportasi massal yang dinamakan dengan Bus Rapid Transit (BRT). BRT hadir dengan berbagai nama seperti Trans Jakarta (Jakarta), Trans Semarang (Semarang), Trans Jogja (Yogyakarta), Batik Solo Trans (Solo), Trans Musi (Palembang) dan yang lainnya. Hadirnya BRT merupakan upaya untuk menyediakan transportasi yang dapat menambah penggunaan transportasi umum, menambah tingkat dan kualitas layanan transportasi publik (Mzee & Chen, 2010).

Sistem BRT di Indonesia memiliki *shelter* / halte tertentu. *Shelter* tersebut memiliki beberapa tipe, misalnya *transit point* dan *transfer point*. Selain itu, setiap bus pada BRT memiliki rute-rute perjalanan tertentu yang biasanya disebut dengan koridor, di mana setiap bus melewati *shelter* yang telah ditentukan. Dengan demikian BRT merupakan sebuah jaringan transportasi. Pemodelan data terkait dengan jaringan transportasi ini penting ketika akan membuat sebuah program/aplikasi komputer.

Basis data relasional telah lama menjadi standar penyimpanan data pada program/aplikasi komputer. Dalam basis data relasional, data disimpan dalam tabel dan kolom di mana antar tabel dapat saling dikaitkan melalui kunci. Apabila diinginkan data dari tabel yang berbeda, dilakukan permintaan (*query*) menggunakan klausa JOIN. Untuk jaringan transportasi, jika disimpan dalam basis data relasional, akan memerlukan banyak operasi JOIN *query* yang menyebabkan komputasi tinggi (Dominguez-Sal et al., 2010). Dengan demikian performa aplikasi dapat menurun.

Graphdatabase dapat menjadi solusi alternatif untuk menyimpan data jaringan transportasi. Hal tersebut disebabkan karena *graphdatabase* menyimpan data seperti halnya *graph*, yaitu dalam *node-node* yang berhubungan satu sama lain. Artinya, analisis jaringan transportasi dapat

memanfaatkan operasi *graph* seperti menemukan daerah sekitar (*neighbourhood*), penjelajahan (*traversal*) jalur transportasi, serta menemukan lintasan terpendek (Dominguez-Sal et al., 2010). Artikel ini mengusulkan sebuah model dari *graphdatabase* yang dapat dimanfaatkan untuk jaringan transportasi Bus Rapid Transit. Disertakan pula contoh implementasi dari *graphdatabase* tersebut menggunakan salah satu produk *graphdatabase* yaitu Neo4J (Neo4J, n.d.). Diharapkan dari model tersebut, dapat berkontribusi dalam memudahkan pengembangan program/aplikasi yang membutuhkan penyimpanan data jaringan transportasi menggunakan GDB.

2.TINJAUAN PUSTAKA

2.1.GRAPH

Graph adalah konsep matematika untuk menyajikan suatu himpunan objek, di mana pasangan dari objek terhubung dengan suatu penghubung (*link*). Suatu *graph* berarah G = (V, E) didefinisikan sebagai suatu himpunan *node* V dan himpunan *edge* E dengan ukuran m. Setiap *edge* terhubung ke dua *node*, satu *node* sebagai sumber dan yang lainnya sebagai target. Tergantung pada skenario yang disiapkan, suatu *edge* dapat memiliki beberapa atribut tambahan. Biasanya, suatu *edge* e € E secara unik diidentifikasi oleh sumbernya yaitu u € V dan target dari v € V. Dalam hal terdapat *edge* paralel dengan sumber dan target yang sama, atribut tambahan dari suatu *edge* atau konteks diperlukan untuk suatu identifikasi unik.

Suatu *path* P di dalam G ialah suatu urutan *edge* (e_1, ..., e_k), sedemikian rupa sehingga target dari e_i dan sumber dari e_{i+1} adalah sama untuk semua $1 \leq i \leq (k-1)$. Dalam hal *graph* tidak mempunyai *edge* paralel atau jika bebas dari konteks, dapat pula disajikan urutan *node* (u_1, ..., u_k, u_{k+1}), di mana u_i adalah sumber dari *edge* e_i, untuk semua $1 \leq i \leq k$, dan u_{k+1} adalah target dari e_k. Dalam hal ini u_1 adalah sumber dari P dan u_{k+1} adalah target dari P.

