OPTIMASI QUERY DATABASE MENGGUNAKAN ALGORITMA GENETIK

Manahan Siallagan, Mira Kania Sabariah, Malanita Sontya
Jurusan Teknik Informatika
Universitas Komputer Indonesia
mira_ljuan@yahoo.com

ABSTRAKSI

Algoritma genetik adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Proses algoritma genetik menggabungkan proses seleksi, penggunaan operator crossover (penyilangan) dan mutasi untuk mendapatkan solusi terbaik. Dua metode crossover yang dapat digunakan untuk menyelesaikan masalah optimasi query database ini, yaitu M2S crossover dan CHUNK crossover.

Penggunaan dua metode crossover tersebut akan diuji dan dianalisa hasilnya, untuk mengetahui metode crossover apa yang terbaik yang dapat digunakan untuk menyelesaikan masalah optimasi query database, dengan mencari nilai minimum.

Hasil yang dicapai dengan algoritma genetik dapat mencapai solusi yang optimum. Semakin besar nilai parameter (Pc, Popsize dan Maxgen) maka hasilnya akan semakin akurat. Sebaliknya nilai Pm tidak perlu terlalu besar karena akan membuat hasil akhir kurang akurat. Namun demikian, hasil pengujian selanjutnya dapat saja berbeda karena komponen algoritma genetik berbasis pada fungsi random.

***Kata Kunci** : Algoritma Genetik, M2S Crossover, CHUNK Crossover, Relasi.*

1. Pendahuluan

Query merupakan bagian dari basis data. Untuk itu sistem basis data memerlukan metode terbaik untuk menjawab *query*. Pencarian strategi dalam pemrosesan *query* sangat penting agar pemrosesan dapat dilakukan secara cepat. Pada beberapa *Data Base Management System* (DBMS), pemroses *query* memilih dari sekumpulan strategi dengan berbasiskan heuristik tertentu.

Optimasi merupakan suatu langkah untuk mengoptimalkan waktu menjadi lebih efisien. Ketika sebuah *query* diberikan pada sistem basis data, optimasi penting dilakukan untuk memilih strategi yang efisien untuk mengevaluasi ekspresi relasi yang ditentukan.

System-R adalah sistem percobaan yang dibangun di *San Jose IBM Research Laboratory* untuk membuktikan bahwa relasi sistem basis data dapat menyatukan tampilan yang bagus dengan fungsi yang sempurna, yang biasa digunakan untuk *Access Path Selection, Join Ordering*. Algoritma *System-R* adalah salah satu algoritma yang diusulkan dan telah diuji untuk optimasi *query* sebelumnya.

Langkah awal pemrosesan Algoritma *System-R* adalah temukan cara terbaik *single-relation*, kemudian temukan cara terbaik *two-relation* dengan mempertimbangkan masing-masing rencana *single-relation* (dari langkah pertama) sebagai *outer relation* dan setiap relasi lainnya sebagai *inner relation*, setelah itu temukan cara terbaik *k-relation*, sampai semua relasi telah dihubungkan. Pada tiap-tiap langkah, semua permutasi dicapai dengan cara termurah yang terus di simpan untuk iterasi selanjutnya. Hasil akhirnya adalah suatu perintah sequensial dari *sub-joins*. Algoritma ini tidak mendukung untuk banyak paralelisme.

2. Landasan Teori

2.1 Optimasi Query

Optimasi Query adalah suatu proses untuk menganalisa query untuk menentukan sumber-sumber apa saja yang digunakan oleh query tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah output. Atau bisa juga dikatakan bahwa optimasi query adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi tersebut menjadi lebih efektif. Optimasi query mencakup beberapa teknik seperti transformasi query ke dalam bentuk logika yang sama, memilih jalan akses yang optimal dan mengoptimumkan penyimpanan data. Tujuan dari optimasi query adalah menemukan jalan akses yang termurah untuk meminimumkan total waktu pada saat proses sebuah query. Untuk mencapai tujuan tersebut, maka diperlukan *optimizer* untuk melakukan analisa query dan untuk melakukan pencarian jalan akses.

