Terbit *online* pada laman web jurnal : http://teknosi.fti.unand.ac.id/

Jurnal Nasional Teknologi dan Sistem Informasi

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Artikel Penelitian

Kajian Implementasi Graph Database pada Rute Bus Rapid Transit

Panji Wisnu Wirawan [a] , *Djalal Er Riyanto* [b]

[a,b] *Departemen Ilmu Komputer/Informatika Universitas Diponegoro, Jl Prof. Soedarto, Semarang, 50275*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 18 Agustus 2017
Revisi Akhir: 08 November 2017
Diterbitkan *Online*: 31 Desember 2017

KATA KUNCI

Bus Rapid Transit,

Graph Database,

Algoritma.

KORESPONDENSI

Telepon: +62 (024) 7474754

E-mail: maspanji@undip.ac.id

A B S T R A C T

Bus Rapid Transit (BRT) merupakan salah satu sarana transportasi publik yang memiliki rute perjalanan tertentu atau disebut sebagai koridor. Satu koridor BRT dengan koridor yang lain bukanlah koridor yang terpisah, melainkan saling terhubung. Dalam melakukan perjalanan, penumpang BRT boleh jadi melakukan perpindahan koridor melalui shelter. Informasi tersebut perlu didapatkan seorang calon penumpang sebelum melakukan perjalanan supaya tidak terjadi perpindahan koridor yang salah. Teknologi informasi memungkinkan representasi informasi pencarian koridor yang tepat ketika penumpang akan melakukan sebuah perjalanan, terlebih dengan hadirnya graph database. Graph Database memungkinkan representasi BRT yang baik karena sifat graph yang secara standar telah menunjukkan node dan relationship. Artikel ini mengkaji penerapan graph database untuk data pada BRT. Selain itu, artikel ini mendesain sebuah algoritma pencarian koridor BRT. Harapannya, algoritma tersebut dapat digunakan untuk membangun aplikasi yang memanfaatkan data pada graph database. Hasil kajian menunjukkan bahwa graph database dapat diterapkan untuk data BRT dan algoritma yang dibangun dapat digunakan untuk menyajikan informasi rute sekaligus menyampaikan informasi perpindahan koridor.

1. PENDAHULUAN

Moda transportasi publik sangat diperlukan untuk menyelesaikan berbagai permasalahan transportasi seperti kemacetan trafik. Optimalisasi penggunaan moda transportasi publik adalah salah satu alternatif solusi dari persoalan tersebut [1]. Dari sisi calon penumpang, calon penumpang harus mampu memutuskan moda transportasi apa yang akan digunakan. Artinya, perlu ketersediaan informasi mengenai moda transportasi dan rute yang akan dilalui untuk membantu memudahkan calon penumpang dalam menentukan moda transportasi yang akan dipilihnya.

Beberapa kota di Indonesia memiliki mode transportasi publik yang dinamakan Bus Rapid Transit (BRT). Beberapa contoh yang dapat ditemui adalah BRT di Kota Semarang, Surakarta, dan Bandung. BRT kemudian menjadi moda transportasi alternatif yang terus dikembangkan. Sebagai contoh, di Kota Semarang BRT, dengan nama TransSemarang, berkembang dari tahun ke tahun. Perkembangan tersebut terutama dari sisi jumlah rute atau koridor yang dimiliki BRT TransSemarang. Pada awal tahun 2014, BRT TransSemarang hanya menjalankan dua koridor / rute perjalanan. Pada tahun 2016, sudah ada empat koridor yang dijalankan oleh BRT TransSemarang.

Karakteristik dari perjalanan menggunakan BRT adalah bahwa bus BRT hanya berhenti pada *shelter*/halte tertentu, dimana rangkaian halte-halte dalam satu perjalanan bus BRT disebut dengan koridor. Pada sistem BRT, biasanya memiliki lebih dari satu koridor dimana satu koridor BRT dengan koridor yang lain bukanlah koridor yang terpisah, melainkan saling terhubung. Artinya, dalam melakukan perjalanan, penumpang BRT boleh jadi melakukan perpindahan koridor melalui halte yang dilaluinya. Informasi tersebut perlu didapatkan seorang calon penumpang sebelum melakukan perjalanan supaya tidak terjadi perpindahan koridor yang salah. Teknologi informasi diharapkan dapat membantu calon penumpang untuk dapat memperoleh informasi rute yang tepat dalam perjalanan menggunakan BRT.

Representasi data yang tepat diperlukan untuk memperoleh informasi perjalanan menggunakan BRT. Telah dikemukakan sebelumnnya, perjalanan BRT memiliki jalur tertentu dimana setiap jalur melewati halte-halte tertentu dan masing-masing halte terhubung dalam setiap koridor. Hubungan halte ke halte yang lain seperti halnya menghubungkan titik (*dot*) ke titik yang lain

https://doi.org/10.25077/ TEKNOSI.v3i3.2017.313-319

menggunakan jalur (*edge*). Dengan demikian, model data graph lebih tepat untuk memodelkan koriodor pada BRT.

