Scientific Journal of Informatics
Vol. 5, No. 1, May 2018
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Genetic Algorithm for Relational Database Optimization in Reducing Query Execution Time

Kukuh Triyuliarno Hidayat[1], Riza Arifudin[2], Alamsyah[3]

[1,2,3] Departement of Computer Science, Universitas Negeri Semarang, Indonesia
Email: [1]kukuhtriyuliarno@students.unnes.ac.id, [2]rizaarifudin@gmail.com, [3]alamsyah@mail.unnes.ac.id

Abstract

The relational database is defined as the database by connecting between tables. Each table has a collection of information. The information is processed in the database by using queries, such as data retrieval, data storage, and data conversion. If the information in the table or data has a large size, then the query process to process the database becomes slow. In this paper, Genetic Algorithm is used to process queries in order to optimize and reduce query execution time. The results obtained are query execution with genetic algorithm optimization to show the best execution time. The genetic algorithm processes the query by changing the structure of the relation and rearranging it. The fitness value generated from the genetic algorithm becomes the best solution. The fitness used is the highest fitness of each experiment results. In this experiment, the database used is MySQL sample database which is named as *employees*. The database has a total of over 3,000,000 rows in 6 tables. Queries are designed by using 5 relations in the form of a left deep tree. The execution time of the query is 8.14247 seconds and the execution time after the optimization of the genetic algorithm is 6.08535 seconds with the fitness value of 0.90509. The time generated after optimization of the genetic algorithm is reduced by 25.3%. It shows that genetic algorithm can reduce query execution time by optimizing query in the part of relation. Therefore, query optimization with genetic algorithm can be an alternative solution and can be used to maximize query performance.

Keywords: Relational Database, Query, Genetic Algorithm, Fitness, Execution Time

1. INTRODUCTION

The relational database is a complex system [1]. Relational model is a model that is often used in processing the database, because there are interrelated information tables. In the relational model, data is formed in relation [2]. Relational database information will be stored into collections in the table [3]. Each table has data storage and will be reused. The processing of relational database uses SQL language, or often known as query. The SQL process is instructed by using Database Management System (DBMS) [3-6].

Structure Query Language (SQL) can be represented in query string [4]. SQL or queries are made from operations performed on table [3]. Query performance can slow down when executing multiple tables of information with large data in physical storage. If that happens, the query requires optimization with query optimizer or can also use optimization algorithm [7]. Troubleshooting needs algorithm. Algorithm is not only used to solve easy problems, but it can also be used on complex issues. Algorithm for optimizations such as genetic algorithm is an easy method for optimization problems [2,

5, 8]. There is a randomly solved method in the algorithm [9, 10, 11] and belongs to the soft computing model [12]. The genetic algorithm was introduced in the 1960s, invented by John Holland and developed by David Goldberg [13, 14]. The genetic algorithm consists of individuals in the population [15]. The concept is to perform a search technique to derive a solution based on the evolutionary process [16-19]. In the process of evolution, individuals who can survive will be obtained, so that the individual who has experienced gene changes for many times, will be able to to adapt. Individual changes occur through breeding. The process of genetic algorithm and the like such as selection, crossover, and mutations to produce the best individuals [20].

Genetic algorithm is used to process database queries in order to obtain optimal queries and best execution time. The process of genetic algorithm is by rearranging the query on the part of the relation. Each relation is calculated for execution time to be processed with genetic algorithm and to create fitness value. Relations are decreased based on the execution time per relation after the genetic algorithm is complete. The database used is a sample database with name of *employees* and the table used is 6 table 5 relation.

2. METHOD

2.1. Experiment

The purpose of this experiment is to generate the best execution time by optimizing the query by using a genetic algorithm. The experiment is useful for experimenting with various possible genetic algorithm parameters such as crossover probabilities and mutation probabilities. This experiment uses 100 experiments to produce the best execution time.

2.2. System Development

Systems for genetic algorithms are created from the PHP programming language. The system is used as a genetic algorithm tool to process queries, the result is a new query by changing the structure of the relation. Flowchart of the system for genetic algorithm can be seen Figure 1.

