JSS JOURNAL OF SMART SYSTEM

Vol. 1 No. 1 Juli 2021

Optimasi Query Hash Join Dan Inner Join Pada Sistem Pencarian Data Tracer Study

(Optimization Of Query Hash Join And Inner Join In Tracer Study Data Search System)

Moh. Erkamim[1)], Farid Fitriyadi[2)], Ikhwan Baidlowi Sumafta[3)]
[1)3)]Sistem Informasi Kota Cerdas, Fakultas Teknik, Universitas Tunas Pembangunan Surakarta
[2)]Informatika, Fakultas Teknik, Universitas Sahid Surakarta
E-mail: [1)]erkamim@lecture.utp.ac.id, [2)]faridfitriyadi@gmail.com, [3)] ibsumafta@lecture.utp.ac.id

Abstrak

Optimasi query merupakana solusi dalam permasalahan kompleknya query yang kita buat guna menghasilkan data dengan kondisi tertentu, optimasi query memberikan sebuah model pemecahan masalah dengan menggabungkan teknik-teknik yang meliputi transformasi logika serta mempresentasikan teknik tersebut dalam sebuah masalah. Semua model query yang menghasilkan (output) yang sama tetapi waktu proses yang berbeda-beda. Pemecahan masalah dengan membandingkan dua bentuk algoritma yang paling banyak digunakan merupakan salah satu yang menjadi pertimbangan peneliti untuk mengetahui mana yang terbaik yang dapat digunakan pada kondisi tertentu. Hash join dan Inner join query merupakan query yang termasuk banyak digunakan untuk sebuah rancang bangun basis data. Masalah paling penting adalah bagaimana menentukan query dengan akses tercepat dari query tersebut pada sebuah basis sistem tertentu, sehingga menjadikan judul ini sebagai pertimbangan dari penulis untuk melakukan optimasi query pencarian data alumni dengan Hash Join query.

Kata Kunci— Optimasi, Tracer Study,Query, Hash Join, Inner Join

Abstract

Query optimization a solution to the complex problem of queries that we make to produce data with certain conditions, query optimization provides a problem solving model by combining techniques that include logical transformations and presenting these techniques in a problem. All query models that produce (output) the same but different processing times. Problem solving by comparing the two forms of the most widely used algorithm is one of the considerations for researchers to find out which one is the best that can be used in certain conditions. Hash join and Inner join query are queries that are widely used for database design. The most important problem is how to determine the query with the fastest access from the query on a certain basis system, so that this title is considered by the author to optimize the alumni data search query with Hash Join queries.

