Analisis Kinerja Optimasi *Query Cross Join*, *Natural Join* Dan *Full Outer Join*

Performance Analysis Of Optimization Query Cross Join, Natural Join And Full Outer Join

Yohanes Aryo Bismo Raharjo[1], Daniel Mantriwira [2], Fendi Sumanto[3]
[1] Magister Teknik Informatika Universitas AMIKOM Yogyakarta
E-mail: [1] aryo.bismo66@gmail.com, [2] danielmantriwira797@gmail.com,
[3]fendisumanto@gmail.com

Abstrak

Kebutuhan sebuah sistem untuk menyimpan data membuat database sangat penting untuk dimiliki institusi, perusahaan maupun organisasi agar dapat membantu dalam memproses data supaya dapat menghasilkan informasi yang berguna yaitu pengolahan basis data atau database. Dalam mengelola data dalam jumlah banyak tentu sering menemui kendala pada saat ingin mengakses data tersebut sehingga memerlukan teknik – teknik khusus untuk mengatasi masalah tersebut salah satunya dengan melakukan optimasi query. Pentingnya optimasi query untuk meminimalisir waktu akses perlunya ada teknik optimasi yang unggul. Dengantiga tahapan, yaitu tahap persiapan ialah proses pengumpulan data sampai pembuatan database, pengujian optimasi query merupakan proses pengujian menggunakan cross join, natural join dan full outer join, kemudian tahap analisis dari hasil pengujian optimasi query. Pada penelitian ini bertujuan untuk menemukan optimasi yang unggul dengan melakukan pengujian perbandingan kinerja pada optimasi query cross join, natural join dan full outer join berdasarkan lamanya respon time sesuai dengan hasil pengujian setiap teknik optimasi query, agar dapat menghasilkan teknik query yang unggul untuk membantu dalam menetukan optimasi query mana yang baik digunakan untuk membantu pekerjaan pengguna dalam skala besar.

Kata Kunci— *Database, Optimasi Query, Cross Join, Natural Join, Full Outer Join*

Abstract

The need of a system to store data makes the database is very important for owned institutions, companies and organizations in order to assist in processing data in order to generate useful information that is processing database or database. In managing the data in large numbers of course often encounter obstacles when wanting to access the data so it requires special techniques to solve the problem one of them by doing query optimization. The importance of query optimization to minimize access time needs a superior optimization technique. With three stages, namely the preparation phase is the process of data collection until the creation of the database, query optimization testing is a test process using cross join, natural join and full outer join, then the analysis phase of query optimization testing results. This research aims to find a superior optimization by performing comparative performance testing on cross-join query optimization, natural join and full outer join based on the duration of response time in accordance with the test results of each query optimization technique, in order to produce superior query technique to assist in determining Which query optimization is good used to help the user work on a large scale.

Keywords— *Database, Optimization Query, Cross Join, Natural Join, Full Outer Join*

1. PENDAHULUAN

1.1 Latar Belakang

Penggunaan teknologi pada sebuah institusi, perusahaan maupun organisasi memiliki yang sangat penting peran penting untuk mencapai tujuan. Suatu perusahaan dituntut untuk bekerja dengan efektif dan efisien supaya dapat bertahan di atas kerasnya persaingan. Salah satu teknologi yang harus dimiliki oleh sebuah institusi, perusahaan maupun organisasi adalah teknologi dalam memproses data sehingga menjadi informasi yang beguna, teknologi yang dimaksud adalah sistem pengolahan basis data atau *database*. Penggunaan *database* yang baik pada perusahaan retail misalnya, mampu membantu seorang kasir bekerja lebih baik dan cepat ketika ingin mengakses ataupun mencari jumlah barang atau harga barang yang akan dijual. Begitupun dengan halnya seorang admin *database* dapat membantu mempermudah ketika ingin melakukan pencarian data stok persediaan, data barang yang paling laku dalam pembuatan laporan. Sistem database merupakan komponen dasar sistem informasi dari perusahaan besar, sistem pengembangan siklus *database* secara melekat terkait dengan siklus hidup sistem informasi. Pentingnya untuk mengakui bahwa tahapan pengembangan siklus *database* tidak harus berurutan tetapi melibatkan beberapa jumlah pengulangan tahapan sebelumnya melalui *feedback loop* [1]. Salah satu aspek yang sulit dalam perancangan database adalah kenyataan bahwa perancang, *programmer* dan pemakai akhir cenderung melihat data dengan cara yang berbeda. Metodologi perancangan adalah sebuah pendekatan terstruktur yang menggunakan prosedur, teknik, peralatan, dan dokumentasi untuk mendukung dan memfasilitasi proses perancangan. Metodologi perancangan terdiri dari beberapa fase dimana setiap fase mengandung beberapa langkah yang akan menuntun desainer dalam menggunakan teknik yang sesuai pada setiap tahap dalam proyek sehigga membantu desainer untuk merencanakan, mengelola, mengatur, dan mengevaluasi pengembangan proyek *database*[1]. Tujuan dari optimasi *query* adalah menemukan jalan akses yang paling baik 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 [2]. SQL adalah *transform-oriented language* atau bahasa yang dirancang dengan penggunaan relasi untuk mengubah masukan menjadi keluaran yang dibutuhkan. Sebagai sebuah bahasa, standar internasional SQL menetapkan dua komponen pokok yaitu *Data Definition Language* (DDL) untuk mendefisinikan struktur basis data dan akses kontrol data dan *Data Manipulation Language* (DML) untuk mengembalikan dan perbarui data [3].

