Analisis dan Implementasi Graph Indexing Pada Graph Database Menggunakan Algoritma Closure Tree

Analysis and Implementation of Graph Indexing for Graph Database Using Closure Tree Algorithm

Riche Julianti Wibowo[1], Kemas Rahmat Saleh[2], Adiwijaya[3]

[1.2.3] *Fakultas Informatika, Telkom Engineering School, Telkom University*
Jalan Telekomunikasi No.1, Dayeuh Kolot, Bandung 40257
richewibowo@gmail.com[1], bagindok3m45@gmail.com[2], kang.adiwijaya@gmail.com[3]

Abstrak

Graph menjadi populer untuk pemodelan data terstruktur seperti senyawa kimia. Teknologi basisdata seperti *relational database* kurang efektif menangani data yang terstruktur. Maka, *graph database* dibutuhkan. Untuk menangani pencarian informasi terhadap data yang terstruktur pada *graph database* digunakan metode *graph indexing* agar lebih cepat dan efisien.

Dari beberapa metode *graph indexing* yang ada, *Closure tree* (C-tree) adalah metode *graph indexing* yang paling tepat digunakan karena menggunakan konsep *graph closure* dimana setiap simpul merangkum informasi dari simpul-simpul keturunannya dan membangun *tree* sebagai *index*. Pada tugas akhir ini diharapkan mampu menerapkan algoritma C-tree pada *graph indexing* dengan *dataset* bertipe molekul serta menganalisis *answer set*, *tree construction*, dan *query time* yang dihasilkan.

Kata kunci : *graph*, *graph database*, *graph indexing*, *graph closure*, C-tree.

Abstract

Graphs have become popular for modeling structured data such as chemical coumpounds. Database technologies such as relational databases are less effective to handling structured data. then, graph database is needed.To handle the information search against structured data in databases used graph indexing methods for more quickly and efficiently.

Of the several methods of indexing the existing graph, tree Closure (C-tree) is a graph indexing method most appropriate to use as it uses the concept of closure in which each node graph summarizes information from nodes offspring, and build tree as a index. In this final project is expected to implement the C-tree algorithm on the graph indexing with the dataset type of molecule and analyze answer set, tree construction, and query time.

Keywords: graph, graph databases, graph indexing, graph closure, C-tree.

1. Pendahuluan

Perkembangan zaman telah berpengaruh pada pertumbuhan data yang makin meningkat dan bervariasi. Untuk menanganinya, perusahaan-perusahaan menggunakan *relational database* karena mudah dan mendukung banyak tipe data. Namun ada suatu tipe data yang berbeda dari tipe data pada umunya, yaitu tipe data graf yang memiliki simpul dan sisi. Dalam tipe data graf, *relational database* kurang efektif untuk menanganinya karena tipe data nya yang saling terkait.

Graf dan pemrosesan *graph query* pun telah berkembang pesat dikarenakan penggunaannya yang terbilang cukup sering. Secara umum, graf telah digunakan untuk memodelkan senyawa kimia[5], struktur protein[9], dan *social networks*[3]. Dari semua kasus tersebut, penggunaan *graph database* dapat mendukung memodelkan suatu data yang terstruktur. Pengimplementasian *graph database* untuk menangani tipe data yang terstruktur dinilai lebih efektif daripada menggunakan *relational database*[14].

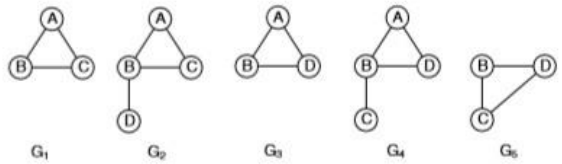
Dalam melakukan pencarian *query* pada *graph database* dibutuhkan suatu metode yang berbeda dari *relational database*. Maka, untuk menangani pencarian *query* terhadap data yang terstruktur pada *graph database* digunakan metode *graph indexing* agar lebih cepat. Beberapa metode *graph indexing* yaitu GIndex[1], C-tree[4], Graphgrep[2], dan Gstring[7]. Dari sekian banyak metode *graph indexing* yang ada, *Closure tree* (C-tree) dianggap paling tepat. Hal ini disebabkan tahapan-tahapannya cukup sederhana yaitu menggunakan konsep *graph closure* untuk menggabungkan beberapa simpul dan/atau sisi sehingga membentuk suatu graf baru yang didasarkan pada pemetaannya, melakukan *search phase* untuk menghasilkan *candidate answer set*, dan melakukan *verification phase* untuk mencari *answer set*.