Penyimpanan dengan model graph dimungkinkan saat ini dengan hadirnya *graph database*. Graph Database memungkinkan representasi BRT yang baik karena sifat graph yang secara standar telah menunjukkan *node* dan *relationship,* representatif untuk memodelkan koridor BRT. Artikel ini mengkaji penerapan *graph database* dalam memodelkan BRT lebih khusus untuk model perjalanan BRT. Artikel ini juga mengusulkan algoritma yang dapat digunakan untuk mencari koridor BRT. Diharapkan, hasil dari artikel ini dapat digunakan untuk mengembangkan aplikasi untuk perjalanan BRT.

2. TINJAUAN PUSTAKA

2.1. Bus Rapid Transit (BRT)

BRT merupakan sebuah sistem bus dengan transit yang memberikan fasilitas yang nyaman dan harga yang terjangkau. BRT memiliki jalur tersendiri dengan karakteristik operasi yang baik dengan dukungan pemasaran dan pelayanan kepada pelanggan [2]. BRT tumbuh sebagai suatu pilihan effektif bagi masyarakat perkotaan dibandingkan dengan sistem bus konvensional. Hal ini karena berbagai karakteristik yang telah disebutkan sebelumnya.

Bus Rapid Transit merupakan salah satu bentuk angkutan yang berorientasi pelanggan dan mengkombinasikan halte, kendaraan, perencanaan, dan elemen-elemen sistem transportasi ke dalam sebuah sistem yang terpadu dan memiliki satu identitas unik. *Bus Rapid Transit* mempunyai sejumlah keluwesan dibandingkan dengan mode transit yang lain, diantaranya ialah yang berhubungan dengan rute layanan. Rute BRT dapat disesuaikan dengan kebutuhan pengguna, kebijakan pemerintah, dan kondisi dinamis yang lain.

2.2. Graph Database

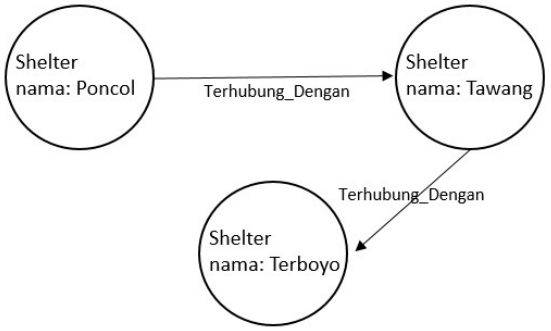
Keterhubungan antar data adalah satu karakter spesifik dari *graph database*. Tanpa keterhubungan antar data, *graph database* menjadi tidak optimal. Pada *graph database*, data dimodelkan dalam nodes dan hubungan antar data menggunakan edges. Hal tersebut berbeda dengan basis data relasional yang menghubungkan data menggunakan kunci. Setiap *node* pada *graph database*, secara internal, memiliki penghubung ke node yang lain ataupun ke sekelompok graph lain atau sub graph [3].

Graph database memiliki sistem manajemen basis data yang hampir sama dengan basis data relasional. Manajemen basis data graf memiliki tata cara dalam *Create*, *Read*, *Update* dan *Delete* (CRUD) yang menampakkan mode data *graph*. Beberapa *graph database* pun mendukung transactional processing (OLTP).

Pada data relasional, operasi *join* akan menurunkan kinerja, terlebih untuk data dengan volume yang besar. Namun, dibandingkan dengan data relasional, kinerja *graph* cenderung tetap, dan relatif konstan, walaupun terjadi pertumbuhan data yang besar. Hal tersebut disebabkan karena pada umumnya query pada graph terbatas pada suatu *segment* atau bagian dari graph serta kemampuan graph untuk merambat dari satu node ke node yang lain dengan mudah. Waktu eksekusi untuk setiap query adalah hanya proporsional ke ukuran dari *segment* graph yang terkait dengan query dan tidak dengan ukuran totalnya.

Model data graph relatif fleksibel karena penambahan *node* dan *relationship* baru dapat dilakukan ke dalam model graph, bahkan penambahan dengan node yang berbeda sama sekali dengan sebelumnya. Penambahan *node* dan *relationship* baru, tidak akan mengganggu fungsi query dan fungsi aplikasi.

Ilustrasi *graph database* ditunjukkan pada Gambar 1. Terdapat 3 node *shelter* yang memiliki properti/atribut nama. Atribut tersebut menunjukkan nama *shelter*. Masing-masing *shelter* saling terhubung dengan hubungan terhubung_dengan.