Pengolahan data tersebut dapat dilakukan dengan mengakses data yang terdapat dalam *database*. Pengaksesan data tersebut dilakukan dengan melakukan *query-query* pada basis data-basis data dengan *database* manajemen sistem [4]. Pengaksesan data pada database perlu memperhatikan ketepatgunaan implementasi dari data itu sendiri serta waktu prosesnya. Ada banyak cara yang dapat dilakukan oleh database manajemen sistem dalam memproses dan menghasilkan jawaban sebuah *query*. Semua cara pada akhirnya akan menghasilkan jawaban (output) yang sama tetapi pasti mempunyai harga yang berbeda-beda, seperti misalnya total waktu yang diperlukan untuk menjalankan sebuah *query* [4]. Optimasi *query* salah satu teknik yang digunakan untuk menghemat *cost of evaluating a query* atau lama nya pengguna menunggu waktu respon, sehingga optimasi *query* sangat dibutuhkan untuk mengakses *big data* atau data yang sangat banyak karena biasanya akan membutuhkan waktu dalam mengakses data yang banyak. Dimana optimasi *query* memberikan sebuah cara pemecahan masalah dengan mencoba menggabungkan teknik-teknik yang meliputi transformasi logika dan mempresentasikan teknik tersebut dalam sebuah masalah. Pemecahan masalah dengan membandingkan dua bentuk algoritma yang paling banyak digunakan untuk mengetahui mana yang terbaik yang dapat digunakan pada kondisi tertentu [5]. Sedangkan *Query* merupakan kemampuan untuk menampilkan suatu data dari *database* dimana mengambil dari tabel-tabel yang ada pada *database* tersebut, namun tabel tersebut tidak semua ditampilkan, hanya sesuai data yang diinginkan atau data yang ingin ditampilkan [5]. Dalam aljabar relasional (*relational algebra*) telah dikemukakan oleh EF Codd yang merupakan keluarga aljabar dengan semantik yang dibangun untuk memodelkan data yang disimpan dalam basis data relasional dan mendefinisikan *query*-nya. Aljabar Relasional (AR) pertama kali digunakan untuk menganalisa secara matematis data perbankan yang sangat besar dengan 5 (lima) operasi dasar yaitu seleksi (*selection*), proyeksi (*projection*), perkalian (*Cartesian product*),

penggabungan himpunan (*set union*) dan selisih himpunan (*set difference*) [6]. *Cartesian product* dalam beberapa literatur disebut juga *cross product* atau *cross join*, sedangkan set difference terdapat empat bagian yaitu *intersection*, *left outer join, right outer join* dan *full outer join*[6]. Penggunaan relasi *cross join* sama dengan *straight join* yaitu untuk menampilkan kedua table yang direlasikan dengan menampilkan semua record meskipun tidak bersesuaian dengan table yang lain dengan jumlah record adalah hasil kali jumlah record table pertama dengan jumlah record tapi kedua. Sedangkan penggunaan *natural join*, bisa digunakn jika pada tabel yang di join terdapat kolom atau atribut yang memiliki nama yng sama dan tipe data yang sama. Pada waktu join kolom tersebut secara otomatis akan digunakan untuk dibandingkan, dan tidak bisanmenentukan kolom mana yang akan digunakn untuk dibandingkan dan *full outer join***,** akan menampilkan semua baris yang terletak pada kiri dan kanan *syntax* join walopun tidak sesuai ketika dibandingkan.