Pada tugas akhir ini, *dataset* yang digunakan adalah bertipe molekul. Karakteristik dari tipe data molekul yaitu tidak berarah dan tidak berbobot dianggap cocok dalam penggunaan tahapan *graph closure*. Karena tahapan *graph closure* tidak memerhatikan arah dan bobot dari *dataset*. Tujuan melakukan *graph closure* pada *dataset* adalah untuk membangun *index* yang berbentuk *tree* sehingga memudahkan dalam melakukan pemrosesan dan pencarian *query*. Karena index yang dibangun adalah berbetuk *tree*, maka tahapan dari *graph closure* akan membuat *query time* yang dibutuhkan menjadi lebih cepat.

2. Landasan Teori

2.1 Graph Database

Graph database adalah teknologi basis data yang seperti pada graf terdiri atas kumpulan sisi (*edges*) dan simpul (*vertices*) yang dapat diakses secara langsung [6]. *Graph database*[11] didasarkan pada teori graf. Secara umum, simpul mewakili entitas, properti mewakili atribut, dan sisi mewakili hubungan.



Gambar 2-1: Contoh graf database

2.2 Closure Tree

Closure Tree merupakan salah satu teknik *graph indexing* yang berbasis *tree*. Setiap simpul berisi informasi dari simpul-simpul keturunannya yang direpresentasikan dengan *graph closure* [12]. Algoritma C-tree terbagi menjadi tiga fase yaitu *index construction*, *search phase*, dan *verification phase*.

2.2.1 Teknik Graph Closure

Index construction pada C-tree dibuat dengan menggunakan teknik *graph closure*. *Graph closure* menggabungkan kedua graf menjadi suatu graf baru sesuai dengan atribut dari setiap simpul dan sisi yang sama. Graf yang baru ini menangkap informasi struktural dari tiap graf sebelumnya. Keterkaitan antar dua graf bergantung kepada *graph mapping* yang mendasarinya. Sehingga dapat disimpulkan bahwa *graph closure* dari dua graf G1 dan G2 akan menghasilkan suatu graf baru yang dilambangkan C1 = *closure*(G1, G2)[4] dimana simpul dari C1 merupakan gabungan dari simpul G1 dan G2 yang memiliki informasi yang sama dan sisi dari C1 merupakan gabungan dari sisi G1 dan G2 yang memiliki informasi yang sama pula.

Namun sebelum melakukan teknik *graph closure*, dari *dataset* yang ada akan dibandingkan antara satu graf dengan graf lainnya untuk diketahui nilai *similarity* yang dimilikinya. Hal ini bertujuan untuk memudahkan dalam melakukan teknik *graph closure*.

2.2.2 Search Phase

Mengunjungi semua C-tree dari atas (*root*) hingga ke bawah (*leaf*). Pada *search phase* ini akan menghasilkan *candidate answer set*.

2.2.3 Verification Phase

Candidate answer set yang telah dihasilkan oleh *search phase* di verifikasi pada fase ini. Hasil dari fase ini yaitu *answer set*.

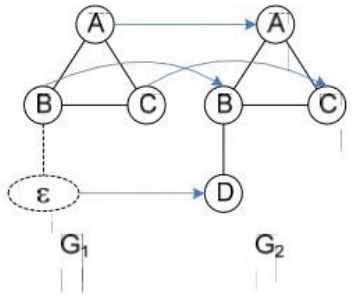
2.3 Teorema Closure Tree

Dalam tugas akhir ini dijabarkan tentang teorema dari C-tree, yaitu:

2.3.1 Teorema 1

Tingkat *similarity* antara dua graf G1 dan G2 berdasarkan pemetaannya adalah jumlah dari kecocokan antara simpul dan sisi di G1terhadap G2.

$$sim(G1, G2) = \sum_{v \in V1} \delta(v) + \sum_{e \in E1} \delta(e) \quad (2.1)$$



Gambar 2-2: Similarity G1 terhadap G2

$\delta(v)$ adalah jumlah dari kecocokan simpul antar graf. Jika dilihat dari gambar 2-2 antara G1 dan G2, maka jumlah kecocokan simpul adalah 3 yaitu :

- simpul A pada G1 yang cocok dengan simpul A pada G2
- simpul B pada G1 yang cocok dengan simpul B pada G2
- simpul C pada G1 yang cocok dengan simpul C pada G2

Maka dari itu $\sum \delta(v) = 3$

Kemudian $\delta(e)$ adalah jumlah dari kecocokan sisi antar graf. Jika dilihat dari gambar 2-2 antara G1 dan G2, maka jumlah kecocokan sisi adalah 3 yaitu :

- sisi AB pada G1 yang cocok dengan sisi AB pada G2
- sisi BC pada G1 yang cocok dengan sisi BC pada G2
- sisi AC pada G1 yang cocok dengan sisi AC pada G2

Maka dari itu $\sum \delta(e) = 3$

Berdasarkan dari rumus (2.1) dan gambar 2-10, maka sim(G1,G2) = 6.

2.3.2 Teorema 2

Closure dari seperangkat simpul adalah gabungan dari nilai-nilai atribut simpul. Demikian juga dengan sisi, *closure* dari seperangkat sisi adalah gabungan dari nilai-nilai atribut sisi.