Graph database memodelkan data sebagai data yang saling terhubung. Tidak seperti basis data relasional yang menyimpan data dalam tabel-tabel, basis data graf menyimpan data dalam bentuk *nodes* dan hubungan antar datanya menggunakan *edges*. Setiap *node*, secara internal, memiliki pointer ke *node* yang lain ataupun ke subgraf (Celko, 2014).

Graph database adalah suatu sistem manajemen basis data online dengan metoda *Create*, *Read*, *Update* dan *Delete* (CRUD) yang menampakkan mode data *graph*. Basis data ini dibangun dengan menggunakan sistem *Online Transaction Processing* (OLTP), sehingga akan memenuhi *transactional integrity* dan *availability*.

Query dengan banyak join (*joint-intensive*) pada *database* relasional akan menurunkan kinerja khususnya untuk data yang bervolume besar. Akan tetapi, kinerja *graphdatabase* cenderung tetap *relative* konstan, meskipun terjadi pertumbuhan atau peningkatan data. Hal ini disebabkan karena pada umumnya *query* terbatas pada suatu *segment* dari *graph*. Waktu eksekusi untuk setiap *query* hanya proporsional ke ukuran dari *segmentgraph* yang terkait dengan *query* dan tidak dengan ukuran totalnya.

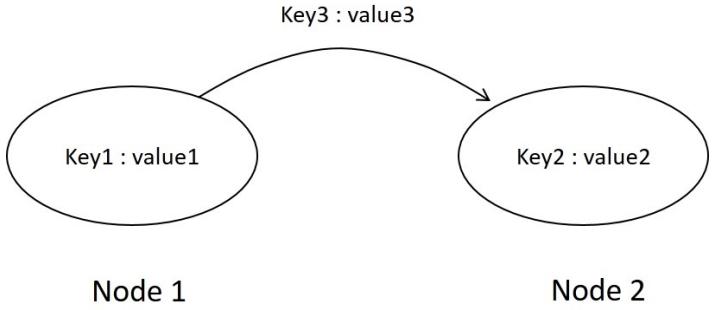
Pada sembarang kesempatan dapat dilakukan penambahan *node* dan *relationship* baru ke dalam model *graph*. Penambahan *node* dan *relationship* ke struktur yang sekarang ada tidak akan mengganggu fungsi *query* dan fungsi aplikasi.

2.2.GRAPH DATA MODEL

Model merupakan representasi konseptual dari sesuatu yang akan dibangun/dilaksanakan. Sehingga dengan menggunakan model, apa yang akan dibangun/dilaksanakan diharapkan dapat dikomunikasikan dengan baik. Dari sudut pandang *database*, setidaknya model dapat menggambarkan struktur data, deskripsi, *maintenance*, dan bagaimana memperoleh data tersebut (Angles & Gutierrez, 2008).

Angles & Gutierrez (2008), melakukan kajian mengenai *graph* data model. Beberapa model dikaji dan salah satunya adalah GDM (*Graph* Data Model). GDM merupakan data model untuk *graph* yang diusulkan oleh Hidders (2003). Dalam model tersebut, terdapat *instance graphs* dan *schema graphs*. Hidders (2003) juga memperkenalkan dua operasi manipulasi *graph*, yaitu pendambahan dan penghapusan.

Pada GDM, *instance graph* merupakan *graph* berlabel dimana *node* menggambarkan entitas, sedangkan *edge*s mencerminkan atribut untuk setiap entitas. *Node-node* yang dimiliki oleh beberapa class, diberi label dari nama kelas yang memilikinya. Lain halnya dengan *Schema graphs*, *node* pada schema *graph*s menggambarkan class sedangkan *edge* yang diberi label dengan nama atribut menandakan bahwa entitas dari class memiliki atribut tersebut. Konsep data model yang diberikan pada GDM masih mengadopsi dari konsep berorientasi objek seperti class, instance, dan inheritance.
Pendekatan data model *graph* yang lain dilakukan oleh De Virgilio, Maccioni, & Torlone, (2014). Model *graph* dibentuk dengan terlebih dahulu membentuk Entity Relationship Diagram (ERD). ERD kemudian ditransformasikan menjadi model *graph*, yang disebut *property graph*. *Propertygraph* merupakan merupakan 'multi*graph*' dimana *node* maupun *edge* diberi label dengan data yang merupakan pasangan kunci-nilai (key-value pair).