Keywords— *Optimization, Tracer Study, Query, Hash Join, Inner Join*

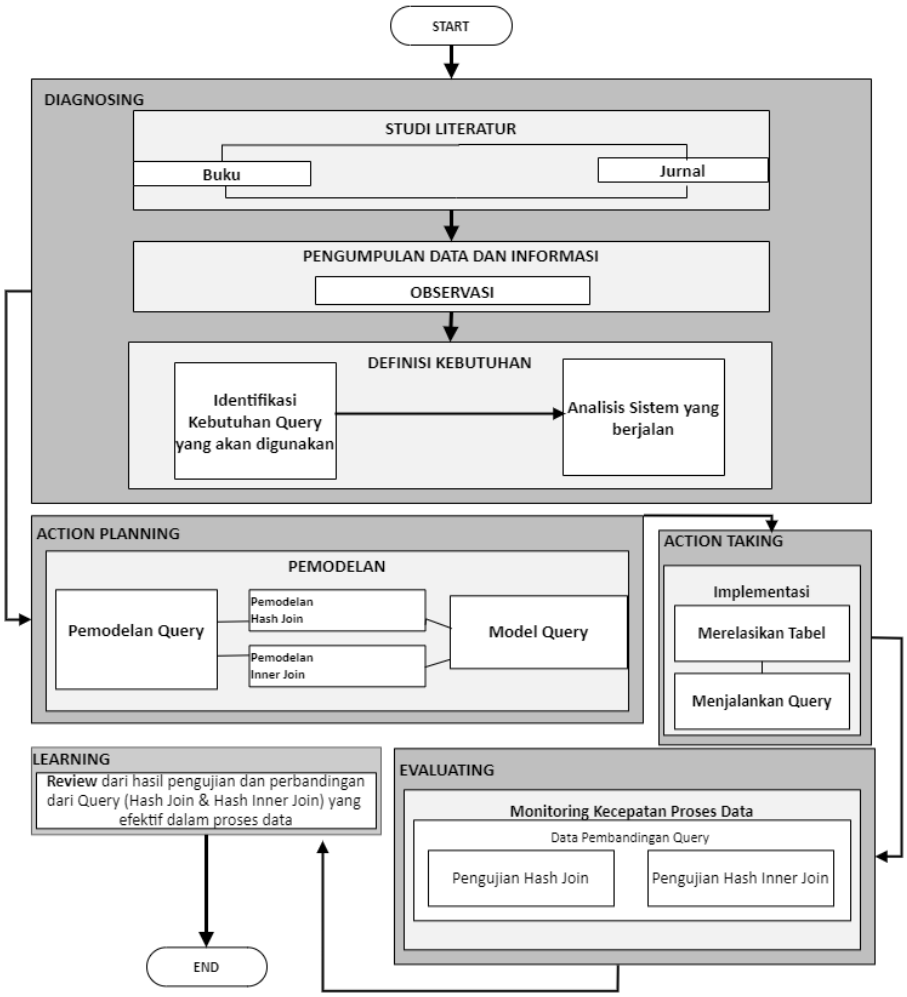
1. Pendahuluan

Query merupakan suatu perintah untuk melakukan pengolahan data dari database dimana dapat memanggil tabel-tabel yang ada pada *database* tersebut, namun tabel tersebut tidak semua ditampilkan, hanya sesuai data yang diinginkan atau data yang ingin ditampilkan [1]. *Join table* merupakan gabungan beberapa tabel menggunakan *Query* yang dilakukan pada kolom *key* yang memiliki nilai terkait untuk mendapatkan satu set data dengan informasi lengkap [2]. *Join* diperlukan karena perancangan tabel pada sistem transaksional kebanyakan dinormalisasi, salah satu alasannya untuk mengurangi redundansi[3]. Pencarian data dengan menggunakan *Query* atau *Join* pada *database* perlu memperhatikan ketepatgunaan implementasi dari data itu sendiri serta waktu prosesnya [3]. Banyak model menemukan sebuah *query*, meskipun banyak *query* yang dapat digunakan dalam sebuah kasus dalam pengolahan *database*, misalnya *query* yang digunakan untuk pencarian data, akan menghasilkan *output* yang sama, namun dengan proses yang berbeda tergantung bagaimana struktur dan banyaknya *record* dalam *database* tersebut [4]. Dari semua *query* yang dapat digunakan, khususnya *query* yang digunakan untuk pencarian data, sangat diperhatikan waktu kecepatan akses yang diperlukan untuk mencari sebuah data, *query* mana saja yang memiliki akses lebih cepat untuk mendapatkan sebuah nilai, khususnya untuk *record* data yang besar. *hash join query* memiliki fungsi untuk efisiensi dalam *database management system* [5].

Permasalahan pada penelitian ini ialah, penulis ingin mencari query untuk pencarian data dengan akses data yang lebih cepat dengan penggunaan waktu yang lebih minimum dengan model *hash join query* dengan menggunakan *database* sistem *tracer study* yang ada pada Universitas Tunas Pembangunan [6]. Dari permasalahan di atas menjadi alasan penulis untuk melakukan penelitian dengan judul " *Optimasi Query Hash Join* Dan *Inner Join* Pada Sistem Pencarian Data *Tracer Study*".

2. Metode Penelitian

Action research atau penelitian tindakan dilakukan dengan membuat sistematik yang terfokus pada penyempurnaan proses atau perbaikan teknik dalam menjalankan query yang sudah digunakan dengan membuat pembandingan antara *query hash join* dan *inner join* dari sisi kecepatan dalam mengelola *record* data.