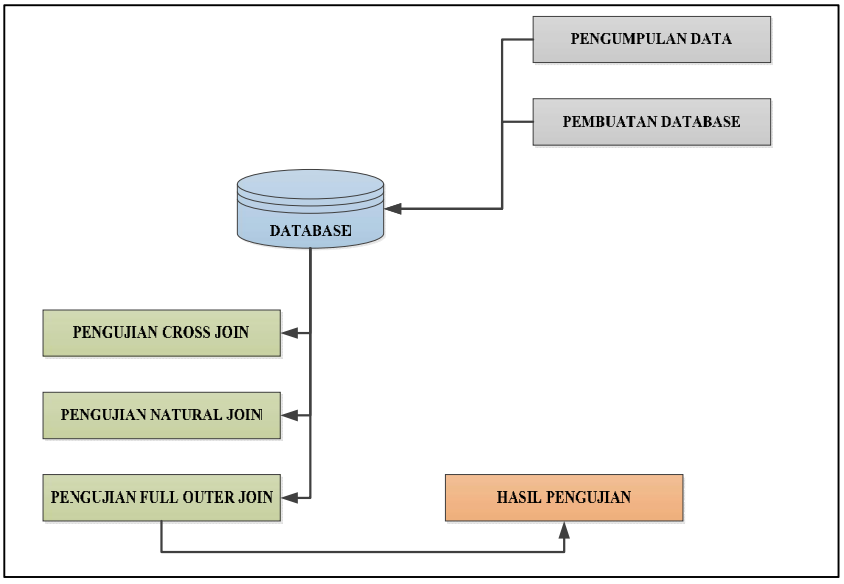
Berdasarkan uraian di atas pentingnya database untuk mengelola penyimpanan data dalam jumlah besar namun dalam melakukan eksekusi database tersebut pengguna dari sistem database tersebut terkendala dalam menampilkan data, melakukan pencarian data dan mengelola data tersebut karena respons time yang memakan waktu maka dibutuhkannya teknik – teknik yang dapat mempercepat proses dalam bekerja yaitu dengan teknik optimasi *query* sehingga dalam penelitian ini akan membantu mencari teknik optimasi *query* yang tercepat dari teknik optimasi *query cross join*, *natural join* dan *full outer join.*

2. METODE PENELITIAN

Penelitian ini dilakukan melalui 3 tahapan, yaitu tahap persiapan ialah proses pengumpulan data sampai pembuatan database, pengujian optimasi *query* merupakan proses pengujian menggunakan cross join, natural join dan full outer join, kemudiantahap analisis dari hasil pengujian optimasi *query*. Pada tahap persiapan, dimulai dengan mengumpulkan beberapa data dengan membuat data dummy yang digunakan untuk tahap pengujian kemudian merancang relasi tabel. Setelah itu dilakukan pembuatan *database*di server dari data yang tersedia dan relasi tabel.

Pada tahap eksperimen, dilakukan eksperimen dengan menggunakan optimasi *querycross join, natural join dan full outer join* pada database yang telah disiapkankemudian dicatat lamanya waktu proses. Pada masing – masing teknik optimasi *query* dilakukan 10 kali percobaan demi memperoleh hasil yang valid dalam penelitian ini.

Pada tahap analisis hasil, melakukan rekapitulasi data hasil dari pengujian yang dilakukan, kemudian dibandingkan hasil dari eksperimen tersebut. Alur penelitian dapat dilihat pada [gambar 1].



Gambar 1. Alur Penelitian

3. HASIL DAN PEMBAHASAN

Pada uraian sebelumnya, maka aplikasi ini harus dijalankan dengan media web browser. Seperti Chrome, Mozilla, Opera, ataupun Safari. Aplikasi web browser kemudian dijalankan dengan sebuah *server domain* yaitu Apache yang sudah dipersiapkan dan diinstal lebih dulu pada aplikasi xampp dan menggunakan bahasa pemrograman php mysql. Dalam penelitian telah mempersiapkan jumlah *record* pada *database* penerbit buku dengan jumlah data 1000 baris data dengan relasi dengan tabel lainnya yang mana nilai kolom relasi saat join tabel penerbit buku tersebut harus memenuhi nilai yang sama dengan tabel lainnya saat penghitungan baris. Penghitungan waktu akses *query* ini adalah dengan menetapkan waktu awal sebelum proses menjalankan *query* dan menetapkan waktu akhir setelah eksekusi selesai sehingga didapatlah selisih waktu tersebut dan bernilai satuan detik yang kemudian menjadi catatan waktu mengujian dengan jumlah data seluruhnya dilakukan pencatatan dari *respon time* dari hasil optimasi *query cross join*, *natural join* dan *full outer join*. Pada penelitian ini dilakukan 3 (tiga) kali pengujian mulai dari 100 baris data, 500 baris data dan 1000 baris data pada setiap masing – masing teknik optimasi *query* untuk mentukan antara *cross join*, *natural join* dan *full outer join* yang lebih cepat jika dilihat dari *respon time* ketiga teknik tersebut.