3. Analisis

3.1 Deskripsi Masalah

Permasalahan yang akan dibahas yaitu menganalisa penggunaan dua metode *crossover* dalam menyelesaikan masalah pengoptimasian *query* database. Optimasi *query* yang akan dilakukan yaitu untuk mengoptimasi *query* yang virtual saja, dimana relasi ***A B C D E F*** diasumsikan sebagai tabel-tabel, dan akan di representasikan kedalam

query tree. Sebagai gambarannya optimasi dilakukan dengan menggabungkan tabel-tabel tersebut dengan mencari *cost model*nya terlebih dahulu. Menggunakan *Left-Deep Strategy*, dengan dua metode *crossover*, yaitu *M2S crossover* dan *Chunk Crossover*.

Penggunaan metode algoritma genetik dalam menyelesaikan masalah optimasi *query* ini, dikarenakan metode algoritma *System-R* yang telah dilakukan untuk masalah ini sebelumnya belum mencapai nilai optimum dan membutuhkan banyak iterasi.

3.2 Penyelesaian Masalah dengan Algoritma Genetika

Metode penyelesaian masalah yang akan digunakan dalam menyelesaikan optimasi *query database* ini adalah dengan menggunakan algoritma genetik. Dalam penyelesaian masalah dengan menggunakan algoritma genetik, akan dilakukan dengan cara mencari hasil optimum dari algoritma genetik yang digunakan pada *Left-Deep Strategy*.

Algoritma genetik merupakan alternatif yang diharapkan sama dengan hasil terbaik. Walaupun sifat pencarian algoritma genetik yang bersifat acak, hasil yang diperoleh tidaklah selalu yang terbaik, namun diharapkan mendekati hasil optimum yang diinginkan. Dalam menyelesaikan permasalahan dengan menggunakan algoritma genetik, ada beberapa hal dasar yang harus diperhatikan, yaitu : representasi kromosom, inisialisasi populasi, fungsi evaluasi, seleksi, jenis operator genetik yang digunakan (*crossover* dan mutasi), dan penentuan parameter. Proses algoritma genetik tersebut, dpat dijelaskan sebagai berikut :

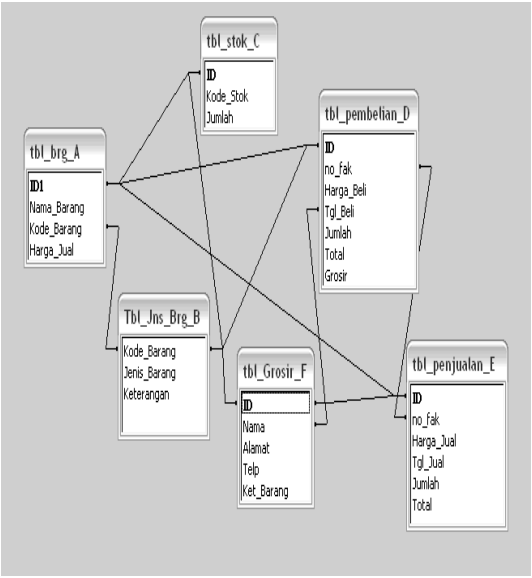
3.2.1 Representasi Kromosom

Elemen utama dalam pengoperasian algoritma genetik adalah kromosom. Pada sebuah kromosom terdiri dari kumpulan gen, yaitu elemen-elemen secara acak yang membentuk sebuah kromosom. Algoritma Genetik bekerja dengan suatu populasi kromosom.masing-masing diantaranya dapat dikodekan ke dalam suatu solusi dari masalah optimasi *query database* ini.

Kromosom dalam algoritma genetik untuk masalah optimasi *query database* ini adalah suatu gen yang terurut, dimana pada masing-masing gen terdiri dari relasi, join dan metode join.

Contoh Permasalahan :

Pada masalah optimasi *query database* ini diberikan sejumlah join dan relasi. Permasalahannya adalah dari 6 relasi tersebut, bagaimana *query* yang akan terbentuk, dimana huruf besar menyatakan relasi. Sebagai contoh, relasi dapat dilihat pada gambar tabel relasi pada gambar 3.1.