Gambar 1. Ilustrasi Graph Database

Beberapa produk *graph database* ada seperti Neo4J, Infinite Graph, Sparksee, TITAN, InfoGrid dan lain sebagainya (NoSQL). Masing-masing produk tersebut diimplementasikan menggunakan bahasa pemrograman yang berbeda dan masing-masing memiliki mekanisme dan *query syntax* yang berbeda pula.

Beberapa penelitian telah dilakukan terkait dengan *graph database*. Pada umumnya, penelitian yang telah dilakukan mengenai pemrosesan query, misalnya penelitian mengenai pemrosesan query yang efisien [4] dan query untuk struktur dan data pada *graph database* [5]. Selain query, kajian pemodelan *graph database* telah dilakukan [6] dan [7]. Sedangkan kajian untuk transportasi telah dilakukan [8] yang berfokus pada pemodelan graph untuk transportasi dengan berbagai moda dan [9] yang berfokus pada pemodelan yang menggunakan graph untuk BRT.

Penelitian mengenai algoritma untuk rute transportasi telah dilakukan oleh beberapa peneliti seperti yan, Liang, & Ling-xiang (2006), Zhang, Jigang, & Duan (2010), dan Olczyk & Galuszka(2014). Olczyk & Galuszka (2014) menggunakan 2800 *stops*, dan 370 *lines* untuk representasi rute transportasi. Graph digunakan representasi data yang dilakukan dari stops dan lines tersebut menggunakan namun bukan *graph database*. Sedangkan Yan, Liang, & Linxiang (2006) serta Zhang,Jigang, & Duan (2010) mengkaji dan mengembangkan algoritma rute terpendek yang telah ada. Hasil pengkajian dan pengembangan tersebut kemudian dibandingkan dengan algoritma yang telah ada sebelumnya.

Pada penelitian dan kajian yang telah disebutkan, belum ada yang membahas mengenai bagaimana implementasi *graph database*

https://doi.org/10.25077/ TEKNOSI.v3i3.2017.313-319

untuk BRT khususnya pembahasan mengenai perpindahan koridor. Artikel ini mengkaji penerapan *graph database* dan mengusulkan algoritma untuk mencari rute Bus Rapid Transit, yang dapat menangani perpindahan koridor yang dapat terjadi ketika menggunakan Bus Rapid Transit.

3. METODOLOGI

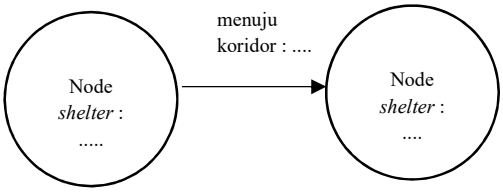
Pada tahap awal, akan dikembangkan model data dalam *graph database*, yang merepresentasikan rute BRT. Berbagai kemungkinan rute perjalanan BRT dibentuk sehingga dapat menggambarkan rute BRT sesungguhnya. Model data kemudian diimplementasikan menggunakan *graph database* Neo4j versi 3.0.4, yang dipasang pada komputer lokal (non-cluster).

Tahap berikutnya adalah menyusun algoritma yang dapat digunakan untuk mencari rute BRT dengan memanfaatkan data BRT yang telah diimplementasi pada Neo4j. Dengan demikian, algoritma untuk menentukan rute BRT dapat disusun. Untuk keperluan pengujian, algoritma diterapkan menggunakan bahasa pemrograman Java sehingga dapat diketahui benar/tidaknya alur dari algoritma yang disusun tersebut.

4. HASIL DAN PEMBAHASAN

4.1. Struktur Data Graph Database

Data dari bus rapid transit yang akan dikaji memiliki struktur seperti yang ditunjukkan pada Gambar 2. Pada Gambar 2, node (gambar bulat) menunjukkan *shelter* BRT dan edge (gambar anak panah) menunjukkan keterhubungan antar *shelter* (rute) serta menunjukkan koridor dari rute tersebut. Rute BRT memiliki koridor-koridor perjalanan tertentu, dimana setiap koridor terdiri dari sekumpulan *node* maupun *edge.*



Gambar 2. Struktur data pada *graph database*

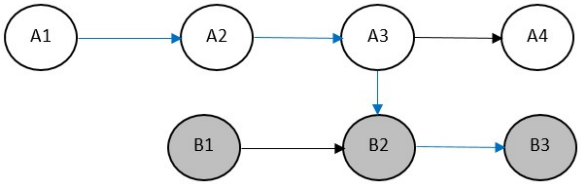
4.2. Algoritma

Algoritma merupakan langkah-langkah yang harus ditempuh untuk menyelesaikan permasalahan tertentu. Dalam penelitian ini, algoritma yang akan dibangun digunakan untuk menyelesaikan permasalahan dalam pencarian rute *bus rapid transit* (BRT). Data terkait dengan rute *bus rapid transit* tersimpan pada *graph database*, mengikuti struktur dari *graph database*, yaitu terdapat *node* dan *edge* dimana masing-masing memiliki atribut yang menjelaskan makna dari *node* maupun *edge*.