3.1. Perancangan Database

Pada tahap ini peneliti memahami pokok masalah yang telah didapat pada analisa tahap pertama dan melakukan sebuah rencana penelitian yang dibentuk dalam satu desain program dan rancangan database pada objek penelitian yang telah ditentukan oleh Peneliti.Tabel-tabel database yang akan diuji adalah sebagai berikut :

Tabel 1. Tabel Buku

Field	Type	Ukuran
No_record	Interger	10
Judul	Varchar	100
Pengarang	Varchar	25
Edisi	Varchar	25
Jumlah	Interger	3

Tabel 2. Tabel Jenis

Field	Type	Ukuran
No_jenis	Interger	10
Nama_jenis	Varchar	35

Tabel 3. Tabel Klas

Field	Type	Ukuran
No_klas	Interger	10
Nm_klas	Varchar	35

Tabel 4. Tabel Koterbit

Field	Type	Ukuran
Id_kt	Interger	11
Nm_kt	Varchar	35

Tabel 5. Tabel Penerbit

Field	Type	Ukuran
No_penerbit	Interger	11
Nama_Penerbit	Varchar	35

3.2. Relasi Tabel

Relasi antar tabel dari data tabel akan di gambarkan pada [gambar 2] dibawah ini :

Gambar 2. Relasi Antar Tabel

3.3. Algoritma Query

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

Cross Join

```
SELECT tabel_buku.judul, tabel_buku.no_record, tabel_buku.pengarang,
tabel_penerbit.no_penerbit, tabel_penerbit.nama_penerbit
FROM tabel_buku
CROSS JOIN tabel_penerbit;
```

Natural Join

```
SELECT tabel_buku.judul, tabel_buku.no_record, tabel_buku.pengarang,
tabel_penerbit.no_penerbit, tabel_penerbit.nama_penerbit
FROM tabel_buku
NATURAL JOIN tabel_penerbit;
```

Full Outer Join

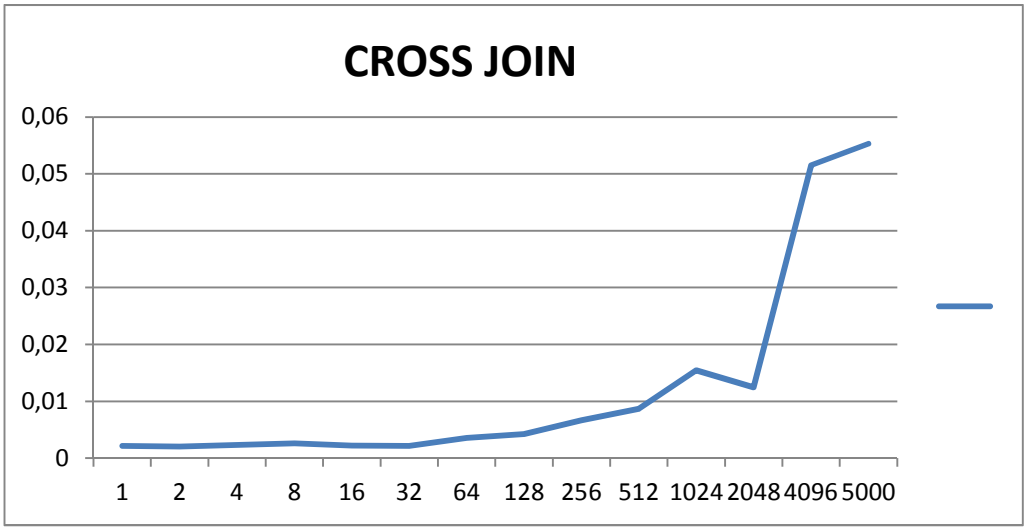
```
SELECT *
FROM tabel_buku
LEFT OUTER JOIN tabel_penerbit ON tabel_buku.no_record = tabel_penerbit.no_penerbit

UNION

SELECT *
FROM tabel_buku
RIGHT OUTER JOIN tabel_penerbit ON tabel_buku.judul = tabel_penerbit.nama_penerbit;
```