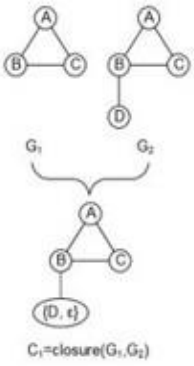
2.3.3 Teorema 3

Closure dari dua graf G1 dan G2 berdasarkan pemetaannya adalah G(V,E) dimana V adalah himpunan *vertex closure* yang sesuai dan E adalah hinpunan *edge closure* yang sesuai. Dilambangkan dengan *closure*(G1, G2).

2.3.4 Teorema 4

Volume dari *graph closure* C adalah hasil dari jumlah *vertex closure* VC dan *edge closure* EC.

$$volume(C) = \prod_{v \in VC} |v| \times \prod_{e \in EC} |e| \quad (2.2)$$



Gambar 2-3: C1=closure(G1,G2)

Vertex closure adalah simpul yang merupakan hasil dari *graph closure* dari dua graf sebelumnya. Begitupun dengan *edge closure* adalah sisi yang merupakan hasil dari *graph closure* dari dua graf sebelumnya. Sebagai contoh jika ada dua graf G1 dan G2 seperti pada gambar 2-3, simpul A pada C1 merupakan hasil *graph closure* dari simpul A pada G1 dan simpul B pada G2. Maka simpul A merupakan *vertex closure*. Dan sisi AB pada C1 merupakan hasil

graph closure dari sisi AB pada G1 dan sisi AB pada G2. Maka sisi AB merupakan *edge closure*.

|□□ adalah hasil dari jumlah *vertex closure*. Jika dilihat dari gambar 2-11maka |□□ = |4| = 4

- simpul A pada C1 merupakan hasil *graph closure* dari simpul A di G1 dan G2
- simpul B pada C1 merupakan hasil *graph closure* dari simpul B di G1 dan G2
- simpul C pada C1 merupakan hasil *graph closure* dari simpul C di G1 dan G2
  simpul {D,∈} pada C1 merupakan hasil *graph closure* dari simpul { } di G1 dan D di G2

Dari persamaan $\prod_{i \in I} \mathbb{X}_{□□} = \{f : I \rightarrow \cup_{i \in I} \mathbb{X}_{□□} \mid (\forall □□)(□□) \in \mathbb{X}_{□□}$ (2.3)

Maka $\prod_{v_{□□} \in VC} |4| = \cup_{v_{□□} \in VC} 4$

|□□ adalah hasil dari jumlah *edge closure*. Jika dilihat dari gambar 2-11maka |□□ = |4| = 4

- sisi AB pada C1 merupakan hasil *graph closure* dari sisi AB di G1 dan G2
- sisi BC pada C1 merupakan hasil *graph closure* dari sisi BC di G1 dan G2
- sisi AC pada C1 merupakan hasil *graph closure* dari sisi AC di G1 dan G2
  sisi BD pada C1 merupakan hasil *graph closure* dari sisi { } di G1 dan D di G2

Dari persamaan (2.3) maka $\prod_{e_{□□} \in EC} |4| = \cup_{e_{□□} \in EC} 4$

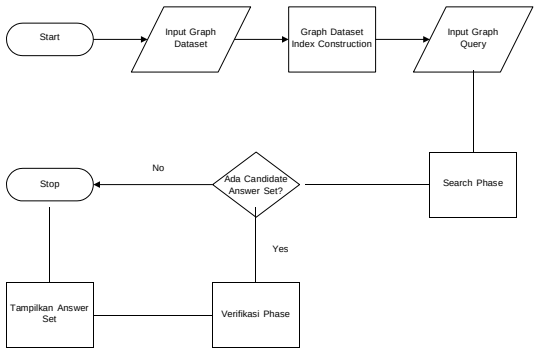
Berdasarkan dari rumus (2.2) pada teorema 4 dan dapat dilihat pada gambar 2-3, maka

$\prod_{v_{□□} \in V□□} |4| \times \prod_{e_{□□} \in EC} |4| = \cup_{v_{□□} \in VC} 4 \text{ X } \cup_{e_{□□} \in EC} 4 = 4$

Jadi *volume* (C1) = 4

3. Analisis Perancangan dan Implementasi

Berikut ini adalah gambaran umum sistem yang akan dibangun pada Tugas Akhir ini



Gambar 3-1: Gambaran sistem

Secara umum sistem akan menerima dua *file*, yang pertama berupa *dataset graph database* dengan format yang telah ditentukan dan yang kedua berupa *query graph* yaitu *file* berisi graf yang digunakan untuk mendapatkan *answer set*.