Gambar 1. Alur Penelitian

a. Melakukan diagnosa (*diagnosing*)

Identifikasi masalah pokok yang digunakan dasar optimasi *query* yang akan digunakan dalam ujicoba pembandingan optimasi.

b. Membuat rencana tindakan (*action planning*)

Menentukan model *query* yang akan digunakan dalam memperoleh informasi dari sisi kecepatan dalam mengelola *record*.

c. Melakukan tindakan (*action taking*)

Mengimplementasikan model *query hash join* dan *inner join* serta melakukan ujicoba dalam merelasikan data maupun tabel.

d. Melakukan evaluasi (*evaluating*)

Setelah implementasi (*action taking*) dilanjutkan dengan evaluasi hasil dari implementasi untuk mengetahui perbandingan dari kedua model *query* yang digunakan.

e. Pembelajaran (*learning*)

Mengetahui informasi model query dari sisi kecepatan dalam mengelola record.

3. Hasil dan Pembahasan

Penelitian ini untuk mengetahui kinerja optimalisasi *query hash join* dan *inner join* dilakukan dengan mengevaluasi hasil kinerja *query* berdasarakan kecepatan proses. Adapun hasil dalam pengujian optimalisasi kinerja kecepatan *query* ditampilkan pada tabel-tabel hasil *record* sebagi berikut

a. Hasil perbandingan waktu proses kinerja *query*

Hasil ujicoba kinerja *query hash join* dengan *inner join* menghasilkan kinerja *query* dari sisi kecepatan dengan 1 sampai 3 relasi, adapun hasil kinerja ini sebagai berikut.

1) Hasil perbandingan waktu akses *query* satu relasi
   Hasil ujicoba kinerja *query hash join* dengan *inner join* satu relasi dalam hal perbandingan waktu terdapat selisih hasil pemprosesan *query*.

Tabel 1. Kinerja *query hash join* dengan *inner join* satu relasi

No	Jumlah Data/*Record*	Waktu (second/detik)	
		Hash Join	Inner Join
1	166	0.0015	0.00015
2	322	0.0022	0.00028

2) Hasil perbandingan waktu akses query dua relasi
   Hasil ujicoba kinerja *query hash join* dengan *inner join* dua relasi dalam hal perbandingan waktu terdapat selisih hasil pemprosesan *query*.

Tabel 2. Kinerja *query hash join* dengan *inner join* satu relasi

No	Jumlah Data/*Record*	Waktu (second/detik)	
		Hash Join	Inner Join
1	166	0.0018	0.00020
2	322	0.0030	0.00033

3) Hasil perbandingan waktu akses *query* tiga relasi
   Hasil ujicoba kinerja *query hash join* dengan *inner join* tiga relasi dalam hal perbandingan waktu terdapat selisih hasil pemprosesan *query*.

Tabel 3. Kinerja *query hash join* dengan *inner join* satu relasi

No	Jumlah Data/*Record*	Waktu (second/detik)	
		Hash Join	Inner Join
1	166	0.0027	0.00029
2	322	0.0032	0.00041

b. Pembahasan

Pada tahap ini memerlukan identifikasi berupa perancangan data yang akan dilakukan ujicoba mulai dari rancangan *database* sampai dengan *query* pada *record* terntentu yang telah ditentukan untuk mengetahui perbandingan penggunaan teknik *query hash join* dengan *inner join* adapun proses dalam proses penelitian ini dengan pengujian pada kasus data yang sudah kita sediakan adapun proses sebagai berikut.

1) Rancangan *Database*
Merupakan data tabel yang disediakan untuk melakukan ujicoba langsung terhadap kinerja masing-masing *query*, adapun *database* tersebut sebagai berikut.

