Analisis Perbandingan Optimasi Query Nasted Join dan Hash Join pada MySQL Server

Comparative Analysis and Optimization Query Nasted Join Hash Join in MySQL Server

Melany Mustika Dewi*[1], Novandi Rezeki[2]
[1,2]Magister Teknik Informatika STMIK AMIKOM Yogyakarta
E-mail: ***[1]melany.m@students.amikom.ac.id,[2]novandinovan@gmail.com**

Abstrak

Optimasi query dapat menjadi salah satu cara yang digunakan untuk melakukan proses pengolahan data dalam mencari jalur tercepat pada database menagement system (DBMS). Membandingkan beberapa teknik optimasi untuk mengetahui mana yang tercepat dalam melakukan proses eksekusi data pada kondisi tertentu. Dalam penelitian ini proses pengujian yang dilakukan adalah mengolah data dengan menerapkan teknik optimasi dengan Nested Join Query dan Hash Join Query yang merupakan dua query yang banyak digunakan dalam basis data. Masalah yang dibahas dalam penelitian ini adalah bagaimana membandingkan kecepatan pengolahan data dengan query yang telah dilakukan join relasi dengan beberapa tabel yang berbeda dan menggunakan dua teknik optimasi. Sehingga dapat diambil kesimpulan teknik apa yang paling cocok digunakan dalam proses sesuai dengan kondisi tertentu. Proses ini dilakukan menggunakan MySQL sebagai DBMS dan data yang akan diolah adalah database yang berisi data pegawai. Hasil akhir dari penelitian ini berbentuk grafik dengan hasil teknik optimasi query yang terbaik yaitu Nasted Join Scalar.

Kata Kunci*—Optimasi Query, Nasted Join, Hash Join*

Abstract

Query optimization can be a way to perform data processing in finding the fastest path on a wide database system process (DBMS). Compares to the several optimization techniques and figure out which one is the fastest in the execution data process in certain conditions. In this study, the process of testing is to manipulate data by applying optimization techniques with Nested Join and Hash Join Query which are two queries that is widely used in the database. The problem discussed in this research is how to compare the speed of processing data with queries that has been done by join partnership with several different tables and using two optimization techniques. So the conclusion is what technique is most suitable in the process depending on certain conditions. The process is done using MySQL as the DBMS and the data to be processed is a database that contains the data clerk. The result of this research is a graphic of the best query optimization techniques i.e. Nasted Join Scalar.

Keywords*— Query Optimation, Nasted Join, Hash Join*

1. PENDAHULUAN

Saat ini DBMS seperti Oracle, MySQL dan SQL Serverdigunakan untuk mengolah data. Untuk mempercepat dalam pengelohan datanya, dapat menggunakan teknik join query seperti Hash Join dan Nasted Join. Optimasi dalam DBMS dapat dilihat dari penggunaan waktu eksekusi yang dibutuhkan dalam memproses data yang disajikan ke dalam DBMS. Setiap teknik akan mengolah data dengan waktu yang berbeda.

Pada penelitian sebelumnya, optimasi dengan teknik nash join dan nasted join dilakukan pada aplikasi pencarian data berbasis web, sedangkan untuk melakukan penelitian berkaitan dengan

teknik optimasi hash join dan nasted join, kita dapat melakukannya secara manual dengan menggunakan SQLYog sebagai user interface. Dengan menggunakan teknik analisis, maka dapat diketahui kemampuan sebuah DBMS dalam mengolah data. [1]

Penelitian berikut akan menampilkan mana yang tercepat dari hasil eksekusi query yang menggunakan hash join dan nested join pada DBMS MySQL dalam bentuk diagram.