Pertama *user* akan memasukkan *dataset*. Sistem lalu memproses *dataset* untuk membangun index dengan cara membuat *graph closure* dari *dataset* setelah itu sistem akan membuat *tree* dari *graph closure* yang telah dibuat. Masukkan yang kedua adalah *query graph*. Tahap selanjutnya yaitu *search phase*. Pada tahap *search phase*, sistem akan menghasilkan *candidate answer set*. Jika *candidate answer set* tidak ditemukan maka sistem akan langsung selesai untuk menyatakan bahwa *graph query* tidak ditemukan di *dataset*. Tapi jika *candidate answer set* ditemukan, maka sistem akan melanjutkan ke tahap selanjutnya yaitu melakukan *verification phase*. Pada tahap ini, sistem memverifikasi antara *graph query* dengan setiap graf dari *candidate answer set* untuk mendapatkan *answer set.*

4. Pengujian dan Analisis

4.1 Skenario Pengujian

Skenario pengujian dilakukan untuk melakukan langkah-langkah dalam melakukan pengujian dengan *dataset* berjumlah 750 graf bertipe molekul. Jumlah skenario pengujian dilakukan sebanyak empat. Berikut skenario yang akan dilakukan :

1. Pengaruh ukuran nilai *threshold* terhadap jumlah *candidate answer set*
2. Pengaruh ukuran nilai *threshold* terhadap jumlah *answer set*
3. Pengaruh ukuran nilai *threshold* terhadap waktu seluruh proses *query*
4. Pengaruh ukuran nilai *threshold* terhadap akurasi dari algoritma C-tree

4.1.1 Pengaruh Ukuran Nilai *Threshold* terhadap Jumlah *Candidate Answer Set*

Pengujian dilakukan untuk menghitung jumlah *candidate answer set* dari proses *search phase* menggunakan 3 nilai *threshold* dari skala 0 sampai 1 yaitu 0.8, 0.85, dan 0.9 untuk setiap jumlah dataset yang telah ditentukan yaitu data *reference* 150, 300, 450, 600, dan 750 serta data *testing* 375. Dan juga menggunakan 2 jenis *query* yaitu *query* yang memiliki 1 siklik dan yang tidak memiliki siklik.

4.1.2 Pengaruh Ukuran Nilai *Threshold* terhadap Jumlah *Answer Set*

Pengujian dilakukan untuk menghitung jumlah *answer set* dari proses *verification phase* menggunakan 3 nilai *threshold* dari skala 0 sampai 1 yaitu 0.8, 0.85, dan 0.9 untuk setiap jumlah dataset yang telah ditentukan yaitu data *reference* 150, 300, 450, 600, dan 750 serta data *testing* 375. Dan juga menggunakan 2 jenis *query* yaitu *query* yang memiliki 1 siklik dan yang tidak memiliki siklik.

4.1.3 Pengaruh Ukuran Nilai *Threshold* terhadap Waktu Seluruh Proses *Query*

Pengujian dilakukan untuk mengukur waktu seluruh proses *query* menggunakan 3 nilai *threshold* dari skala 0 sampai 1 yaitu 0.8, 0.85, dan 0.9 untuk setiap jumlah dataset yang telah ditentukan yaitu data *reference* 150, 300, 450, 600, dan 750 serta data *testing* 375. Waktu seluruh proses *query* sudah termasuk waktu hasil *search phase* dan *verification phase*. Dan juga menggunakan 2 jenis *query* yaitu *query* yang memiliki 1 siklik dan yang tidak memiliki siklik.

4.1.4 Pengaruh Ukuran Nilai *Threshold* terhadap Akurasi Dari Penggunaan Algoritma C-tree

Pengujian dilakukan untuk mengukur waktu seluruh proses *query* menggunakan 3 nilai *threshold* dari skala 0 sampai 1 yaitu 0.8, 0.85, dan 0.9 untuk setiap jumlah dataset yang telah ditentukan yaitu data *reference* 150, 300, 450, 600, dan 750 serta data *testing* 375. Nilai akurasi adalah nilai dengan skala 0 sampai 1 yang berarti semakin tinggi nilai akurasi, maka algoritma C-tree yang digunakan adalah makin bagus. Dan nilai akurasi didapatkan dengan membagi jumlah *answer set* dan jumlah *candidate answer set*. Dan juga menggunakan 2 jenis *query* yaitu *query* yang memiliki 1 siklik dan yang tidak memiliki siklik.