3.4. Pengujian Query Relasi Menggunakan Cross Join

Tabel 6. Tabel Hasi Pengujian Menggunakan *Cross Join*

JUMLAH DATA	CROSS JOIN										RATA - RATA
	1	2	3	4	5	6	7	8	9	10	
1	0.0021	0.0021	0.0021	0.0020	0.0029	0.0020	0.0020	0.0020	0.0021	0.0020	0.00213
2	0.0020	0.0021	0.0020	0.0021	0.0020	0.0021	0.0021	0.0020	0.0020	0.0020	0.00203
4	0.0027	0.0020	0.0020	0.0036	0.0026	0.0024	0.0020	0.0020	0.0020	0.0020	0.00233
8	0.0020	0.0069	0.0020	0.0020	0.0020	0.0020	0.0021	0.0020	0.0028	0.0021	0.00259
16	0.0021	0.0021	0.0020	0.0025	0.0022	0.0024	0.0020	0.0025	0.0025	0.0020	0.00223
32	0.0021	0.0026	0.0022	0.0021	0.0021	0.0020	0.0021	0.0020	0.0022	0.0020	0.00214
64	0.0021	0.0039	0.0039	0.0042	0.0034	0.0033	0.0033	0.0034	0.0035	0.0045	0.00355
128	0.0045	0.0044	0.0040	0.0042	0.0041	0.0041	0.0045	0.0040	0.0040	0.0048	0.00426
256	0.0054	0.0064	0.0053	0.0059	0.0054	0.0113	0.0054	0.0103	0.0059	0.0054	0.00667
512	0.0103	0.0088	0.0085	0.0081	0.0081	0.0085	0.0080	0.0105	0.0081	0.0080	0.00869
1024	0.0139	0.0139	0.0133	0.0321	0.0136	0.0134	0.0136	0.0136	0.0139	0.0133	0.01546
2048	0.0254	0.0243	0.0243	0.0246	0.0246	0.0242	0.0260	0.0251	0.0244	0.0241	0.01247
4096	0.0735	0.0456	0.0456	0.0460	0.0456	0.0752	0.0454	0.0455	0.0471	0.0458	0.05153
5000	0.0546	0.0546	0.0558	0.0547	0.0544	0.0542	0.0545	0.0611	0.0546	0.0546	0.05531

Pada pengujian menggunakan *cross join* dilakukan pengujian sebanyak 10 kali pada tiap jumlah baris data kemudian dihitung nilai rata – rata pada pengujian setiap jumlah baris data. Dari hasil pengujian menggunakan *cross join* pada jumlah data 1 sampai 1024 lamanya *respon time* menunjukan terus meningkat seiring bertambahnya jumlah data yang dieksekusi, namun pada pengujian jumlah data 2048 *respon time* mengalami penurunan dan pengujian sampai jumlah data 5000 *respon time* kembali naik, dijelaskan pada gambar grafik berikut :



Gambar 3. Grafik Pengujian *Cross Join*

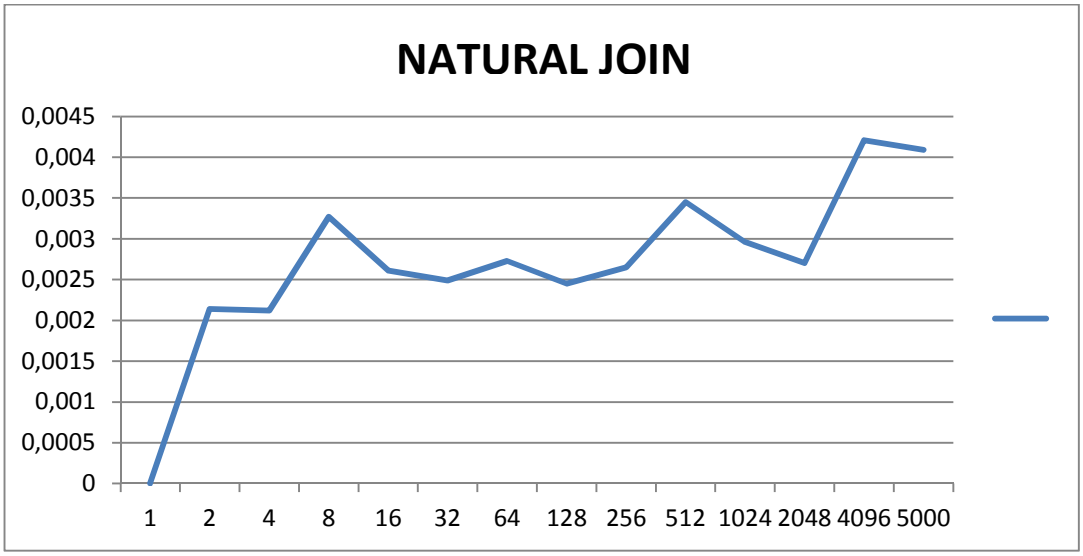
3.5. Pengujian Query Relasi Menggunakan Natural Join