Tabel 4. Tabel *User*

Nama *Field*	*Type*	Ukuran
user_id	*int*	10
username	*varchar*	10
password	*varchar*	50
level	*enum*	-
aktif	*enum*	-
date_create	*timestamp*	-
enc	*varchar*	50

Tabel 5. Tabel Biodata

Nama *Field*	*Type*	Ukuran
biodata_id	*int*	10
id_user	*int*	10
nim	*varchar*	10
nama	*varchar*	50
id_prodi	*int*	1
tempat_lahir	*varchar*	50
tanggal_lahir	*date*	-
jenis_kelamin	*enum*	-
agama	*enum*	-
warga_negara	*varchar*	30
alamat	*varchar*	100
tlpn	*varchar*	15
email	*varchar*	50
tahun_lulus	*date*	
foto	*varchar*	40
ipk	*decimal*	3.2
skripsi	*text*	
status_kerja	*varchar*	10
tempat_kerja	*varchar*	50
tahun_wisuda	*varchar*	4

Tabel 6. Tabel Keluarga

Nama *Field*	*Type*	Ukuran
keluarga_id	int	10
nama_ayah	varchar	50
nama_ibu	varchar	50
alamat_keluarga	varchar	100
tlpn_keluarga	varchar	15
id_user	Int	10

Tabel 7. Tabel Fakultas

Nama *Field*	*Type*	Ukuran
fakultas_kode	varchar	10
fakultas	varchar	50

Tabel 8. Tabel Prodi

Nama *Field*	*Type*	Ukuran
prodi_id	int	2
prodi	varchar	50
prodi_kode	varchar	10
kode_fakultas	varchar	10

2) *Query* Algoritma

Pengujian pencarian data menggunakan query dua algoritma yaitu *Natural Join* dan *Hash Join*. Pengujian *Query* dalam hal pencarian data berdasarkan banyak jumlah data untuk melakukan pencarian data dibagi atas beberapa tahap relasi. Pengujian aplikasi ini menggunakan phpmyadmin sql sebagai *server* penyimpanan *database*. Pengujian kinerja *query* Algoritma dalam pencarian data dari sebuah aplikasi maka *query* dibagi berdasarkan jumlah tabel yang saling berhubungan.

a) *Query* 1 Relasi
Menggunakan 2 (dua) tabel yaitu tabel *user* dan biodata yang saling berhubungan antar tabel.

```
Hash Join Query
Select user.*, biodata.* from user, biodata where user.user_id =
biodata.id_user ;
Inner Join Query
Select user.*, biodata.* from user
join biodata on user.user_id = biodata.id_user;
```

b) *Query* 2 Relasi
Menggunakan 3 (tiga) tabel yaitu tabel *user*, biodata, keluarga yang saling berhubungan antar tabel.

```
Hash Join Query
Select user.*, biodata.*, keluarga.* from user, biodata, keluarga
where user.user_id = biodata.id_user and user.user_id =
keluarga.id_user;
    Inner Join Query
```

```
Select user.*, biodata.*, keluarga.* from user join biodata on
user.user_id = biodata.id_user join keluarga on user.user_id =
keluarga.id_user;
```

c) *Query* 3 Relasi
Menggunakan 4 (empat) tabel yaitu tabel user, biodata, keluarga dan prodi yang saling berhubungan antar tabel.

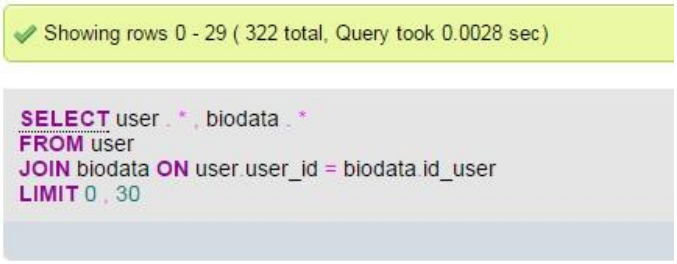
```
Hash Join Query
Select user.*, biodata.*, keluarga.*, prodi.* from user, biodata,
keluarga, prodi where user.user_id=biodata.id_user and
user.user_id=keluarga.id_user and
biodata.id_prodi=prodi.prodi_id;
Inner Join Query
Select user.*, biodata.*, keluarga.*, prodi.* from user join biodata
on user.user_id=biodata.id_user
join keluarga on user.user_id = keluarga.id_user join prodi on
biodata.id_prodi=prodi.prodi_id;
```

3) Hasil Pengujian *Query*
a) Query 1 Relasi
Menggunakan 2 (dua) tabel yaitu tabel *user* dan biodata yang saling berhubungan antar tabel
Hash Join Query

Gambar 2. *Hash Join Query* 1 Relasi

Pada hasil proses *hash join query* eksekusi 1 relasi 2 tabel dengan jumlah 322 *record* didapat proses waktu yang dibutuhkan untuk menampilkan data 0.0022 detik/*second* seperti pada gambar 2.