4.2 Analisis Hasil Pengujian

Bagian ini memaparkan hasil implementasi yang dilakukan berdasarkan skenario yang sudah di laksanakan

4.2.1 Analisis Hasil Pengaruh Ukuran Nilai *Threshold* terhadap Jumlah *Candidate Answer Set*

Pengujian dilakukan dengan menggunakan 4 *query* untuk masing-masing pengujian, yaitu:

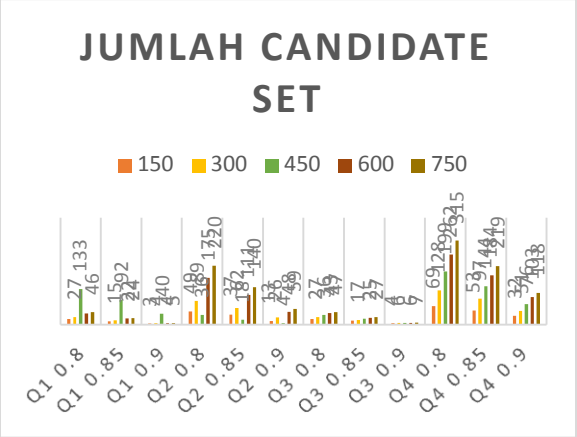
Tabel 4-1: Tabel Query

No	Query	Jumlah Simpul	Siklik
1	BrC(Br)=Cc1ccccc1	10	Ya
2	O=C(OCC)CNNCC	10	Tidak
3	BrCCc1ccc(cc1)CCBr	12	Ya
4	OC(C)(C#CC(O)(C)CC)CC	12	Tidak

Berikut merupakan hasil pengujian yang telah didapatkan

Tabel 4-2: Tabel analisis jumlah candidate answer set

TH	Jumlah Candidate Answer Set				
	150	300	450	600	750
Q1 0.8	20	27	133	42	46
Q1 0.85	12	15	92	22	24
Q1 0.9	3	4	40	4	5
Q2 0.8	49	89	36	175	220
Q2 0.85	37	62	18	111	140
Q2 0.9	13	26	4	48	59
Q3 0.8	20	27	36	43	47
Q3 0.85	14	17	21	25	27
Q3 0.9	4	6	6	6	7
Q4 0.8	69	128	199	262	315
Q4 0.85	53	97	144	184	219
Q4 0.9	32	51	76	103	118



Gambar 4-1: Diagram jumlah candidate answer set

Analisis yang dapat diambil berdasarkan data tabel 4-2 bahwa semakin besar nilai *threshold* maka semakin kecil jumlah *candidate answer set* yang didapatkan untuk masing-masing jumlah *dataset*. Dan semakin besar nilai *threshold* maka semakin besar jumlah *candidate answer set* yang didapatkan untuk jumlah *dataset* yang semakin membesar juga. Hal ini dikarenakan semakin besar nilai *threshold* maka akan memperkecil kemungkinan jumlah *candidate set* yang didapatkan. Artinya, yang diambil hanya graf yang memiliki tingkat kemiripan tinggi dengan *query*. Karena jika nilai *threshold* semakin besar, maka hanya nilai *similarity* tertinggi saja yang diambil untuk menjadi *candidate answer set*. Semakin tinggi nilai *similarity* antar graf maka akan semakin tinggi kemiripan antar graf.

Kesimpulan lain yang didapatkan yaitu bahwa nilai *threshold* dapat mempengaruhi jumah *dataset* jika

dataset yang digunakan semakin besar. Hal ini dikarenakan semakin besar jumlah *dataset*, maka akan semakin tinggi kemungkinan untuk menemukan lebih banyak *candidate answer set*.

Dari data tabel 4-2 juga didapatkan terdapat perbedaan antara keempat *query*. Secara keseluruhan, *query* 2 dan *query* 4 menghasilkan jumlah *candidate answer set* yang lebih besar dibandingkan *query* 1. Karena *query* 2 dan *query* 4 merupakan *query* yang sederhana maka sebagian besar *dataset* memiliki nilai *similarity* yang tinggi terhadap *query* 2 sehingga memiliki *candidate answer set* yang berjumlah lebih besar.

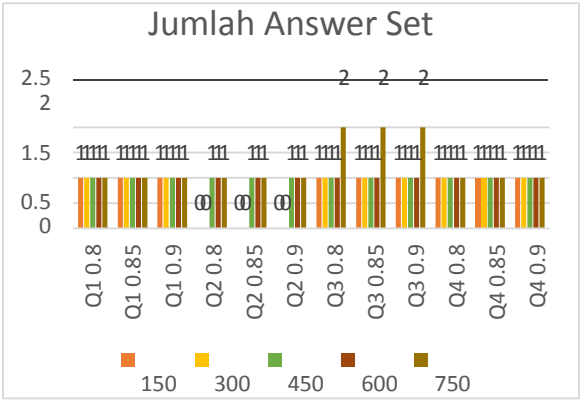
Jika diambil contoh dari *dataset* 750 pada *query* 2, maka perbedaan jumlah *candidate answer set* adalah rata-rata sebesar 55.801%. Hal ini menunjukkan bahwa ukuran *threshold* sangat mempengaruhi jumlah *candidate answer set* hingga melebihi 50%.