Tabel 7. Tabel Hasi Pengujian Menggunakan *Natural Join*

JUMLAH DATA	Natural Join										RATA - RATA
	1	2	3	4	5	6	7	8	9	10	
1	0.0019	0.0019	0.0022	0.0018	0.0018	0.0022	0.0023	0.0025	0.0020	0.0024	0.00121
2	0.0028	0.0018	0.0018	0.0023	0.0018	0.0017	0.0019	0.0021	0.0029	0.0023	0.00214
4	0.0018	0.0022	0.0019	0.0018	0.0029	0.0018	0.0018	0.0029	0.0023	0.0018	0.00212
8	0.0026	0.0025	0.0019	0.0023	0.0023	0.0025	0.0023	0.0025	0.0025	0.0023	0.00327
16	0.0024	0.0049	0.0023	0.0019	0.0038	0.0023	0.0019	0.0024	0.0023	0.0019	0.00261
32	0.0022	0.0021	0.0028	0.0024	0.0020	0.0028	0.0028	0.0024	0.0026	0.0028	0.00249
64	0.0024	0.0021	0.0027	0.0022	0.0027	0.0027	0.0022	0.0027	0.0022	0.0027	0.00273
128	0.0021	0.0025	0.0026	0.0024	0.0024	0.0025	0.0025	0.0024	0.0026	0.0025	0.00245
256	0.0024	0.0021	0.0024	0.0027	0.0021	0.0024	0.0027	0.0027	0.0025	0.0028	0.00265
512	0.0052	0.0019	0.0045	0.0031	0.0027	0.0027	0.0022	0.0031	0.0031	0.0027	0.00345
1024	0.0037	0.0025	0.0024	0.0025	0.0022	0.0034	0.0032	0.0025	0.0031	0.0041	0.00296
2048	0.0027	0.0025	0.0023	0.0027	0.0021	0.0025	0.0031	0.0035	0.0029	0.0027	0.00270
4096	0.0020	0.0025	0.0037	0.0028	0.0028	0.0037	0.0036	0.0031	0.0047	0.0051	0.00421
5000	0.0065	0.0028	0.0038	0.0048	0.0035	0.0027	0.0024	0.0032	0.0027	0.0047	0.00409

Pada pengujian menggunakan *natural join* dilakukan pengujian sebanyak 10 kali pada tiap jumlah baris data kemudian dihitung nilai rata – rata pada pengujian setiap jumlah baris data. Dari hasil pengujian menggunakan *natural join* hasil percobaan pada jumlah data 1 sampai dengan 5000 data mengalami perubahan namun relatif stabil kenaikan *respon time* pada setiap jumlah data dan

hanya mengalami kenaikan *respon time* yang tidak terlalu signifikan, dijelaskan pada gambar grafik berikut : Digambarkan pada grafik berikut ini [gambar 4]:



Gambar 4. Grafik Pengujian *Natural Join*

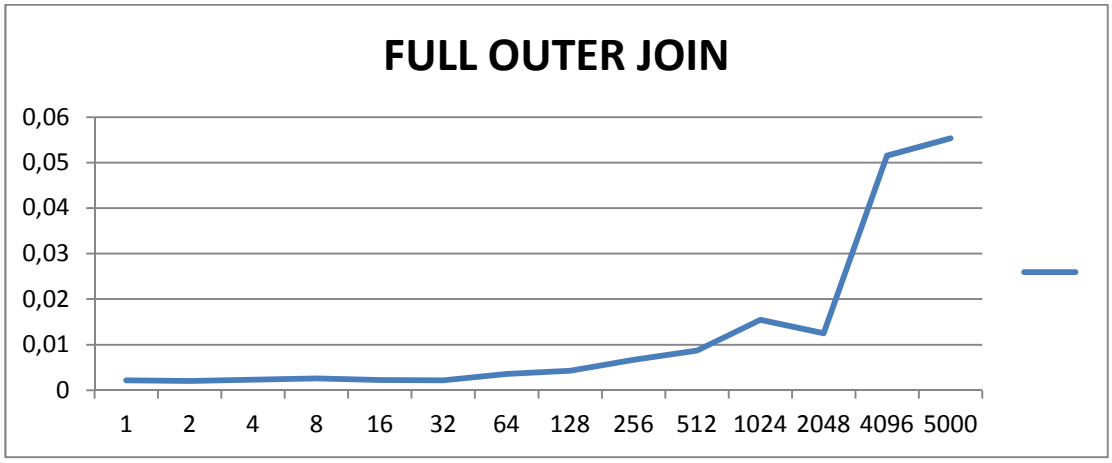
3.6. *Pengujian Query Relasi Menggunakan Full Outer Join*