Inner Join Query

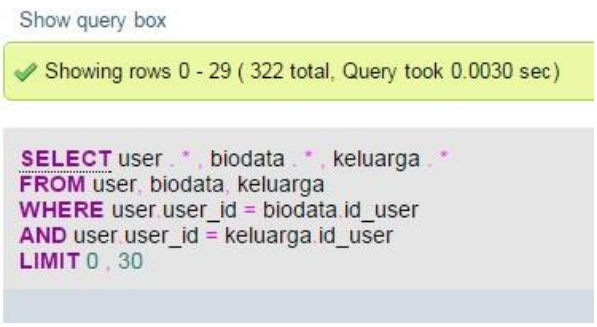


Gambar 3. *Inner Join Query* 1 Relasi

Hasil proses *Inner Join Query* eksekusi 1 relasi 2 tabel dengan jumlah 322 *record* didapat proses waktu lebih lama yang dibutuhkan untuk menampilkan data 0.0028 detik/*second* seperti hasil pada gambar 3.

b) *Query* 2 Relasi
Menggunakan 3 (tiga) tabel yaitu tabel *user*, biodata, keluarga yang saling berhubungan antar tabel

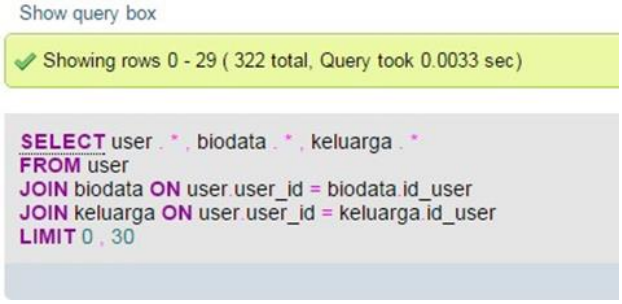
Hash Join Query



Gambar 4. *Hash Join Query* 2 Relasi

Pada hasil proses *hash join query* eksekusi 2 relasi 3 tabel dengan jumlah 322 *record* didapat proses waktu yang dibutuhkan untuk menampilkan data 0.0030 detik/*second* jika dibandingkan dengan 2 tabel menghasilkan selisih waktu proses seperti pada gambar 4.

Inner Join Query



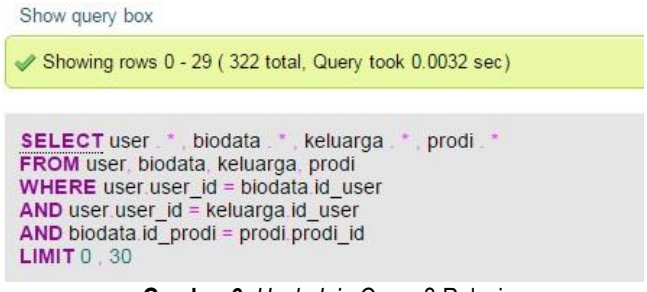
Gambar 5. *Inner Join Query* 2 Relasi

Hasil proses Inner Join Query eksekusi 2 relasi 3 tabel dengan jumlah 322 *record* didapat proses waktu lebih lama dengan sebelumnya yang membutuhkan waktu proses menampilkan data 0.0033 detik/*second* seperti hasil pada gambar 5.

c) *Query* 3 Relasi

Menggunakan 4 (empat) tabel yaitu tabel *user*, biodata, keluarga dan prodi yang saling berhubungan antar tabel