Pencarian rute BRT dalam penelitian ini didefinisikan sebagai pencarian rute perjalanan dari satu *shelter*/node ke node yang lain baik dalam satu koridor maupun dalam koridor yang berbeda. Terdapat beberapa kekmungkinan dalam pencarian rute tersebut yaitu :

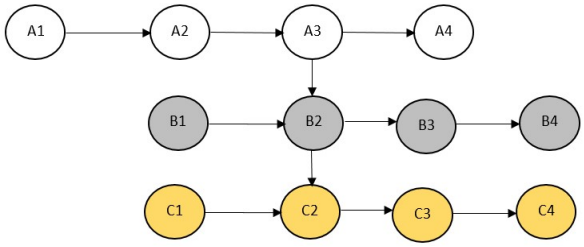
a. Tidak ditemukan *shelter* / node yang dimaksud, yaitu ketika tidak ada koridor yang menuju ke *shelter* yang dimaksud dari *shelter* asal pencarian.
b. Terdapat rute tanpa perpindahan koridor, yaitu ketika *shelter* yang dituju dapat dilalui hanya melalui satu koridor yang sama dengan *shelter* asal. Ilustrasi ditunjukkan pada Gambar 3, dimana *shelter* asal adalah A1 dan *shelter* tujuan adalah A4, dimana perpindahan dari A1 sampai dengan A4 ada dalam satu koridor.
c. Terdapat rute dengan perpindahan koridor, yaitu ketika *shelter* yang dituju dengan *shelter* tujuan melalui koridor yang berbeda. Ilustrasi dari pencarian rute ini ditunjukkan pada Gambar 4, dimana *shelter* asal adalah A1 dan *shelter* tujuan adalah B3.

Gambar 3. Rute tanpa perpindahan koridor



Gambar 4. Rute dengan perpindahan koridor.

Ketiga kemungkinan hasil pencarian yang sudah disebutkan, digunakan sebagai acuan untuk membentuk algoritma. Untuk menguji algoritma yang telah dibentuk, disusun data set yang mewakili semua kemungkinan hasil pencarian. Gambar 5 merupakan model data yang digunakan sebagai data untuk menunjukkan algoritma yang akan ditentukan kemudian.



Gambar 5. Data untuk menguji algoritma

Model data yang digunakan mewakili tiga koridor yaitu A, B dan C, dimana, property informasi koridor pada edge tidak ditunjukkan. Pada gambar 5, koridor A terhubung melalui koridor B melalui *shelter* B2, sedangkan koridor B terhubung melalui koridor C melalui *shelter* B2. Artinya, B2 memiliki 2 percabangan, yaitu ke koridornya sendiri dan ke koridor lain. Model tersebut telah mewakili kemungkinan-kemungkinkan rute BRT, dan akan digunakan sebagai model untuk *graph database*.

Graph database memiliki kemampuan untuk merambat (*traverse*) antar node dengan metode *breadth first* ataupun *depth first*. Hasil dari traversing tersebut diperoleh melalui permintaan (*query*) tertentu, dan *graph database management system* yang akan menjalankan salah satu dari kedua metode traversing. Node-node yang terlibat dalam rute dari node asal sampai dengan node tujuan, dikembalikan sebagai hasil dari permintaan tersebut.

Algoritma-1 merupakan algoritma untuk mencari rute untuk data pada *graph database*. Algoritma-1 digunakan untuk menyajikan informasi node-node yang dilewati pada sebuah rute perjalanan. Algoritma-1 pun menyajikan informasi perpindahan koridor yang diperlukan untuk sebuah rute perjalanan.

Algoritma-1 menerima masukan berupa *shelter* asal penumpang dan *shelter* tujuannya. Data tersebut kemudian akan dimasukkan ke dalam query *graph database* untuk dieksekusi (baris 1-3). Apabila rute dari *shelter* asal dan tujuan tersambung, maka algoritma akan melacak masing-masing *shelter* yang dilalui oleh rute tersebut. Untuk setiap *shelter* yang dilewati, algoritma-1 akan mendeteksi nama *shelter* beserta koridornya. Ketika koridor sebuah *shelter* sama dengan *shelter* sebelumnya, yang dideteksi dari *edge*/relationship yang menuju koridor tersebut, maka berarti kedua *shelter* masih dalam koridor yang sama. Sedangkan apabila edge yang menuju sebuah *shelter* berbeda dengan sebelumnya, maka *shelter* tersebut berbeda koridor dan berarti penumpang harus turun ke *shelter* dan berganti koridor.