4.2.2 Analisis Hasil Pengaruh Ukuran Nilai *Threshold* terhadap Jumlah *Answer Set*

Answer set didapatkan dari *candidate anwer set* yang telah di verifikasi. Berikut merupakan hasil pengujian yang telah didapatkan

Tabel 4-3: Tabel analisis jumlah answer set

DR	Jumlah Answer Set				
	150	300	450	600	750
Q1 0.8	1	1	1	1	1
Q1 0.85	1	1	1	1	1
Q1 0.9	1	1	1	1	1
Q2 0.8	0	0	1	1	1
Q2 0.85	0	0	1	1	1
Q2 0.9	0	0	1	1	1
Q3 0.8	1	1	1	1	2
Q3 0.85	1	1	1	1	2
Q3 0.9	1	1	1	1	2
Q4 0.8	1	1	1	1	1
Q4 0.85	1	1	1	1	1
Q4 0.9	1	1	1	1	1



Gambar 4-2: Diagram jumlah answer set

Berdasarkan hasil pengujian pada tabel 4-3 dapat disimpulkan bahwa nilai *threshold* tidak mempengaruhi hasil *answer set*. Hal ini dibuktikan pada hampir keseluruhan *dataset* memiliki jumlah *answer set* yang sama walaupun nilai *threshold* berbeda. Hal ini dikarenakan *answer set* hanya

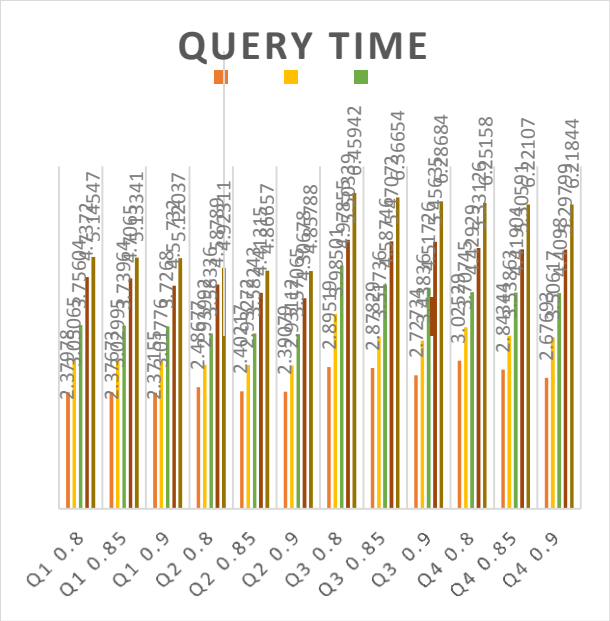
dipengaruhi oleh *candidate answer set* dan *query* itu sendiri yaitu dengan membandingkan antara *candidate answer set* yang telah didapatkan sebelumnya pada pengujian 4.2.1 dengan *query* yang diinginkan untuk menghasilkan nilai *similarity* yang memiliki nilai 1. Itu sebabnya, hasil *answer set* pada tabel 4-3 adalah bernilai 1 atau 0. Nilai 0 yang ditujukkan pada *query* 2 di data 150 dan 300 hasil *answer set* menunjukan kosong, ini dikarenakan *query* yang diminta tidak ditemukan didalam *dataset.*

4.2.3 Analisis Hasil Pengaruh Ukuran Nilai *Threshold* terhadap Waktu Seluruh Proses *Query*

Berikut merupakan hasil yang didapat berdasarkan pengujian:

Tabel 4-4: Tabel waktu seluruh proses query

DR	Query Time (detik)				
	150	300	450	600	750
Q1 0.8	2.37978	3.05065	3.75604	4.7372	5.14547
Q1 0.85	2.37673	3.02995	3.73964	4.7065	5.13341
Q1 0.9	2.37155	3.01776	3.7268	4.55732	5.12037
Q2 0.8	2.48677	2.93992	3.58336	4.58789	4.92311
Q2 0.85	2.40217	2.93272	3.58243	4.41315	4.86657
Q2 0.9	2.39079	2.93112	3.57065	4.30678	4.85788
Q3 0.8	2.89519	3.98501	4.97855	5.50339	6.45942
Q3 0.85	2.87829	3.51736	4.58746	5.47072	6.36654
Q3 0.9	2.72734	3.43836	4.51726	5.45635	6.28684
Q4 0.8	3.02529	3.70745	4.42929	5.33126	6.25158
Q4 0.85	2.84344	3.53863	4.41904	5.30591	6.22107
Q4 0.9	2.67693	3.50617	4.4098	5.29799	6.21844



Gambar 4-3: Diagram waktu seluruh proses query

Berdasarkan hasil pengujian pada tabel 4-4 dapat disimpulkan bahwa nilai *threshold* dapat mempengaruhi waktu seluruh proses *query*. Proses *query* yang dimaksud adalah waktu dalam melakukan perhitungan nilai *similarity*, pencarian *candidate answer set*, pencarian *answer set*, dan mengukur akurasi. Dalam satu *dataset*, semakin besar nilai *threshold* maka semakin kecil *query time* yang dibutuhkan. Hal ini dapat dilihat dari banyaknya *candidate answer set* yang dihasilkan sebelumnya pada pengujian 4.2.1 karena dari data pengujian *candidate answer set* yang dihasilkan semakin banyak maka akan membutuhkan *query time* yang lebih lama.