Gambar 1. *Property graph*

Lebih lanjut, De Virgilio et al. (2014) memberikan panduan untuk transformasi ERD menjadi model *graph* sebagai berikut :

a. Relasi one-to-one bertransformasi menjadi *graph* dengan *edge* memiliki dua arah (*double directed*)
b. Relasi one-to-many bertransformasi menjadi *graph* dengan *edge* memiliki satu arah, dari entitas berkardinalitas rendah ke tinggi.
c. Relasi many-to-many bertransformasi menjadi *graph* dengan *edge* dua arah (*double directed*) dengan bobot yang berbeda dengan relasi one-to-one.

3.METODE PENELITIAN

Penelitian ini menggunakan beberapa tahapan dalam membentuk model *graphdatabase* sebagai berikut :

1. Identifikasi kebutuhan data.
   Pada tahap ini dilakukan identifikasi data yang diperlukan untuk membentuk sebuah rute bus rapid transit. Sebagai studi kasus, dipilih rute pada Bus Rapid Transit Semarang.
2. Pemodelan *graphdatabase*.
   Berdasarkan kebuthan data yang telah diidentifikasikan, dibuatlah model *graphdatabase*. Model yang dipilih adalah model yang dikemukakan oleh De Virgilio et al. (2014).
3. Implementasi model.
   Neo4J akan digunakan sebagai *graphdatabase* untuk mengimplementasikan model yang telah dikembangkan.

4.HASIL DAN PEMBAHASAN

4.1.KEBUTUHAN DATA

Identifikasi kebutuhan data diperlukan untuk mendapatkan pemodelan data yang tepat. Pada bagian ini, akan diidentifikasi kebutuhan data yang diperlukan untuk rute Bus Rapid Transit. Sebagai studi kasus, dipilih Bus Rapid Transit Semarang.
BRT memiliki rute berbasis koridor atau jalur-jalur yang dilewati. Rute BRT Trans Semarang mencakup empat koridor. Sepanjang rute terdapat sejumlah tempat pemberhentian bus yang disebut dengan *shelter*. Terdapat dua jenis *shelter*, yaitu: a) *sheltertransit point*, dan b) *sheltertransfer point*. Kedua *shelter* tersebut mempunyai fungsi utama yang sama, yaitu untuk tempat menurunkan atau menaikkan penumpang.
Sheltertransfer point digunakan untuk menaikkan dan menurunkan penumpang yang berasal dan atau mempunyai tujuan yang sesuai dengan rute BRT yang dinaiki. Sedangkan untuk *sheltertransit point*, titik lokasi *shelter* berada pada lokasi yang memungkinkan penumpang meneruskan perjalanan ke rute lain yang di luar rute BRT, atau pindah ke moda angkutan lain (bus umum, pesawat udara, kereta api, kapal laut).
Berdasarkan informasi tersebut, maka dapat ditentukan data yang diperlukan untuk rute BRT yaitu:
(a) *Shelter* sebagai *node* ; dan
(b) Jalur penghubung *shelter* sebagai *edge*, yang merupakan bagian dari suatu koridor.

4.2.PEMODELAN GRAPH DATABASE UNTUK BRT

Graphdatabase disusun dari *node* dan *relationship*. Baik *node* maupun *relationship*, dapat memiliki satu atau lebih atribut yang dapat menjelaskan mengenai *node* atau *relationship* tersebut. Dalam konteks BRT Semarang, pembentukan *node* dan *relationship* berikut :

a. *Shelter* dapat dimodelkan menggunakan *Node*. Hal ini dapat dilakukan karena *shelter* merupakan titik dimana BRT berhenti untuk menurunkan atau menaikkan penumpang. Masing-masing *shelter* BRT terletak di lokasi tertentu (memiliki nama), dan memiliki tipe *shelter* tersendiri yaitu sebagai *shelter transfer point* ataupun *shelter transit point*.

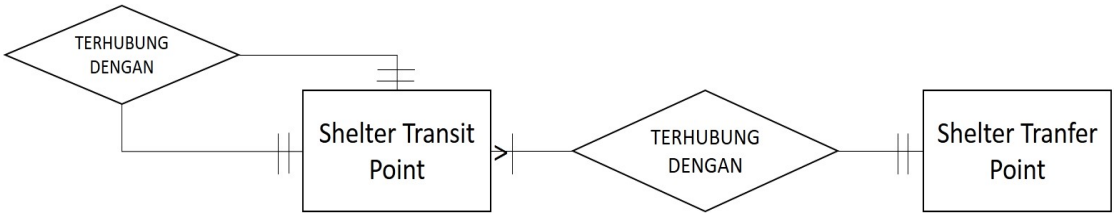
b. Koridor dapat dimodelkan dengan *Relationship*. *Relationship* yang menghubungkan antar *node* bisa digunakan untuk mengasosiasikan hubungan antara *shelter-shelter* yang terhubung menggunakan koridor bus BRT yang ada. Setiap koridor memiliki nama yang menunjukkan rute BRT.