2. METODE PENELITIAN

2.1. Optimasi Query

Optimasi Query adalah suatu proses yang digunakan untuk menganalisa query dalam 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 tercepat dan mengoptimumkan penyimpanan data. [2]

2.2. Hash Join Query

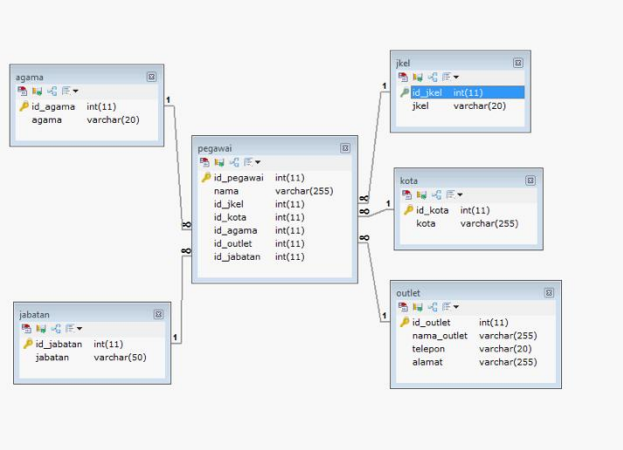
Hash join query digunakan untuk mengolah data yang berskala besar. Cara kerja hash join adalah membuat sebuah hash tabel berdasarkan predikat join. Setiap tabel yang memiliki inner atau outer dijadikan kode dangan hash fuction. Dari kode hash dari inner dan outer akan dibandingkan apabila kode kedua sama, maka dilakukan proses pengecekan nilai dalam kolom. Jika nilainya sama, maka akan dimasukkan kedalam hasil.

2.3. Nasted Join Query

Nasted join query adalah join yang digunakan ketika subset berjumlah sedikit. Cara kerja nasted join menentukan sebuah tabel yang dijadikan outer dan tabel lainnya sebagai inner. Setiap baris yang ada pada outer akan dilakukan join pada baris yang ada pada inner tabel.

2.4. Perancangan Database

Perancangan database adalah



Gambar 1. Perancangan Database

2.5. Query Algoritma

Penelitian ini di mulai dengan menginputkan data sebanyak 1.048.576 pada 6 tabel di database Pegawai. Selanjutnya menyiapkan empat query dengan relasi yang berbeda beda, dalam satu relasi terdapat tiga teknik join yang berbeda yaitu Hash Join Query, Nasted Join Query Scalar, dan Nasted Join Correlated. Dalam satu query dilakukan 24 kali pengujian dengan limit data 2n dan per limit data dieksekusi sebanyak 3 kali. Hasil waktu eksekusi dari tiga kali percobaan tersebut dibuat rata rata dengan cara dibagi tiga dan menghasilkan kecepatan rata rata dari eksekusi query tersebut.

Dalam pengujiannya, database yang telah disiapkan dibuat 4 buah query, yaitu Query 1 relasi, 2 relasi, 3 relasi, dan 4 relasi. Setiap query relasi tersebut di eksekusi menggunakan 3 teknik yang berbeda yaitu Hash Join Query, Nasted Join Query Scalar, dan Nasted Join Correlated.

Berikut adalah 4 buah query yang akan di eksekusi.

Query 1 adalah query yang join dengan 1 relasi.

```
/* Query 1 */

/*Hash Join Query :*/
Select pegawai.* from pegawai, outlet
where pegawai.id_outlet= outlet.
id_outlet limit x,x;

/*Nested Join Query Scalar :*/
select*from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet) limit x,x;

/*Nested Join Correlated :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet)  limit x,x;
```

Query 2 adalah query yang join dengan 2 relasi.

```
/* Query 2 */

/*Hash Join Query :*/
select pegawai.* from pegawai, outlet,
jabatan where pegawai.id_outlet=
outlet. id_outlet and
pegawai.id_jabatan = jabatan.id_jabatan
limit x,x;

/*Nested Join Query Scalar :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_jabatan in(select id_jabatan from
jabatan))
limit x,x;

/*Nested Join Correlated :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet and pegawai.id_jabatan
in(select id_jabatan
from jabatan where
pegawai.id_jabatan=jabatan.id_jabatan ))
limit x,x;
```