Tabel 8. Tabel Hasi Pengujian Menggunakan *Full Outer Join*

JUMLAH DATA	Full Outer join										RATA - RATA
	1	2	3	4	5	6	7	8	9	10	
1	0.0038	0.0067	0.0036	0.0038	0.0046	0.0039	0.0097	0.0037	0.0054	0.0040	0.00492
2	0.0043	0.0038	0.0039	0.0039	0.0035	0.0041	0.0038	0.0039	0.0039	0.0041	0.00431
4	0.0043	0.0039	0.0043	0.0041	0.0041	0.0043	0.0042	0.0042	0.0041	0.0043	0.00418
8	0.0043	0.0042	0.0060	0.0045	0.0042	0.0042	0.0039	0.0045	0.0039	0.0042	0.00439
16	0.0044	0.0048	0.0044	0.0044	0.0045	0.0048	0.0045	0.0044	0.0044	0.0044	0.00450
32	0.0068	0.0065	0.0046	0.0047	0.0048	0.0046	0.0046	0.0048	0.0047	0.0072	0.00533
64	0.0062	0.0060	0.0062	0.0056	0.0057	0.0057	0.0056	0.0057	0.0057	0.0056	0.00642
128	0.0092	0.0083	0.0082	0.0083	0.0082	0.0082	0.0083	0.0082	0.0083	0.0092	0.00844
256	0.0125	0.0114	0.0109	0.0126	0.0125	0.0114	0.0109	0.0126	0.0127	0.0109	0.01214
512	0.0184	0.0691	0.0302	0.0186	0.0188	0.0190	0.0199	0.0217	0.0234	0.0345	0.02736
1024	0.0475	0.0794	0.0361	0.0361	0.0334	0.0356	0.0323	0.0309	0.0443	0.0354	0.04111
2048	0.0696	0.0692	0.0777	0.1009	0.0767	0.0388	0.0788	0.0361	0.0450	0.0769	0.06697
4096	0.1247	0.1631	0.0777	0.0886	0.0986	0.0777	0.0667	0.1234	0.0876	0.0889	0.09907
5000	0.1786	0.2341	0.2221	0.2431	0.1699	0.1699	0.1699	0.2442	0.2314	0.1456	0.20028

Pada pengujian menggunakan *full outer join* dilakukan pengujian sebanyak 10 kali pada tiap jumlah baris data kemudian dihitung nilai rata – rata pada pengujian setiap jumlah baris data. Dari

hasil pengujian menggunakan *full outer join* menunjukan grafik lamanya *respon time* terus mengalami kenaikan seiring dengan banyak nya jumlah data yang di eksekusi, digambarkan pada grafik berikut ini [gambar 5]:



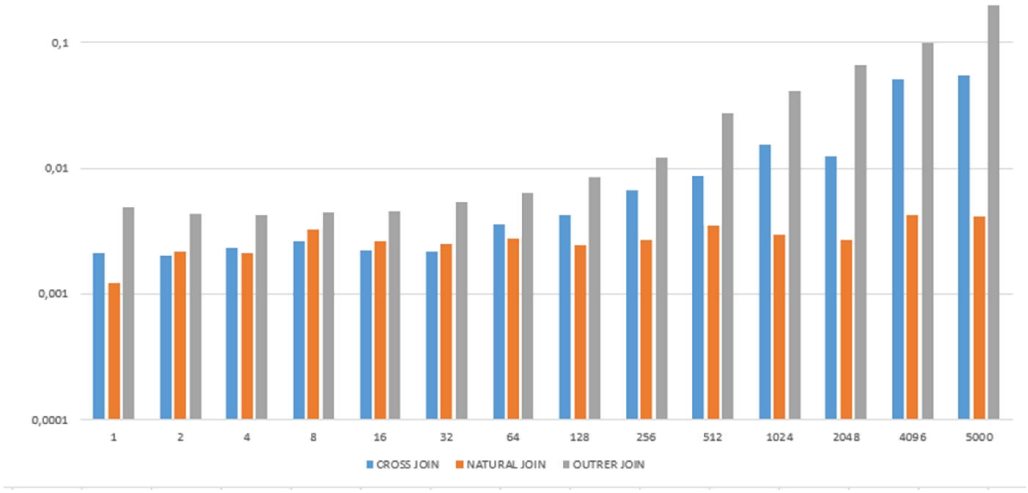
Gambar 5. Grafik Pengujian *Full Outer Join*