Gambar 3.1 Skema Relasi

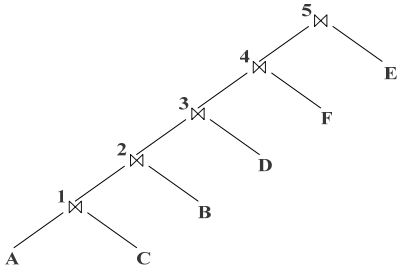
Dari setiap tabel yang ada, untuk membuat *query*, *field-field*nya dapat direlasikan ke tabel mana saja. Dengan kata lain, setiap tabel dapat dijoinkan dengan tabel-tabel lainnya.

Bentuk Join : (A ⋈ C) and (B ⋈ C) and (C ⋈ D) and (D ⋈ E) and (D ⋈ F)

Query diatas dapat digambarkan dengan *query graph*, yang memiliki relasi *query* sebagai *nodes* dan *join* antara relasi sebagai sisi yang tidak langsung. Bentuk representasi dari kromosom diatas dapat dilihat pada gambar 3.2, dengan *J* menggambarkan suatu metode *join*.

Kromosom untuk *Left-Deep Strategy* adalah sebagai berikut :

$$^{J}A\ ^{J}C\ ^{J}B\ ^{J}D\ ^{J}F\ ^{J}E$$



Gambar 3.2 *Left-Deep Tree*

Left-Deep Strategy dilambangkan dengan (*L*). Kromosom pada *Left-Deep Strategy* adalah suatu gen yang terurut, dimana masing-masing gen tersebut terdiri dari relasi dan suatu metode *join* yang akan digunakan.

Pada pengkodean ini, metode *join* dengan inner (kanan) relasi pada masing-masing *join* akan digabungkan, dengan demikian untuk menciptakan ulang *join processing tree*, gabungkan relasi pertama dengan relasi kedua menggunakan metode yang digabungkan dengan relasi kedua.

3.2.2 Inisialisasi Populasi

Pada inisialisasi populasi ini akan dibuat sebuah populasi dari kumpulan kromosom. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Populasi awal dipilih secara acak.

Contoh untuk inisialisasi populasi adalah sebagai berikut :

(Kromosom 1) X = ᵐA ⁿC ᵐB ᵐD ⁿF ⁿE
(Kromosom 2) Y = ᵐB ⁿC ᵐD ᵐF ᵐE ⁿA

3.2.3 Fungsi Evaluasi

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Ada 2 hal yang harus dilakukan dalam melakukan evaluasi kromosom yaitu :

1. Menentukan fungsi tujuan (evaluasi fungsi objektif)

   Fungsi tujuan dari masalah optimasi query database ini adalah meminimalkan biaya strategi.
2. Menentukan fungsi *fitness*

   Fungsi *fitness* diturunkan dari fungsi evaluasi. Fungsi *fitness* yaitu mencari nilai yang paling tinggi, semakin besar nilai *h*, maka semakin kecil nilai *fitness*. Pada *Left-Deep Strategy*, mencari nilai fitness yaitu dengan cara menghitung jarak tiap-tiap gen pada kromosom yang telah terpilih.

Menentukan hasil dari nilai *fitness*, dicari dengan menghitung *cost model*, yaitu :

C = (|R| + |S|) * *hash* + |R| * *move* + |S| * *comp* * F

Dimana : ***R* = inputan 1 (pertama)** dan ***S* = inputan 2 (kedua)**

Dan nilai dari *Hash, Move, Comp* dan *F* adalah sebuah *constant*.

***Hash* = 0.1**
***Move* = 0.01**
***Comp* = 0.1**
***F* = 0.01**

Berikut adalah tabel ukuran dari relasi yang akan digunakan untuk menghitung *cost model*.