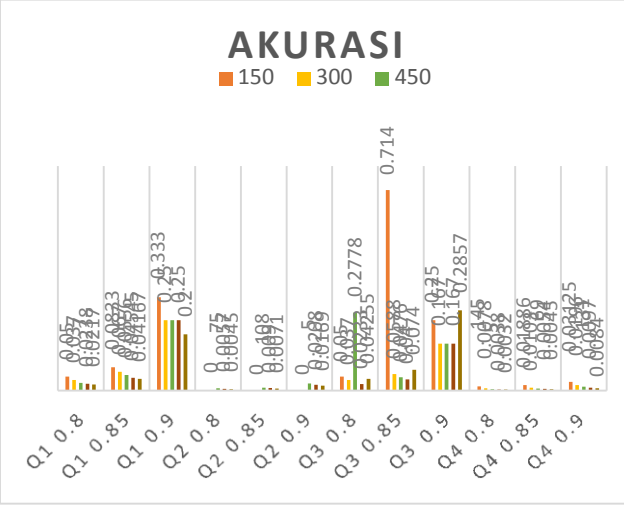
Hal ini juga sama dengan jumlah *dataset*, dapat dilihat pada diagram 4-3 semakin besar jumlah *dataset* dan semakin besar nilai *threshold* maka semakin membutuhkan *query time* yang lebih lama pula. Hal ini dikarenakan, semakin besar jumlah *dataset* maka akan membutuhkan waktu lebih lama pada perhitungan nilai *similarity* karena harus dilakukan pada tiap *dataset* yang ada.

4.2.4 Analisis Hasil Pengaruh Ukuran Nilai *Threshold* terhadap Akurasi Dari Penggunaan Algoritma C-tree

Berikut merupakan hasil yang didapat berdasarkan pengujian:

Tabel 4-5: Tabel akurasi

DR	Akurasi				
	150	300	450	600	750
Q1 0.8	0.05	0.037	0.0278	0.0238	0.0217
Q1 0.85	0.0833	0.067	0.0556	0.04545	0.04167
Q1 0.9	0.333	0.25	0.25	0.25	0.2
Q2 0.8	0	0	0.0075	0.0057	0.0045
Q2 0.85	0	0	0.0108	0.009	0.0071
Q2 0.9	0	0	0.025	0.0208	0.0169
Q3 0.8	0.05	0.037	0.2778	0.0233	0.04255
Q3 0.85	0.714	0.0588	0.0476	0.04	0.074
Q3 0.9	0.25	0.167	0.167	0.167	0.2857
Q4 0.8	0.0145	0.0078	0.005	0.0038	0.0032
Q4 0.85	0.01886	0.0103	0.0069	0.0054	0.0045
Q4 0.9	0.03125	0.0196	0.0131	0.0097	0.0084



Gambar 4-4 : Diagram akurasi

Pada hasil pengujian pada tabel 4-5 dapat disimpulkan bahwa nilai *threshold* dapat mempengaruhi nilai akurasi dari algoritma C-tree. Nilai *threshold* dapat mempengaruhi jumlah *candidate answer set*, dan jumlah *candidate answer set* merupakan salah satu faktor yang mempengaruhi nilai akurasi. Dari data yang didapatkan, dalam satu *dataset* semakin besar nilai *threshold* maka semakin besar pula nilai akurasi.

Nilai akurasi didapatkan dengan membagi antara jumlah *answer set* dengan jumlah *candidate answer set*. Sebagai contoh pada *threshold* 0.8 di *dataset* 150 dengan *query* 1, *candidate answer set* yang dihasilkan adalah 20, dan *answer set* yang dihasilkan adalah 1. Maka akurasi yang didapatkan adalah 1/20 = 0.05. Jika diukur dengan skala 0 sampai 1 maka tingkat akurasi algoritma C-tree kurang bagus. Hal ini karena dari 20 *candidate answer set* yang didapat hanya bisa menghasilkan 1 *answer set* yang didapat.