3.7. Hasil Perbandingan Pengujian

Dari hasil pengujian optimasi *query* yang telah dilakukan menggunakan *cross join, natural join*dan *full outer join*pada jumlah baris data 1sampai dengan 5000 baris data digambarkan pada gambar diagram berikut ini [gambar 6]:

Tabel 9. Tabel Perbandingan Pengujian

No	Jumlah Data	Cross Join	Natural Join	Full Outer Join
1	1	0.00213	0.00121	0.00492
2	2	0.00203	0.00214	0.00431
3	4	0.00233	0.00212	0.00418
4	8	0.00259	0.00327	0.00439
5	16	0.00223	0.00261	0.00450
6	32	0.00214	0.00249	0.00533
7	64	0.00355	0.00273	0.00642
8	128	0.00426	0.00245	0.00844
9	256	0.00667	0.00265	0.01214
10	512	0.00869	0.00345	0.02736
11	1024	0.01546	0.00296	0.04111
12	2048	0.01247	0.00270	0.06697
13	4096	0.05153	0.00421	0.09907
14	5000	0.05531	0.00409	0.20028



Gambar 6. Diagram Perbandingan Hasil Pengujian

4. KESIMPULAN

Dari hasil pengujian optimasi *query* menggunakan *cross join, natural join dan full outer join*yang dilakukan pada pengujian jumlah data 1, 2, 4, 8,16, 32, 64, 128, 256, 512, 1024, 2048, 4096 dan 5000 jumlah baris data. Berdasarkan hasil pengujian pada jumlah data 1, 4, 64, 128, 256, 512, 1024, 2048, 4096, dan 5000 baris data hasil *respon time natural join* lebih cepat jika dibandingkan dengan *cross join* dan *full outer join*, namun pada pengujian jumlah data 2, 8, 16, dan 32 *cross join* lebih unggul dibandingkan dari *natural join* dan *full outer join*, sedangkan pengujian menggunakan *full outer join* hasil perbandingan *respon time*terus mengalami kenakikan dan relatif lebih lama dibandingkan dengan *cross join* maupun *natural join*.

5. SARAN

Mengelola database dengan jumlah data yang banyak memerlukan kerja ekstra dan memakan waktu yang lama agar dapat mempermudah pengelolaan data dan pekerjaan menjadi lebih efektif dan efisien dibutuhkannya teknik optimasi *query* yang unggul semoga pada penelitian – penelitian berikutnya dapat mengembangkan teknik optimasi *query* yang unggul dan lebih banyak lagi penelitian tentang optimasi *query*.

DAFTAR PUSTAKA

[1]Gat. 2015. Perancangan Basis Data Perputakaan Sekolah dengan Menerapkan Model Data Relasional.Citec Journal, Vol. 2, No. 4

[2]Sinuraya, Junus. 2013 Analisis *Query* Pencarian Data Menggunakan Algortima Hash Join Dan Nested Join. Medan: Universitas Sumatera Utara

[3]Saputra, Des Julius 2012, Pengoptimasian Data Menggunakan Subset *Query*. Palembang, Universitas Binadarma

[4]Berry Maulani, A.haidar Mirza, Maria Ulfa. Analisis Perbandingan Optimasi *Query* Nested Join Dan Hash Join Pada Aplikasi Pencarian Data Berbasis Web.

[5]Ermatita. 2009.AnalisisOptimasi *Query* Pada Data Mining.Jurnal Sistem Informasi (JSI), VOL. 1, NO. 1.

[6]Eko Darmanto. 2015. Analisa Optimasi Bahasa SQL Berdasarkan Relational Algebra Pada Kasus Rekapitulasi Mahasiswa Layak Wisuda