Tabel 3.1 Tabel Ukuran Relasi

Tabel	Catalog
A ms	22
B ph	5015
C dv	728
D mx	5015
E mn	1000
F ip	22249
G in	505
H sh	1100
I mv	1100
J pf	1000
K nv	1100
L iz	500

Contoh menghitung *Cost Model* :

Diberikan kromosom (Relasi) ***A C B D F E***

Maka perhitungannya adalah

***Cost*$_{(A\text{-}C)}$ = (22 + 728) * 0.1 + (22) * 0.01 + 728 * 0.1 * 0.01 = 75.984** (input 1)

***Cost*$_{(A\text{-}C)B}$ = (75.984 + 5015) * 0.1 + (75.984) * 0.01 + 5010 * 0.1 * 0.01 = 514.8693** (input 1)

***Cost*$_{(A\text{-}C\text{-}B)D}$ = (514.8693+ 5015) * 0.1 + (514.8693) * 0.01 + 5010 * 0.1 * 0.01 = 563.1506** (input 1)

***Cost*$_{(A\text{-}C\text{-}B\text{-}D)F}$ = (563.1506 + 22249) * 0.1 + (563.1506) * 0.01 + 22249 * 0.1 * 0.01 = 2309.096** (input 1)

***Cost*$_{(A\text{-}C\text{-}B\text{-}D\text{-}F)E}$ = (2309.096 + 1000) * 0.1 + (2309.096) * 0.01 + 1000 * 0.1 * 0.01 = 355.0005** (hasil akhir)

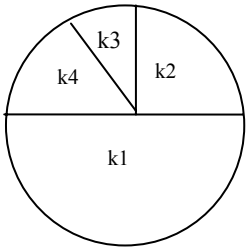
Hasil akhir ini yang nantinya akan dijadikan **nilai *fitness***.

3.2.4 Metode Seleksi yang Digunakan

Metode seleksi yang digunakan pada algoritma ini adalah metode *Roulette Wheel* (piringan rolet). Seleksi ini bertujuan untuk memberikan kesempatan seleksi yang lebih besar bagi anggota populasi yang memiliki *fitness* tinggi untuk melakukan seleksi.

Algoritma dari seleksi roda *roulette wheel* adalah sebagai berikut :

1. Hitung total *fitness* (F) :
   $TotFitness = \sum F_k; k=1,2,\ldots,popsize$
2. Hitung fitness relatif tiap individu :
   $p_k = F_k / TotFitness$
3. Hitung *fitness* komulatif :
   - $q_1 = p_1$
   - $q_k = q_{k-1} + p_k; \quad k = 2,3,\ldots,popsize$
4. Pilih induk yang akan menjadi kandidat untuk di-*crossover* dengan cara :
   - Bangkitkan bilangan *random* r.
   - Jika $q_k \leq r$ dan $q_{k+1} > r$, maka pilih kromosom ke (k+1) sebagai kandidat induk.



Gambar 3.3 Seleksi *Roulette Wheel*

3.2.5 Operator Genetik yang Digunakan

Operator genetik digunakan untuk mengkombinasikan individu-individu yang terdapat di dalam populasi untuk menghasilkan individu pada generasi berikutnya. Operator genetik yang digunakan adalah operator *crossover* dan mutasi. Untuk *crossover*, metode yang digunakan yaitu *permutation crossover,* proses penyilangan dengan permutasi, kromosom-kromosom anak diperoleh dengan cara memilih sub-barisan suatu *tour* dari satu induk dengan tetap menjaga urutan dan posisi sejumlah relasi yang mungkin terhadap induk yang lain.

3.2.5.1 Crossover

Ada 2 metode operator *crossover* yang diterapkan pada operator *crossover* ini, yaitu :

1. Metode *M2S (Modified Two Swap)*

Proses algoritmanya adalah sebagai berikut :

1. Pilih secara acak 2 gen dalam *X*
2. *Replace* dengan gen yang sesuai dari *Y*
3. Pertahankan urutannya tetap seperti itu dari *Y* untuk menciptakan satu *offspring*.