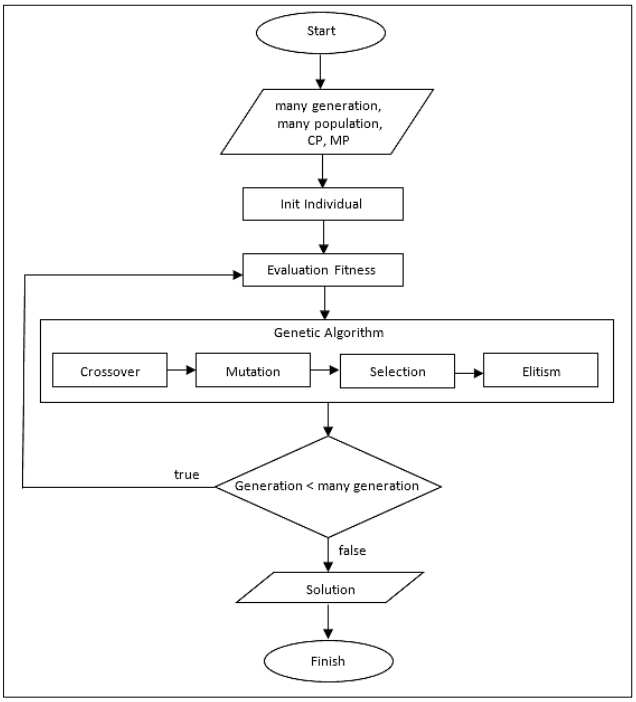

Figure 1. The Flowchart of Genetic Algorithm System

3. RESULTS AND DISCUSSION

3.1. Query and Database for Experiment

In this experiment, the database used is MySQL sample database with the name of employees. The database can be accessed at https://github.com/datacharmer/test_db, but the data contained in the database is selected first to get the *near-to-real* data.

The query used is in the form of left deep tree [3], as follows:

```
SELECT a.first_name, a.last_name, b.salary, c.title, d.dept_no, e.dept_name, f.emp_no
FROM employees a, salaries b, titles c, dept_emp d, departments e, dept_manager f
WHERE a.emp_no = b.emp_no AND
       b.emp_no = c.emp_no AND
       c.emp_no = d.emp_no AND
       d.dept_no = e.dept_no AND
       e.dept_no = f.dept_no AND
       a.first_name = "Ramzi" AND c.title = "Senior Engineer" AND
      b.salary BETWEEN 55025 AND 59700
```

The execution time of the query above was 8.14347 seconds.

3.2. Genetic Algorithm for Database Query

Steps to optimize relational database query with genetic algorithm can be seen in the description below:

1) Population Formation

The initial population produces an initial solution [21]. The population is formed by

arranging chromosomes in the individual, the individual is the problem solution of the genetic algorithm [10], each chromosome is represented as follows:

$T_1R_i(t_it_{i+1})$ and $T_2R_i(t_{i+1}t_i)$

Where as;
T: query execution time per relation
R: query relation
t: table in database (example: table 1, table 2, etc)
i: relation number (1, 2, 3, ..,etc.)

For the T, there are only T_1 and T_2 because there are only two execution time per relation, for example in relation a = b as T_1 and b = a as T_2.
Individual representations uses combinations between T_1 and T_2, but to make individuals, it is randomly assigned by selecting T_1 or T_2 and the position of the relation must be in position.

2) Fitness Evaluation
Fitness shows the advantages of the individual [22]. In fitness evaluation, the fitness value of each individual will be calculated with equation 1.

$$\text{fitness} = \frac{1}{\sum_{j=1}^{n} W_j} .10 \quad (1)$$

Where as;
W: the execution time each chromosome

3) Crossover
The crossover process uses single point crossover or one cut point. The purpose of the crossover is to add variation in the population [20]. The process with chromosomes in individuals is exchanged with other individuals to create new individuals [14]. Crossover-Individuals are selected randomly based on the probability of crossover. Chromosome positions are also randomly selected to be exchanged. Single point crossover process can be seen in Figure 2.

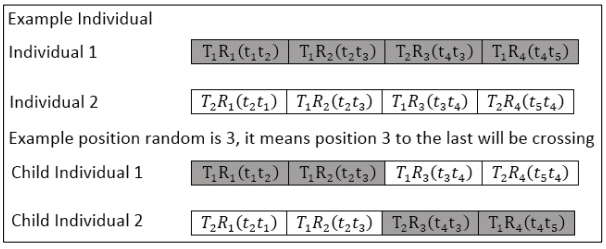

Figure 2. Single Point Crossover

4) Mutation
Mutation is a modification of chromosomes in individuals [23]. Individuals are selected on the basis of mutation probabilities and the chromosomes selected in the mutation are also replaced by the pair of chromosomes. The mutation process can be seen in Figure 3.