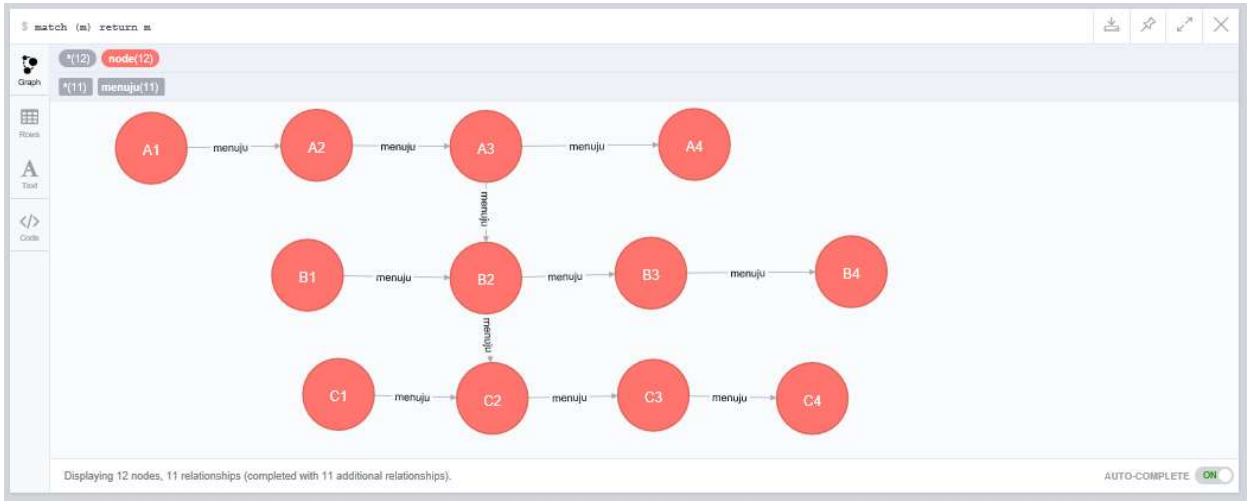
Algoritma-1 : algoritma pencarian rute perjalanan bus rapid transit

```
1:read shelter_asal, shelter_tujuan
2:query_rute ← graph database query sesuai shelter_asal dan shelter_tujuan
3:rute ← eksekusi(query_rute)
4:if rute != nil :
5: koridor_sebelum ← nil
6: koridor_sekarang ← nil
7: while(rute belum berakhir) :
8:   shelter ← rute.get(shelter)
9:   koridor ← rute.get(koridor)
10:  koridor_sekarang ← koridor
11:  if koridor_sekarang!=koridor_sebelum&&koridor_sebelum != nil :
12:    koridor_sebelum ← koridor_sekarang
13:    output informasi pindah koridor
```

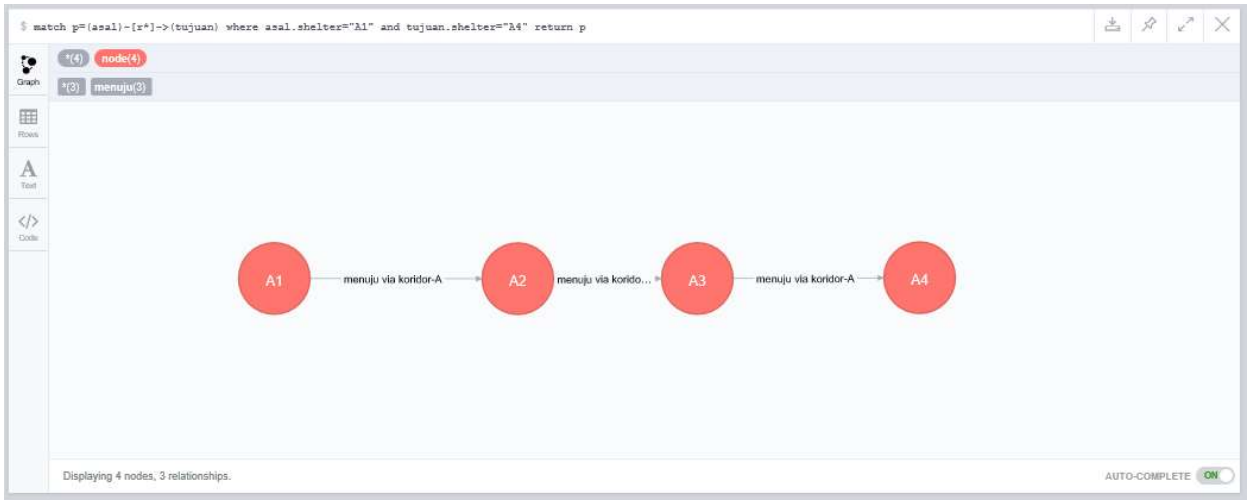
Kode-1 : Penerapan algoritma-1 pada pemrograman Java

```
public void cariRute(String asal, String tujuan) {
        Driver driver = GraphDatabase.driver("bolt://localhost",
                    AuthTokens.basic("neo4j", "panji"));
        Session session = driver.session();
        String cypher = "match p=(asal)-[r*]->(tujuan) where asal.shelter=\"" +
                    asal + "\" and tujuan.shelter=\"" + tujuan + "\" return p";
        StatementResult result = session.run(cypher);
        if (result != null) {
            List<Record> list = result.list();
            Record rec = list.get(0);
            Value val = rec.get("p");
            Path path = val.asPath();
            Iterator nodes = path.nodes().iterator();
            Iterator relationship = path.relationships().iterator();
            String koridorSblm = "";
            String koridorSkrng = "";
            while (nodes.hasNext()) {
                Node node = (Node) nodes.next();
                String shelter = node.get("shelter").asString();
                print(shelter);
                if (relationship.hasNext()) {
                    Relationship rel = (Relationship) relationship.next();
                    koridorSkrng = rel.get("koridor").asString();
                    if (!koridorSkrng.equalsIgnoreCase(koridorSblm)
                                        && !koridorSblm.equals("")) {
                        print("Turun di "+shelter+" pindah " + koridorSkrng);
                    }
                    print(koridorSkrng);
                    koridorSblm = koridorSkrng;
                }
            }
            session.close();
        }
}
```