Secara ringkas, data beserta *property* untuk masing masing *shelter* dan relationship ditunjukkan tabel 1.

Tabel 1. *Node* dan *property* pada *graphdatabase*

Data	Tipe	*Property*
ShelterTransit point	*Node*	SID (*shelter* ID) Nama *Shelter*
ShelterTransfer point	*Node*	SID (*shelter* ID) Nama *Shelter*
Koridor	Relationship	Nama koridor

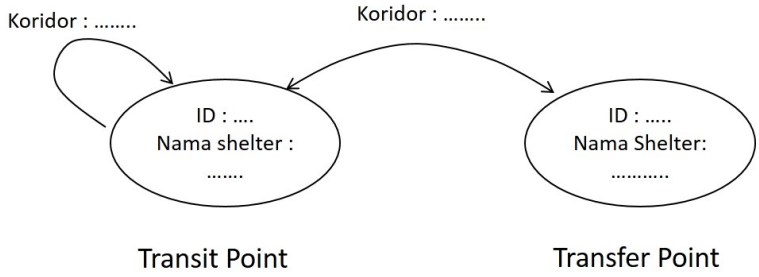
Dari sisi pemodelan ERD, terdapat dua buah entitas yaitu *sheltertransit point* dan *sheltertransfer point*. Sedangkan koridor akan menjadi relationship. Relationship menghubungan antara satu *node* (*shelter*) dengan *node* yang lain. *Node* yang merupakan *transit point* terhubung dengan satu *transit point* yang lain. Sedangkan untuk *transfer point*, dapat terhubung dengan lebih dari *transit point*. Sehingga hubungan ERD yang tepat untuk *sheltertransit point* dan *sheltertransfer point* adalah one-to-many. Untuk hubungan antar *transit point* adalah one-to-one.



Gambar 2. ERD untuk BRT

Selanjutnya, ERD yang dihasilkan digunakan untuk membentuk model *propertygraph*. Akan terdapat dua *node* yaitu *sheltertransit point* dan *sheltertransfer point*. *Transit point* dengan *transfer point* memiliki hubungan many-to-one sehingga hubungan antar keduanya memiliki dua arah (bidirectional). Antar *transit point* memiliki hubungan dengan satu *transit point* yang lain dan searah sehingga antar *transit point* memiliki hubungan satu arah (uni directional).

Property dari masing-masing *node* dan relationship kemudian ditempatkan pada posisinya masing-masing. Sehingga didapatkan *propertygraph* seperti pada gambar 3.



Gambar 3. Hasil model *graphdatabase* untuk BRT

Berikutnya, *propertygraph* tersebut akan diimplementasikan dalam sebuah *graphdatabase*, untuk membuktikan bahwa model tersebut telah sesuai dengan kebutuhan.

4.3.IMPLEMENTASI MODEL

Model gambar 3 diimplementasikan menggunakan *graphdatabase* Neo4J dengan mengambil data pada semua koridor BRT TransSemarang, yaitu koridor I, II, IIIA,IIIb dan koridor IV. Secara keseluruhan terdapat 206 *shelter*, yang terdiri dari 16 *sheltertransit point* dan 190 *sheltertransfer point*. Masing-masing memiliki nama *shelter* yang dapat digunakan sebagai data pada *propertygraph*.

Neo4j memiliki *query* yang disebut Cypher *query*. *Query* tersebut selanjutnya akan digunakan untuk membentuk transfer maupun *transit point*. *Query* Q1 dan Q2 adalah contoh cypher *query* yang berturut-turut digunakan untuk membentuk *transfer point* dan *transit point*. *Query* Q1 merupakan pembentukan *node* dengan label TransferPoint dan *shelter* id (sid) 1 dan nama *shelter* Balai Kota. *Query* Q2 akan membentuk *node* dengan label TransitPoint dan *shelter* id 10 dan nama *shelter* Pekunden. *Query* Q1 dan Q2 diulang untuk setiap *shelter* yang bersesuaian sehingga terbentuklah *node-node* yang diperlukan pada semua koridor.