Contoh *crossover* dengan metode *M2S* :
Diberikan 2 induk kromosom, pilih secara acak label ***A*** dan ***D***
Hasil kromosom baru yang diperoleh dengan metode *M2S* adalah

***X* = $^{m}\underline{A}$ ^{n}C ^{m}B $^{m}\underline{D}$ ^{n}F ^{n}E**
***Y* = ^{m}B ^{n}C ^{m}D ^{m}F ^{m}E ^{n}A**
***Offspring X* = $^{m}\underline{D}$ ^{n}C ^{m}B $^{n}\underline{A}$ ^{n}F ^{n}E**

Kemudian, tukar peran antara ***X*** dan ***Y***,menjadi

***Y* = ^{m}B ^{n}C $^{m}\underline{D}$ ^{m}F ^{m}E $^{n}\underline{A}$**
***X* = ^{m}A ^{n}C ^{m}B ^{m}D ^{n}F ^{n}E**
***Offspring Y* = ^{m}B ^{n}C $^{m}\underline{A}$ ^{m}F ^{m}E $^{m}\underline{D}$**

2. Metode *CHUNK*

Proses algoritmanya adalah sebagai berikut :

1. Bangkitkan secara acak *CHUNK* yang dipilih.
2. Kemudian kromosom hasil dari *X* di*copy* kedalam posisi yang sama dalam *offspring*nya.
3. Hapus gen yang sesuai pada *Y*
4. Gunakan gen *Y* untuk memenuhi sisa posisi pada *offspring*.

Contoh *crossover* dengan metode *CHUNK* :
Diberikan 2 induk kromosom, pilih *CHUNK* secara acak urutan ke **[3,5]**, kemudian proses.

***X* = ^{m}A ^{n}C $^{m}\underline{B}$ ^{m}D $^{n}\underline{F}$ ^{n}E**
***Y* = ^{m}B ^{n}C ^{m}D ^{m}F ^{m}E ^{n}A**

Hapus **B** dan **D** yang ada di ***Y,*** dan hasil kromosom baru yang diperoleh dengan metode *CHUNK* adalah

***Offspring X* = ^{n}C ^{m}D $^{m}\underline{B}$ ^{m}E $^{n}\underline{F}$ ^{n}A**

Kemudian, tukar peran antara ***X*** dan ***Y***,menjadi

***Y* = ^{m}B ^{n}C $^{m}\underline{D}$ ^{m}F $^{m}\underline{E}$ ^{n}A**
***X* = ^{m}A ^{n}C ^{m}B ^{m}D ^{n}F ^{n}E**

Hapus **B** dan **D** yang ada di ***X*** dan hasil kromosom baru yang diperoleh dengan metode *CHUNK* adalah

***Offspring Y* = ^{m}A ^{n}C $^{m}\underline{D}$ ^{m}B $^{m}\underline{E}$ ^{n}F**

3.2.5.2 Mutasi

Mutasi yang digunakan dalam penelitian ini adalah mutasi permutasi. Mutasi yang dapat dilakukan yaitu dengan memilih dua nilai gen dari kromosom dan kemudian nilai tersebut saling dipertukarkan. Penukaran untuk setiap nilai dilakukan secara acak dengan masing-masing nilai gen yang menyatakan relasi mempunyai peluang yang sama untuk terpilih sebagai nilai gen penukar. Proses permutasi yang digunakan pada *Left-Deep Strategy* ada dua tipe, yaitu mengubah metode gabungan secara acak dan menukar order dua gen bersebelahan. Pembangkit *Left-Deep strategy* akan memastikan bahwa inisialisasi populasi tidak berisi kartesian produk. Ini dicapai dengan mengedarkan relasi permutasi yang dihasilkan secara acak, hanya menambahkan relasi pada kromosom.

3.2.6 Penentuan Parameter

Parameter genetik berguna dalam pengendalian operator-operator genetik. Beberapa parameter yang digunakan adalah : jumlah relasi, ukuran populasi, maksimum generasi, probabilitas *crossover* (P_c), dan probabilitas mutasi (P_m).Untuk permasalahan ini, parameter yang dapat digunakan adalah : ukuran populasi = 1 – 100 dan maksimum generasi = 1 - 1000. Sedangkan untuk probabilitas *crossover* (P_c) dan probabilitas mutasi (P_m) nilainya disesuaikan pada saat pengujian.