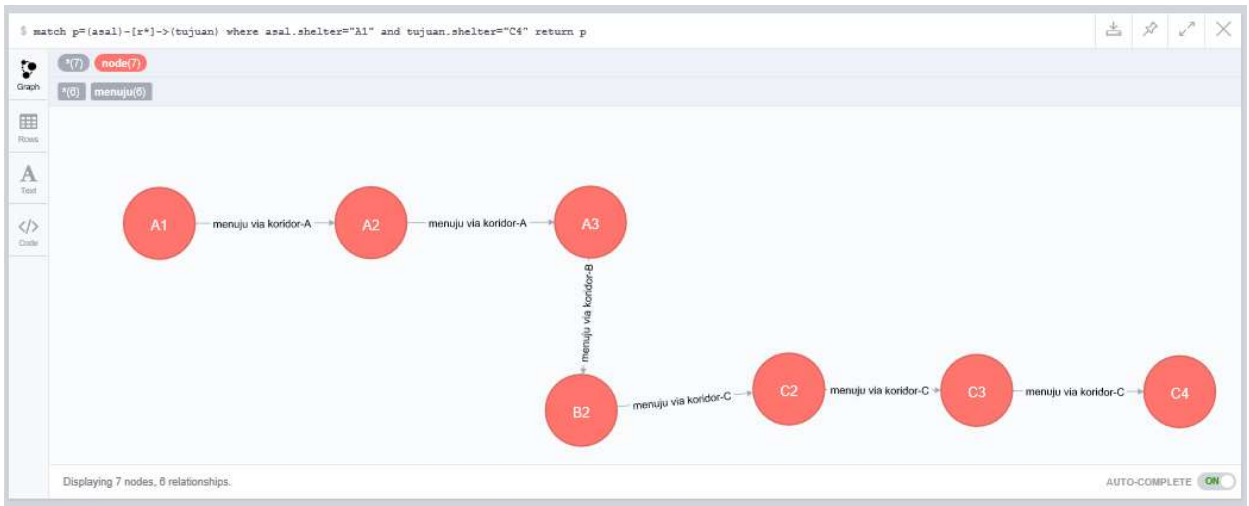
 https://doi.org/10.25077/ TEKNOSI.v3i3.2017.313-319



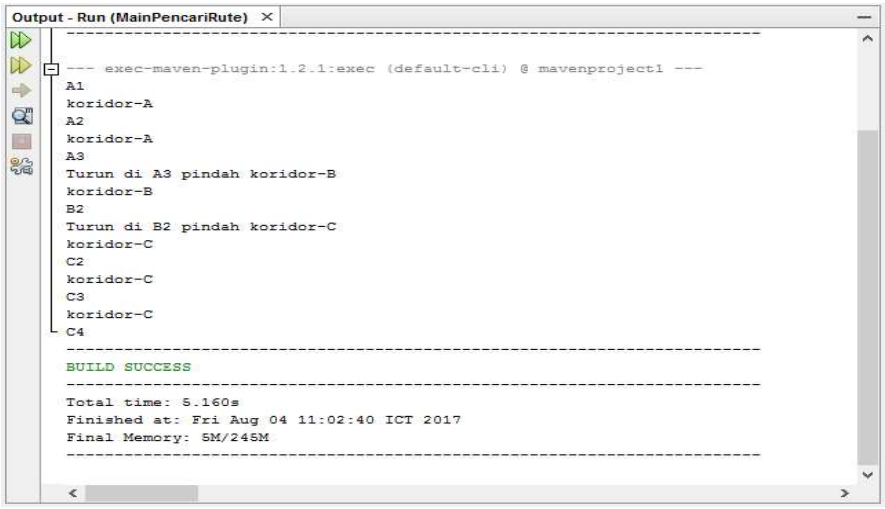
Gambar 6. Hasil implementasi pada Neo4j



Gambar 7. Hasil pengujian tanpa perpindahan koridor.