Q1 : membentuk *transit point*

```
CREATE  (s:shelter:TransitPoint{sid:1,nama_shelter:"Balai  Kota"
}) RETURN shelter
```

Q2 : membentuk *transfer point*

```
CREATE   (s:shelter:TransferPoint{sid:10,nama_shelter:"Pekunden"
}) RETURN shelter
```

Ketika *node-node* telah terbentuk, langkah selanjutnya adalah membentuk *relationship* untuk setiap *node*. *Query* Q3 merupakan *query* untuk membentuk *relationship* antar *node*. *Query* tersebut membentuk relationship antara *shelter* Pandanaran 2 dan Pekunden. Karena antara Pandanaran 2 dan Pekunden merupakan *sheltertransfer point*, maka arah *graph* pada *query* tersebut adalah searah, dari dari Pandanaran 2 ke Pekunden.

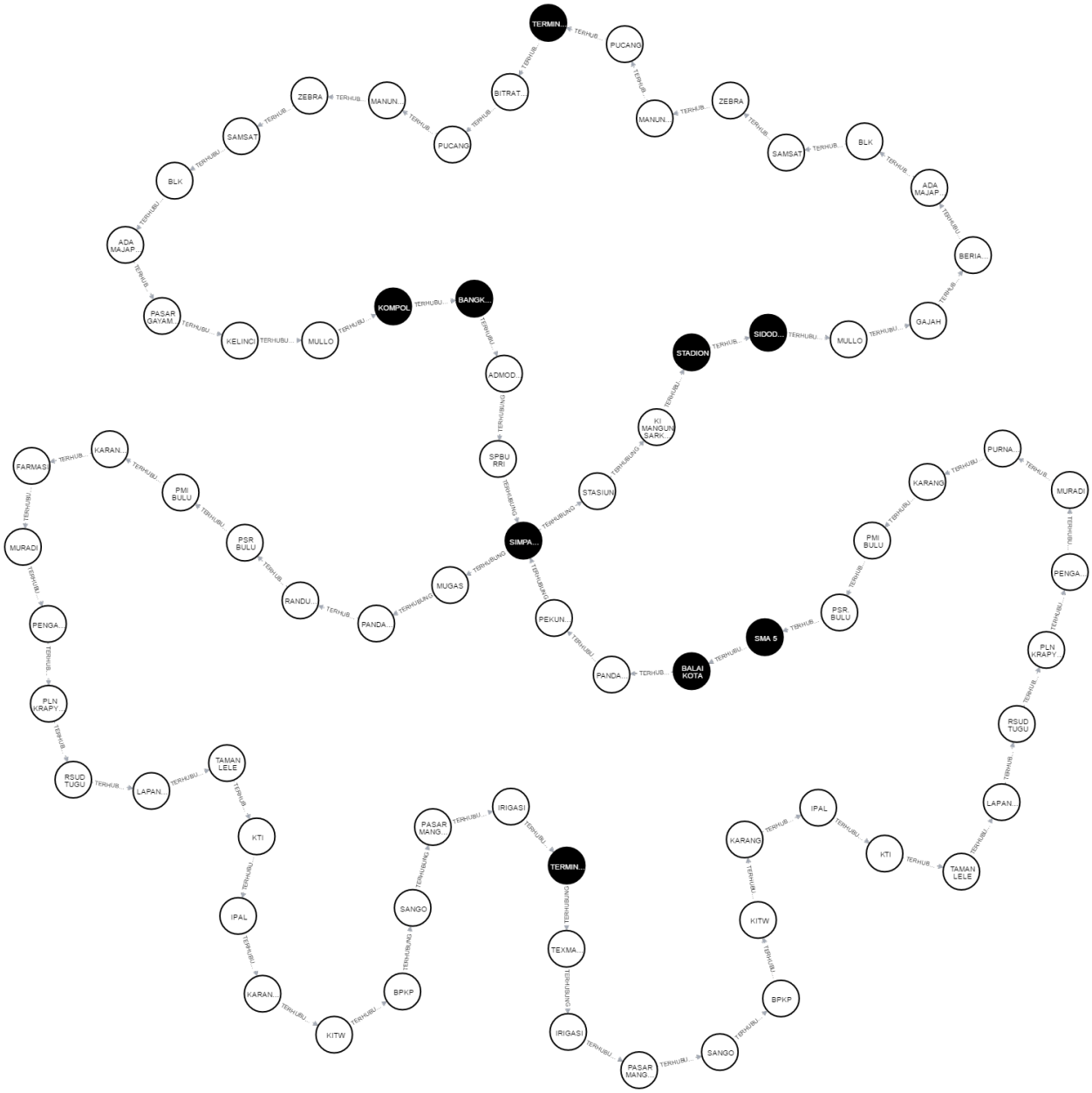
Q3 : membentuk *relationship* antar *transfer point*

```
MATCH (a:shelter),(b:shelter)
WHERE a.nama_shelter = "Pandanaran 2" AND
b.nama_shelter="Pekunden"
CREATE (a)-[m:TERHUBUNG{koridor:"KORIDOR 1"}]->(b)
```