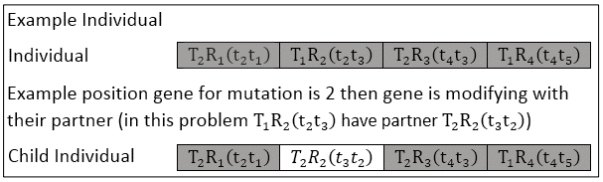

Figure 3. Mutation Process

5) Selection

After crossover, the next mutation process is selection. Selection is almost the same as natural selection with the survival of the fittest principle [16, 17]. The selection used is roulette wheel selection [24]. In the roulette wheel, each individual is selected based on the number that appears in the roulette wheel. Selected individuals are included in the population.

6) Elitism

Elitism is used to store the best individuals with the highest fitness values and possibly will be reused in the next generation.

3.3. Experimental Result

Parameters for experiments with genetic algorithm are as follows:

1. Generation: 100
2. Population size: 120
3. The probability of crossover (PC) and mutation probability (MP) used 10 to 100 with the increment of 10.

The experiment results in this article showed 10 best experiments from 100 experiments. The best results of 10 experiments can be seen in Table 1. The computer specification in this experiment were Windows 7 Ultimate 64 bit, 4 GB RAM, and Intel i3 2370 processor.

Table 1. Best Experiment Result

No.	PC (%)	MP (%)	Fitness	Time (s)
1	70	50	0,90509	6,08535
2	100	50	0,9577	6,32636
3	60	90	0,93724	6,36236
4	10	90	0,94862	6,36436
5	70	70	0,95807	6,39137
6	70	40	0,94853	6,41237
7	20	30	0,93724	6,46037
8	80	70	0,92862	6,46937
9	60	30	0,92819	6,47637
10	60	70	0,94907	6,53637

In Table 1, the best execution time was 6.08535 seconds with the fitness value of 0.90509. The result was the best experiment of 100 experiments. Query results with genetic algorithm optimization can be seen in below:

```
SELECT a.first_name, a.last_name, b.salary, c.title, d.dept_no, e.dept_name, f.emp_no
FROM salaries b, employees a, titles c, dept_emp d, dept_manager f, departments e
WHERE b.emp_no=c.emp_no AND
      c.emp_no=d.emp_no AND
      b.emp_no=a.emp_no AND
      e.dept_no=d.dept_no AND
      f.dept_no=e.dept_no AND
      a.first_name = "Ramzi" AND c.title = "Senior Engineer" AND
      b.salary BETWEEN 55025 AND 59700
```

3.4. Comparison of Query and Execution Time

From the experiment results, the best results with the optimization of genetic algorithm and without genetic algorithm can be compared, it can be seen in Table 2. Comparison of queries can be seen in Table 3.

Table 2. Camparison of Execution Time

GA (s)	No GA (s)	Time Difference (s)	Reduce (%)
6,08535	8,14347	2,05812	25,3

Table 3. Query Comparison

GA	No GA
SELECT a.first_name, a.last_name, b.salary, c.title, d.dept_no, e.dept_name, f.emp_no **FROM** salaries b, employees a, titles c, dept_emp d, dept_manager f, departments e **WHERE** b.emp_no=c.emp_no **AND** c.emp_no=d.emp_no **AND** b.emp_no=a.emp_no **AND** e.dept_no=d.dept_no **AND** f.dept_no=e.dept_no **AND** a.first_name = "Ramzi" **AND** c.title = "Senior Engineer" **AND** b.salary BETWEEN 55025 AND 59700	**SELECT** a.first_name, a.last_name, b.salary, c.title, d.dept_no, e.dept_name, f.emp_no **FROM** employees a, salaries b, titles c, dept_emp d, departments e, dept_manager f **WHERE** a.emp_no = b.emp_no **AND** b.emp_no = c.emp_no **AND** c.emp_no = d.emp_no **AND** d.dept_no = e.dept_no **AND** e.dept_no = f.dept_no **AND** a.first_name = "Ramzi" **AND** c.title = "Senior Engineer" **AND** b.salary BETWEEN 55025 AND 59700

Where as;
GA : optimization of genetic algorithm
No GA : no genetic algorithm

In Table 2, the time decreased 25.3% of the query execution time without the genetic algorithm. Table 3 of the FROM query showed the difference in the order, the structure of the order is reshaped in descending manner, based on the size of the tables in the database. WHERE query also showed the difference that the structure was arranged in descending manner, based on the execution time per relation, but the position of the table field for the relation was processed by genetic algorithm.