Gambar 8. Hasil eksekusi Query-2.



Gambar 9. Hasil eksekusi kode-1 untuk pencarian rute A1-C4.

Setelah menyusun algoritma, tahapan selanjutnya adalah menguji algoritma tersebut dengan cara mengimplementasikan model data dan membuat implementasi algoritma pada program. Neo4j digunakan sebagai *graph database management system* yang mengimplementasikan model data. Sedangkan pemrograman Java dipilih untuk mengimplementasikan algoritma tersebut

4.3. Implementasi Algoritma

Model data pada Gambar 7 diimplementasikan menggunakan Neo4j dengan Cypher query. Query 1 merupakan cypher query untuk membentuk data pada *graph database* sesuai Gambar 5. Query tersebut digunakan untuk membentuk berbagai node serta relationship. Hasil eksekusi dari query 1 adalah Gambar 6, yaitu data yang telah diimplementasikan pada Neo4j.

Query 1 : implementasi model data
create (a1:node {shelter:"A1"}), (a2:node {shelter:"A2"}), (a3:node {shelter:"A3"}),(a4:node {shelter:"A4"}), (b1:node {shelter:"B1"}), (b2:node {shelter:"B2"}), (b3:node {shelter:"B3"}), (b4:node {shelter:"B4"}), (c1:node {shelter:"C1"}),(c2:node {shelter:"C2"}), (c3:node {shelter:"C3"}), (c4:node {shelter:"C4"}) create (a1)-[:menuju{koridor:"koridor-A"}]->(a2) create (a2)-[:menuju{koridor:"koridor-A"}]->(a3) create (a3)-[:menuju{koridor:"koridor-A"}]->(a4) create (a3)-[:menuju{koridor:"koridor-B"}]->(b2) create (b1)-[:menuju{koridor:"koridor-B"}]->(b2) create (b2)-[:menuju{koridor:"koridor-B"}]->(b3) create (b3)-[:menuju{koridor:"koridor-B"}]->(b4) create (b2)-[:menuju{koridor:"koridor-C"}]->(c2) create (c1)-[:menuju{koridor:"koridor-C"}]->(c2) create (c2)-[:menuju{koridor:"koridor-C"}]->(c3) create (c3)-[:menuju{koridor:"koridor-C"}]->(c4)

Query / permintaan untuk rute pada *graph database* dilakukan oleh Query-2 tanpa memandang metode *traversing* yang digunakan, karena telah dilakukan oleh *graph database management system*. Pada prinsipnya, Query -2 memiliki masukan node asal (*shelter* A1) dan node tujuan dengan berbagai relationship yang memungkinkan (r*) dari asal hingga tujuan (*shelter* C4). Jika node dan relationship yang diminta ada, maka node-node dan relationship yang sesuai akan dikembalikan sebagai luaran.

Pengujian pertama kali adalah pengujian rute tanpa perpindahan koridor, yaitu dari *shelter* A1 ke *shelter* A4. Query 2 menunjukkan permintaan pada Neo4j untuk menampilkan rute ari A1 ke A4. Query tersebut berhasil dan hasilnya ditunjukkan pada Gambar 7.

Query-2 : permintaan rute tanpa perpindahan koridor.
match p=(asal)-[r*]->(tujuan) where asal.shelter="A1" and tujuan.shelter="A4" return p

Query-3 : permintaan rute dengan perpindahan koridor.
match p=(asal)-[r*]->(tujuan) where asal.shelter="A1" and tujuan.shelter="C4" return p

Hasil eksekusi Query-2 (gambar 8) menunjukkan bahwa rute perjalanan dari A1 sampai dengan C4, walaupun berbeda koridor, dapat dikerjakan. Hasil tersebut sekaligus menunjukkan bahwa, query tersebut dapat memilih jalur ke C4 walaupun terdapat percabangan ke koridor yang lain pada *shelter* B2.

Algoritma-1 diterapkan menggunakan pemrograman Java. Kode-1 menunjukkan penerapan algoritma-1 pada pemrograman Java, dimana yang ditunjukkan merupakan *method*/fungsi utama penerapan Algoritma-1. Fungsi tersebut menerima masukan berupa *shelter* awal dan tujuan bertipe data String. Fungsi diawali dengan mendefinisikan driver untuk mengakses Neo4j dan Cypher query seperti Query-2 dan Query-3, kemudian query tersebut akan dieksekusi oleh statement program di bawahnya. Apabila ada hasil query, maka program akan melakukan perulangan sepanjang node yang ditemukan dan akan mengevaluasi masing-masing node dan koridornya. Apabila koridor berbeda, fungsi tersebut akan menampilkan informasi untuk turun pada *shelter* tertentu dan berganti koridor.