4. Implementasi

4.1 Hasil Pengujian

1. Pengujian dengan (Pc) yang berbeda-beda. (Pm) = 50
   Fitness dan waktu proses yang yang dihasilkan dari tahap pengujian, dapat dilihat pada tabel 4.1 :

ISBN: 1907-5022

Tabel 4.1 Tabel hasil fitness dan waktu komputasi dengan Chunk Crossover

Probabilitas Crossover (Pc)	Banyak Relasi	Fitness	Waktu Proses (ms)	Tree yang dihasilkan
10	6	12.328	406	ACEBDF
20	6	12.328	469	ACEDBF
30	6	12.328	500	ACEBDF
40	6	12.328	422	ACEBDF
50	6	12.328	531	ACEBDF
60	6	12.328	516	ACEBDF
70	6	12.328	516	ACEDBF
80	6	12.328	547	ACEDBF
90	6	12.328	562	ACEBDF
100	6	12.328	563	ACEBDF

Tabel 4.2 Tabel hasil fitness dan waktu komputasi dengan M2S Crossover

Probabilitas Crossover (Pc)	Banyak Relasi	Fitness	Waktu Proses (ms)	Tree yang dihasilkan
10	6	12.328	438	FBDECA
20	6	12.328	453	FDBECA
30	6	12.328	484	FDBECA
40	6	12.328	468	FDBECA
50	6	12.328	453	FDBECA
60	6	12.328	531	FBDECA
70	6	12.328	531	FDBECA
80	6	12.328	516	FDBECA
90	6	12.328	500	FDBECA
100	6	12.328	562	FDBECA

5. Kesimpulan

5.1 Kesimpulan

Kesimpulan yang diperoleh dari optimasi *query database* ini antara lain :

1. Metode terbaik untuk permasalahan optimasi *query database* pada percobaan yang dilakukan adalah metode M2S crossover.
2. Berdasarkan percobaan yang dilakukan, dapat dilihat bahwa dengan algoritma genetik, jika jumlah Pm diperkecil maka hasil optimum dapat dicapai.
3. Jumlah relasi kecil, harga pencarian tidak berubah-ubah.
4. Dari hasil beberapa kali percobaan, bahwa semakin banyak jumlah yang digunakan maka waktu proses yang diperlukan algoritma genetik untuk menyelesaikan optimasi *query* akan semakin lama.

DAFTAR PUSTAKA

1. http://www.cwi.nl/themes/ins1/publications/docs/Pe:DISS:97.pdf. Probabilistic and Transformation Based Query Optimization, 29 November 2006, 9:35 AM.
2. http://citeseer.ist.psu.edu/cache/papers/cs/1665/ftp:zSzzSzftp.cs.wisc.eduzSztech-reportszSzreportszSz91zSztr1004.pdf/bennett91genetic.pdf. A Genetic Algorithm for Database Query Optimization, 13 Oktober 2006. 12:40 AM.
3. http://www.cs.berkeley.edu/~brewer/cs262/SystemR.pdf. History System-R, 4 November 2006, 2:46 AM.
4. http://www-i5.informatik.rwth-aachen.de/i5new/lehre/IDB06/download/Literature/QO.pdf. Left-Deep VS Bushy Trees : An Analysis of Strategy Spaces and Its Implications for Query Optimization, 3 Januari 2007, 10:25 PM.
5. P. Selinger et al. *Access Path Selection in a Relational Data Base System*. In Proc. Of the 1979 ACM-SIGMOD Conference on The Management of Data, pages 23-34, Boston, MA, June 1979.
6. Fatansyah, *Buku Teks Komputer Sistem Basis Data*, Informatika, 2004.
7. Hariyanto, Bambang, MT, Ir., *Sistem Manajemen Basis Data*, Informatika Bandung, 2004.
8. Goldberg, D.E. *Genetic Algorithms in Search Optimization & Machine Learning*. New York: Addision-Wesley Publishing Company. 1989.
9. M. Jarke and J. Koch. Query Optimization in Database Systems. ACM Computing Surveys, 16(2): 111-152, June 1984.