3.5. System Implementation

System of genetic algorithm was used as tools, user can fill some parameter of genetic algorithm and query to be processed to be optimized, it can be seen in Figure 4.

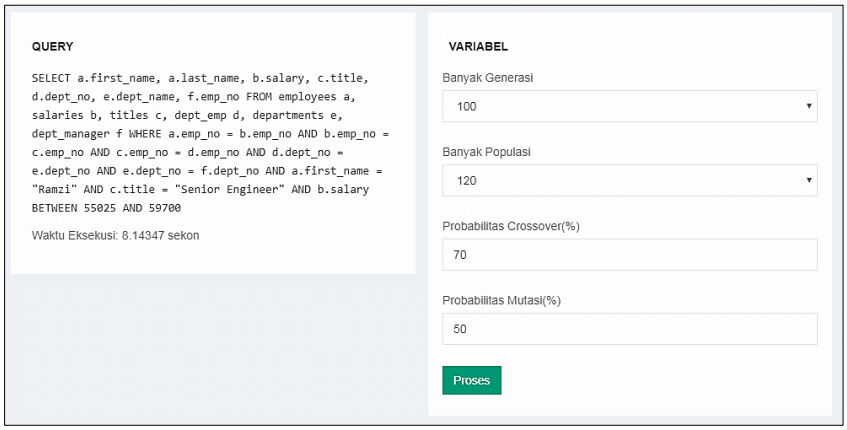

Figure 4. Input of Genetic Algorithm Parameters

After filling the parameters of the genetic algorithm, the system displayed the results of the genetic algorithm process in the form of query and execution time. The results shown were the results of per experiment in the system, if many experiments were performed then the user must do the same thing, but if it used the same query, the user sould only fill the parameters of the genetic algorithm. The results of the system can be seen in Figure 5.

Figure 5. Experiment Result of The System

4. CONCLUSION

The experiment results shows that queries with genetic algorithm optimization can reduce execution time, if compared to queries without genetic algorithm optimatization. Changing the order in the query can shorten the query execution time by scrambling all possible relations. In this study, query execution time without

genetic algorithm optimatization was 8,14347 seconds and query execution time with genetic algorithm optimatization was 6,08535 seconds and the fitness value was 0,90509. The time was reduced by 2,05812 seconds or 25.3% of the queries without genetic algorithm optimization. The use of genetic algorithm can be an alternative solution, but not the best result, since genetic algorithm have a random value in the process.

5. REFERENCES

[1] Bajaj, P L. (2015). A Survey on Query Perfomance Optimization by Index Recomendation. *International Journal of Computer Appplication*. 113, 36-40.

[2] Bennett, K., Ferris, M. C., & Ioannidis, Y. E. (1991). *A genetic algorithm for database query optimization* (pp. 400-407). Computer Sciences Department, University of Wisconsin, Center for Parallel Optimization.

[3] Ahmed, I., Beg, M. R., Gupta, K. K., & Mansoori, M. I. (2012). A Novel approach of query optimization for genetic population. *International Journal of Computer Science Issues (IJCSI)*, *9*(2), 85-91.

[4] Sharma, R. V, Pushpneel. E, & Chaundhary, E. S. (2015). Query Optimization Concepts in Distributed Database. *International Journal of engineering Technology Science and Research* 2: 60-65.

[5] Butey, P. K., Meshram, S., & Sonolikar, R. L. (2012). Query Optimization by Genetic Algorithm. *Journal of Information Technology and Engineering*, 3(1), 44-51.

[6] Garg, A. & Juneja, D. (2012). A Comparison and Analysis of Various Extended Techniques of Query Optimization. *International Journal of Advantages in Technology*, 3(3), 184-194.

[7] Arebi, P., & Gonbadipoor, N. (2011, March). A Genetic Algorithm for Query Optimization in Database Grid by Dynamic Cost Estimation. In *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on* (pp. 81-86). IEEE.