Hasil eksekusi program Kode-1 untuk pencarian rute dari *shelter* A1 ke *shelter* C4 ditunjukkan pada Gambar 9, sebagai luaran dari program Java. Hasil tersebut menunjukkan bahwa algoritma-1 dapat diterapkan untuk pencarian rute dengan perpindahan koridor sekalipun. Program mampu menampilkan informasi

 https://doi.org/10.25077/ TEKNOSI.v3i3.2017.313-319

koridor dan mampu memberikan informasi di mana penumpang harus turun dan pindah koridor.

5. KESIMPULAN

Artikel ini menghasilkan model data BRT untuk *graph database* yang mendukung adanya perjalanan dengan perpindahan koridor. Dengan kemampuan *traversing*-nya, *graph database* mampu memberikan informasi mengenai *shelter* dan koridor yang dilewati oleh sebuah rute perjalanan BRT. Algoritma yang disusun dapat dengan mudah menampilkan informasi *shelter* dan koridor yang dilewati oleh rute perjalanan BRT. Algoritma tersebut diharapkan dapat bermanfaat untuk membuat program yang memerlukan pencarian rute pada perjalanan menggunakan BRT.

DAFTAR PUSTAKA

[1] R. Ferdiansyah, "KEMUNGKINAN PERALIHAN PENGGUNAAN MODA ANGKUTAN PRIBADI KE MODA ANGKUTAN UMUM PERJALANAN DEPOK – JAKARTA," *J. Perenc. Wil. dan Kota*, vol. 20, no. 3, pp. 183–198, 2009.

[2] L. Wright and W. Hook, *Bus Rapid Transit Planning Guide*, 3rd ed. Institute for Transportation & Development Policy, 2007.

[3] J. Celko, "NoSQL and Transaction Processing," in *Joe Celko's Complete Guide to NoSQL*, Elsevier, 2014, pp. 1–14.

[4] J. Cheng, Y. Ke, and W. Ng, "Efficient query processing on *graph database*s," *ACM Trans. Database Syst.*, vol. 34, no. 1, pp. 1–48, Apr. 2009.

[5] L. Libkin, W. Martens, and D. Vrgoč, "Querying Graphs with Data," *J. ACM*, vol. 63, no. 2, pp. 1–53, Mar. 2016.

[6] R. De Virgilio, A. Maccioni, and R. Torlone, "Model-Driven Design of Graph Databases," 2014, pp. 172–185.

[7] R. Angles and C. Gutierrez, "Survey of *graph database* models," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 1–39, Feb. 2008.

[8] J. Booth, P. Sistla, O. Wolfson, and I. F. Cruz, "A data model for trip planning in multimodal transportation systems," in *Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology - EDBT '09*, 2009, p. 994.

[9] P. W. Wirawan, D. E. Riyanto, and K. Khadijah, "PEMODELAN GRAPH DATABASE UNTUK MODA TRANSPORTASI BUS RAPID TRANSIT," *J. Inform.*, vol. 10, no. 2, Jul. 2016.

[10] Z. yan, Z. Liang, and Z. Ling-xiang, "Public Transportation Guidance Model and Algorithm," in *2006 International Conference on Communication Technology*, 2006, pp. 1–4.

[11] Z. Zhang, W. Jigang, and X. Duan, "Practical algorithm for shortest path on transportation network," in *2010 International Conference on Computer and Information Application*, 2010, pp. 48–51.

[12] A. Olczyk and A. Galuszka, "Finding routes in a public transport network. A case study," in *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2014, pp. 800–803.

BIODATA PENULIS

Panji Wisnu Wirawan

Penulis adalah staff pengajar di Departemen Ilmu Komputer/Informatika Fakultas Sains dan Matematika Universitas Diponegoro. Penulis menyelesaikan sarjana dan magister dari Universitas Gadjah Mada dan saat ini menekuni bidang penelitian dalam teknologi informasi khususnya dalam penerapan NoSQL database untuk berbagai . Penulis saat ini aktif sebagai anggota IEEE Computer Society.

Djalal Er Riyanto

Penulis saat ini aktif sebagai staff pengajar di Departemen Ilmu Komputer/Informatika Fakultas Sains dan Matematika Universitas Diponegoro. Penulis menyelesaikan program sarjana dari Universitas Diponegoro dan master dari Univresitas Indonesia dan saat ini menekuni bidang penelitian dalam sistem informasi dan basis data.