Query Q3 bekerja dengan cara menemukan *shelter* dengan nama Pandaranan 2 dan Pekunden, memberikan label jika ditemukan, kemudian membuat relationship antara keduanya. Relasi antara keduanya adalah TERHUBUNG dengan *property* koridor dengan data KORIDOR 1.

Koridor 1 pada BRT Trans Semarang melewati 75 *shelter*, yang terdiri dari 9 *sheltertransit point* dan 66 *sheltertransfer point*. Masing-masing *shelter* dibentuk dengan *query* Q1 dan Q2 dan hubungan antar *shelter* dibuat menggunakan Q3. Hasil dari *query-query* yang telah diimplementasikan pada Neo4J untuk koridor 1, ditunjukkan pada gambar 4.

Gambar 4 merupakan hasil visualisasi yang dilakukan oleh Neo4J, tidak menunjukkan posisi *shelter* pada posisi sebenarnya. Pada gambar 4, *node* (ditunjukkan dengan lingkaran) dengan warna hitam merupakan *transit point* sedangkan *node* dengan warna putih merupakan *transfer point*. Jalur yang menghubungan *shelter* (*edge*) ditunjukkan dengan garis yang memiliki arah. Visualisasi tersebut menunjukkan bahwa model yang dihasilkan dari penelitian ini dapat membentuk *shelter* dan jalur antar *shelter* sedemikian hingga membentuk sebuah koridor atau rute perjalanan BRT.

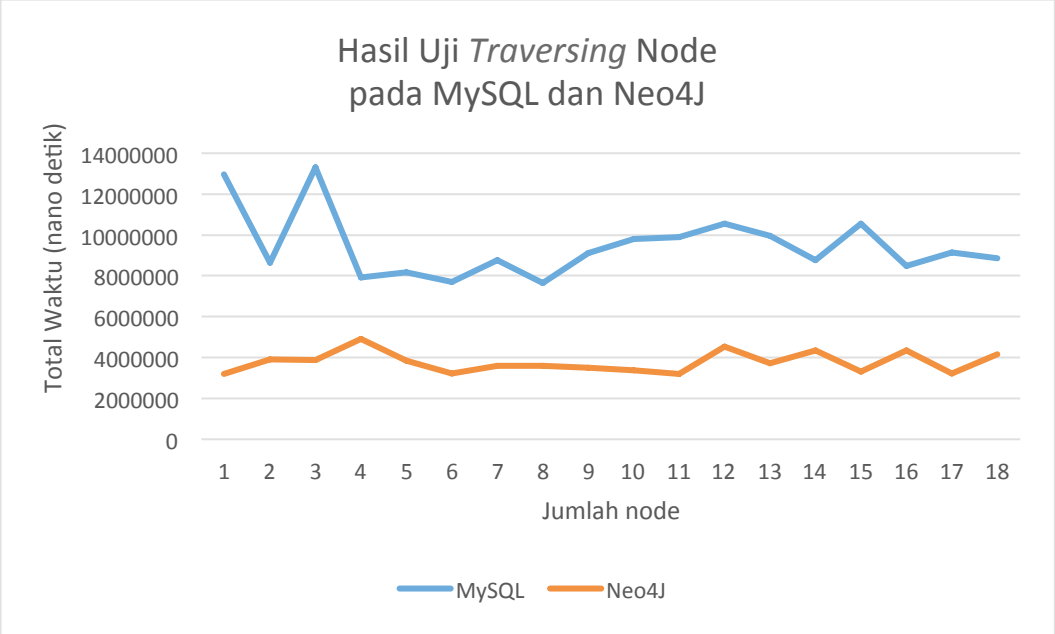


Gambar 4. Hasil implementasi pada Neo4J pada koridor 1.