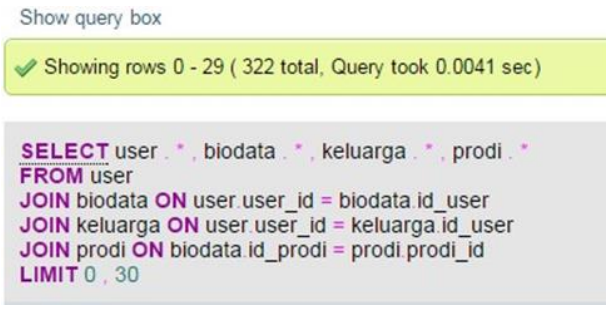
Hash Join Query



Gambar 6. *Hash Join Query* 3 Relasi

Pada hasil proses *hash join query* eksekusi 3 relasi 4 tabel dengan jumlah 322 *record* didapat proses waktu yang dibutuhkan untuk menampilkan data 0.0032 detik/*second* jika dibandingkan dengan 3 tabel menghasilkan selisih waktu proses seperti pada gambar 6.

Inner Join Query



Gambar 7. *Inner Join Query* 3 Relasi

Hasil proses Inner Join Query eksekusi 3 relasi 4 tabel dengan jumlah 322 *record* didapat proses waktu lebih lama dengan sebelumnya yang membutuhkan waktu proses menampilkan data 0.0041 detik/*second* seperti hasil pada gambar 7.

4. Kesimpulan

Penggunaan *query* begitu penting dalam pengelolaan *Database Managenent System* (DBMS) yang menghubungkan setiap tabel yang tersimpan didalam sebuah database. Karena banyaknya bentuk query yang bisa digunakan sebagai alternative untuk mencari *query* yang efektif dan efisien dalam setiap mengambilan data relasinya, maka dapat mengetahui query apa yang efektif dalam aplikasi kita. Kecepatan query sangat tergantung dari kondisi seperti *server*, kecepatan jaringan struktur database, perangkat, keras/*hardware*, dan juga sistem operasi. Adapun hasil dari pengujian dua query pada penelitian ini adalah sebagai berikut didapat bahwa dari setiap pengujian query, bahwa hash join query mencatat jumlah waktu lebih cepat saat record data berjumlah 1-322 data, atau dengan jumlah data yang lebih banyak. Perhitungan waktu atau kecepatan query pada web browser cenderung berubah pada setiap sesi pengujian, namun tidak mengubah posisi query tercepat dan terlama waktu aksesnya.

Referensi

[1] J. Teknovasi, J. Sinuraya, M. Zarlis, E. B. Nababan, and H. Join, “Perbandingan Pencarian Data Menggunakan,” vol. 01, pp. 71–93, 2014.

[2] T. Hastono, “Optimasi Query Sistem Informasi Menggunakan Stored Procedure,” *J. Din. Inform.*, vol. 8, no. 2, pp. 79–89, 2019.

[3] E. Helmud, “Optimasi Basis Data Oracle Menggunakan Complex View Studi Kasus : Pt. Berkat Optimis Sejahtera (Pt.Bos) Pangkalpinan,” *J. Inform.*, vol. 7, no. 1, pp. 80–86, 2021.

[4] M. M. Dewi and N. Rezeki, “Analisis Perbandingan Optimasi Query Nasted Join dan Hash Join pada MySQL Server,” *CSRID (Computer Sci. Res. Its Dev. Journal)*, vol. 9, no. 1, p. 31, 2017, doi: 10.22303/csrid.9.1.2017.31-42.

[5] F. Rochman, “Analisis Perbandingan Cartesian Product, Cross Join, Inner Join dan Outer Join dalam Si Akad,” *Techno (Jurnal Fak. Tek. Univ. Muhammadiyah Purwokerto)*, vol. 19, no. 1, p. 37, 2018, doi: 10.30595/techno.v19i1.2184.

[6] N. Fajaryati, T. Sukardiyono, A. Dwi, W. Utami, S. Pambudi, and B. Destiana, “Studi Penelusuran (Tracer Study) Terhadap Alumni Program Studi Pendidikan Teknik Informatika Jurusan,” *J. Electron. Informatics, Vocat. Educ.*, vol. 1, no. November, pp. 231–248, 2015.