[8] Widodo, A. W, & Mahmudy, W. F. (2010). Penerapan Algoritma Genetika Pada Sistem Rekomendasi Wisata Kuliner. *Jurnal Ilmiah Kursor* 5(4): 205-211.

[9] Ashari, I. A, Muslim, M. A., & Alamsyah. (2016). Comparison Perfomance of Genetic Algorithm and Ant Colony Optimization in Course Scheduling Optimizing. *Scientific Journal of Informatics*, 3(2), 149-158.

[10] Indra, Zulfahmi & Subanar. 2014. Optimasi Biaya Distribusi Rantai Pasok Tiga Tingkat dengan Menggunakan Algoritma Genetika. *IJCCS* 8(2): 189-200.

[11] Indroprasto & Suryani, E. (2012). Analisis Pengendalian Persedian Produk dengan Metode EOQ Menggunakan Algoritma Genetika untuk Mengefisiensi Biaya Persediaan. *Jurnal Teknik ITS*, 1(1), 305-309.

[12] Muzid, S. (2014). Dinamisasi Parameter Algoritma Genetika Menggunakan Population Resizing On Fitness Improvement Fuzzy Evolutionary Algorithm (PROFIFEA). *Prosiding SNATIF*, 471-478.

[13] Haupt, Randy L. & Haupt, Ellen Sue. (2004). *Practical Genetic Algorithms.* 2nd. New Jersey: A John Wiley & Sons, Inc.

[14] Bayir, M. A., Toroslu, Ismail H., & Cosar, Ahmet. (2007). Genetic Algorithm for the Multiple-Query Optimization Problem. *IEEE Transaction On System.*

Man. and Cybernetics-Part C: Application and Reviews 37(1), 147-153.
[15] Potgieter, A. & Engelbrencht, A.P. (2007). Genetic Algorithms for The Structural Optimsation of Learned Polynomial Expression. *Applied Mathematics and Computation* 186: 1441-1466.
[16] Petkovic, D. (2010, April). Comparison of different solutions for solving the optimization problem of large join queries. In *Advances in Databases Knowledge and Data Applications (DBKDA), 2010 Second International Conference on* (pp. 51-55). IEEE.
[17] Li, H., & Luo, B. (2008, September). A Tree-based genetic algorithm for distributed database. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on* (pp. 2614-2618). IEEE.
[18] Fitriana, E. N, & Sugiharti, E. (2015). Implementasi Algoritma Genetika dengan Teknik Kendali Logika Fuzzy untuk Mengatasi Traveling Salesman Problem Menggunakan Matlab. *UNNES Journal of Mathematics*, 4(2): 114-121.
[19] Mahmudy, W. F., & Rahman, M. A. (2011). Optimasi Fungsi Multi-Obyektif Berkendala Menggunakan Algoritma Genetika Adaptif dengan Pengkodean Real. *Jurnal Ilmiah Kursor*, 6(1): 19-26.
[20] Hoseini, P., & Shayetsteh, M. G. (2013). Efficient Contrast Enhancement of Images using Hibrid Ant Colony Optimization, Genetic Algorithm, and Simulated Annealing. *Digital Signal Processing*, 23(3): 879-893.
[21] Arifudin, R. (2012). Optimasi Penjadwalan Proyek dengan Penyeimbangan Biaya Menggunakan Kombinasi CPM dan Algoritma Genetika. *Jurnal Masyarakat Informatika*, 2(4): 1-14.
[22] Sofwan, Aghus, Handoyo, Eko, & WD, Ramadhony. (2008). Algoritma Genetika Dalam Pemilihan Spesifikasi Komputer. *Seminar Nasional Aplikasi Teknologi Informasi.* Yogyakarta. 1-6.
[23] Purwana, N., Esmeralda Djamal, C., & Renaldi, F. (2016). Optimalisasi Penempatan Dosen Pembimbing Dan Penjadwalan Seminar Tugas Akhir Menggunakan Algoritma Genetika. In *Seminar Nasional Teknologi Informasi dan Komunikasi, Maret*.
[24] Sari, F. A., Sugiharti, E., & Dwijanto. 2013. Implementasi Algoritma Genetika untuk Menyelesaikan Traveling Salesman Problem. *UNNES Journal of Mathematics* 2(2): 117-120.