4.4.PENGUJIAN PERFORMA

Pengujian performa dilakukan untuk mengetahui seberapa cepat respon *graph database* dalam melakukan proses perambatan (*traversing*) shelter. Sebagai pembanding, model dari data rute bus TransSemarang diimplementasikan pula pada sistem pengelolaan basis data relasional MySQL, berpedoman pada gambar 2.

Pengujian dilakukan dengan mengakses kedua data, yaitu pada MySQL dan Neo4J, menggunakan program Java dengan *driver* standar (*official*) untuk masing-masing sistem pengelolaan basis data MySQL dan Neo4J. Akses data dilakukan untuk melakukan perambatan 18 node pada koridor IV, dimulai dari terminal Cangkiran. Pengukuran waktu permintaan (*query*) dimulai dari ketika program memulai permintaan sampai dengan menutup koneksi basis data. Satuan waktu yang digunakan dalam pengujian ini adalah dalam nano detik. Hasil pengujian ditunjukkan pada gambar 5.



Gambar 5. Hasil uji *tranversing* node pada koridor IV.

Gambar 5 menunjukkan bahwa untuk setiap jumlah node yang diuji, MySQL memiliki waktu permintaan (*query*) untuk perambatan node yang lebih lama dibandingkan dengan Neo4J. Hasil tersebut menunjukkan pula bahwa untuk data yang memerlukan proses perambatan seperti pada data transportasi, lebih tepat ketika basis data yang digunakan adalah basis data dengan model data graph.

5.KESIMPULAN

Model *graphdatabase* dapat dibuat untuk BRT menggunakan *propertygraph*. Hasil penelitian ini menunjukkan bahwa untuk studi kasus BRT TransSemarang, model *graph* yang dihasilkan adalah adanya *node* yang merupakan *sheltertransit point* dan *sheltertransfer point* pada model yang dihasilkan. Satu *sheltertransit point* terhubung dengan satu *sheltertransit point* yang lain, sedangkan satu *sheltertransfer point* dapat terhubung dengan banyak *sheltertransit point*. Model yang dibuat kemudian dapat diimplementasikan menggunakan basis data *graph* Neo4J. Hasil pengujian menunjukkan bahwa performa basis data graph untuk masalah perambatan (*traversing*) menunjukkan hasil yang lebih baik daripada menggunakan basis data relasional.

6.DAFTAR PUSTAKA

Angles, R., & Gutierrez, C. (2008). Survey of *graphdatabase* models. *ACM Computing Surveys*, *40*(1), 1–39. http://doi.org/10.1145/1322432.1322433

Celko, J. (2014). NoSQL and Transaction Processing. In *Joe Celko's Complete Guide to NoSQL* (pp. 1–14). Elsevier. http://doi.org/10.1016/B978-0-12-407192-6.00001-7

De Virgilio, R., Maccioni, A., & Torlone, R. (2014). Model-Driven Design of *GraphDatabase*s (pp. 172–185). http://doi.org/10.1007/978-3-319-12206-9_14

Dominguez-Sal, D., Urbón-Bayes, P., Giménez-Vañó, A., Gómez-Villamor, S., Martínez-Bazán, N., & Larriba-Pey, J. L. (2010). Survey of *GraphDatabase* Performance on the HPC Scalable *Graph* Analysis Benchmark (pp. 37–48). http://doi.org/10.1007/978-3-642-16720-1_4

Hidders, J. (2003). Typing *Graph*-Manipulation Operations. In *Database Theory --- ICDT 2003: 9th International Conference Siena, Italy, January 8--10, 2003 Proceedings* (pp. 394–409). http://doi.org/10.1007/3-540-36285-1_26

Mzee, P. K., & Chen, Y. (2010). Implementation of Bus Rapid Transit System as an Alternative for Public Transportation in Developing Countries Case of Dart System in Dar Es Salaam. In *2010 International Conference on Intelligent Computation Technology and Automation* (pp. 489–493). IEEE. http://doi.org/10.1109/ICICTA.2010.233

Neo4J. (n.d.). Neo4J : The World's Leading *GraphDatabase*. Retrieved from http://neo4j.com