Dari diagram 4-4 juga dapat dilihat dengan jelas perbedaan pada masing-masing *threshold* terutama pada nilai *threshold* 0.9 yang memiliki jumlah jauh lebih tinggi dibanding dengan nilai *threshold*. Hal tersebut menunjukkan bahwa semakin tinggi nilai *threshold* maka hasil yang didapat akan lebih akurat dalam penggunaan algoritma C-tree.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan pengujian yang telah dilakukan pada tugas akhir ini, maka dapat diperoleh kesimpulan sebagai berikut:

1. Semakin besar nilai *threshold* maka semakin kecil jumlah *candidate answer set* yang didapatkan untuk masing-masing jumlah *dataset*.
2. Nilai *threshold* tidak mempengaruhi hasil *answer set*. Karena *answer set* dipengaruhi oleh *candidate answer set* dan *query* yaitu dengan membandingkan antara *candidate answer set* yang telah didapatkan sebelumnya dengan *query* untuk menghasilkan nilai *similarity* yang memiliki nilai 1.
3. Nilai *threshold* dapat mempengaruhi waktu seluruh proses *query*. Dalam satu *dataset*, semakin besar nilai *threshold* maka semakin kecil *query time* yang dibutuhkan. Hal ini juga sama dengan jumlah *dataset*, semakin besar nilai *threshold* maka semakin membutuhkan *query time* yang lebih lama pula.
4. Nilai *threshold* dapat mempengaruhi nilai akurasi dari algoritma C-tree. Dalam satu *dataset* semakin besar nilai *threshold* maka semakin besar pula nilai akurasi. Dan semakin besar jumlah *dataset* ternyata

membuat nilai akurasi beragam, ada yang bertambah besar nilai akurasi nya dan ada yang semakin mengecil jumlah akurasi nya.

5. Dari pengujian yang telah dilakukan, semakin besar nilai *threshold* yang digunakan maka akan lebih optimal untuk menghasilkan *candidate answer set*, *query time*, dan nilai akurasi
6. Dari pengujian yang didapatkan, dapat disimpulkan bahwa algoritma C-tree memiliki kelebihan yaitu kecepatannya dalam proses *query* dapat dilihat dari hasil uji yang hanya membutuhkan waktu beberapa detik. Namun untuk akurasi dari penggunaan algoritma C-tree dalam pengujian ini kurang begitu bagus .

5.2 Saran

Berdasarkan kesimpulan yang didapatkan dari analisis dan pengujian dari pengerjaan tugas akhir ini, maka berikut ini beberapa saran sebagai pertimbangan untuk penelitian selanjutnya yaitu:

1. *Dataset* pada tugas akhir ini menggunakan molekul yang memiliki sisi tidak berarah dan tidak berbobot. Untuk penelitian selanjutnya bisa menggunakan *dataset* yang struktur graf nya berbeda seperti *social network*, peta, dan lain-lain yang memiliki sisi berbobot dan berarah.
2. Untuk penelitian selanjutnya dapat menggunakan *dataset* dengan ukuran yang lebih besar misalnya hingga mencapai 2000 atau lebih dan menggunakan lebih banyak unsur molekul dalam *dataset*.
3. Untuk penelitian selanjutnya, dapat memberikan solusi untuk tingkat akurasi yang lebih baik.
4. Untuk penelitian selanjutnya dapat menerapkan algoritma lain pada proses *Verification Phase* untuk mencari *answer set*

Daftar Pustaka

[1] Yan, X., P. S. Yu, & J. Han. 2004. *Graph indexing: A frequent structure-based approach*. In SIGMOD.

[2] D. Shasha, J. T. Wang, and R. Giugno. *Algorithmics and applications of tree and graph searching*. In PODS, pages 39–52, 2002.

[3] S. Whiteand P. Smyth. *Algorithms for estimating relative importance in networks*. In Proc. SIGKDD, 2003.

[4] H. He and A. K. Singh. Closure-tree: *An index structure for graph queries*. In ICDE, page 38, 2006.

[5] J. Wilson, Robin, and J. Watkins, John. 1990. *Graphs An Introduction Approach*.

[6] I Robinson, J Webber, E Eifrem. 2013. *Graph databases*.

[7] Jiang, H., Wang, H., Yu, P. S., & Zhou, S. (2007). *GString: A novel approach for efficient search in graph databases*. In Proceedings of the 23rd International Conference on Data Engineering (ICDE) (pp. 566-575).

[8] Dickson, Allen. 2006. *Introduction to Graph Theory*.

[9] H. Berman et al. *The protein data bank*. Nucleic Acids Research, (28):235–242, 2000.

[10] Ian R.,J. W.(2013). *Graph Databases*. O'Reilly Media.

[11] Sherif Sakr and Ghazi Al-Naymat. 2012. *An overview of graph indexing and querying techniques*.

[12] Jain R., Iyengar S., Arora A. 2013. *Overview of Popular Graph Databases*.

[13] Chad Vicknair et al. 2010. *A Comparison of a Graph Database and a Relational*. in *ACM Southeast Regional Conference*, New York.

[14] Adamchick, V. 2005. *Graph Theory*.

[15] Weininger, D. 1988. *SMILES, a Chemical and Information System. 1. Introduction to Methodology and Encoding Rules*. J. Chem. Inf. Comput. Sci