Query 3 adalah query yang join dengan 3 relasi.

```
/* Query 3 */
/*Hash Join Query :*/
select pegawai.* from pegawai, outlet,
jabatan, agama where pegawai.
id_outlet = outlet. id_outlet
and pegawai. id_jabatan = jabatan.
id_jabatan and pegawai.id_agama =
agama.id_agama
limit x,x;

/*Nested Join Query Scalar :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_jabatan in(select id_jabatan from
jabatan)) and pegawai.id_jkel in
(select id_jkel from jkel)
limit x,x;

/*Nested Join Correlated :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet
 and pegawai.id_jabatan in(select id_jabatan
from jabatan where
pegawai.id_jabatan=jabatan.id_jabatan ) and
pegawai.id_jkel in(select id_jkel
from jkel where
pegawai.id_jkel=jkel.id_jkel ))
limit x,x;
```

Query 4 adalah query yang join dengan 4 relasi.

```
/* Query 4 */

/*Hash Join Query : */
select pegawai.* from pegawai, outlet,
jabatan, klasifikasi, jenis where
pegawai.no_outlet =
outlet.no_outlet and
outlet.id_jabatan = jabatan.id_jabatan and
pegawai.id_jenis = jenis.id_jenis and
pegawai.no_klas = klasifikasi.no_klas;

/*Nested Join Query Scalar :*/
select * from pegawai where
pegawai.no_outlet in(select
no_outlet from outlet where
outlet.id_jabatan in(select id_jabatan from
jabatan)) and pegawai.id_jenis in
(select id_jenis from jenis) and
pegawai.no_klas in(select no_klas from
```

```
klasifikasi);

/*Nested Join Correlated :*/
select * from pegawai where
pegawai.no_outlet in (select
no_outlet from outlet where
pegawai.no_outlet
=outlet.no_outlet and
outlet.id_jabatan in (select id_jabatan
from jabatan where
outlet.id_jabatan=jabatan.id_jabatan))
and pegawai.id_jenis in (select
id_jenis from jenis where
pegawai.id_jenis=jenis.id_jenis) and
pegawai.no_klas in(select no_klas from
klasifikasi where pegawai.no_klas =
klasifikasi.no_klas ).
```

2.6. Proses Pengujian

Proses eksekusi query dilakukan dengan menggunakan laptop yang memiliki spesifikasi sebagai berikut :

Laptop : Acer Travelmate
Sistem Operasi : Windows 7 Ultimate
Processor : Intel(R) Core(TM)2 Duo CPU
RAM : 2.00 GB
System Type : 32-bit OS

Aplikasi yang digunakan adalah SQLyog sebagai Mysql user interface dan XAMPP. Dalam menjalakan prosesnya, maka ada beberapa tahap proses dalam menjalankan SQLyog. Yang pertama menjalankan XAMPP dengan menjalankan tombol start pada MySql. Setelah XAMPP sudah di jalankan, maka query yang sudah di masukkan jumlah recordnya dapat di eksekusi dan didapatkan informasi berkaitan dengan waktu yang diperlukan dalam setiap proses eksekusi.

3. HASIL DAN PEMBAHASAN

Dari setiap teknik optimasi dilakukan 3 kali pengujian. Dari waktu yang dihasilkan akan dicari kecepatan rata-rata eksekusi query. Angka pada limit query dibawah ditentukan sesuai dengan jumlah data yang akan di esksekusi seperti pada tabel.

Tabel 1. adalah waktu ekseskusi untuk query dengan satu relasi, tabel 2. hasil ekseskusi query dua relasi, tabel 3. hasil ekseskuis query tiga relasi dan tabel 4. hasil ekseskusi empat relasi. Berikut adalah hasil dari eksekusi query dengan berbagai limit dan relasi yang berbeda.

```
/* Query 1 */

/*Hash Join Query :*/
Select pegawai.* from pegawai, outlet
where pegawai.id_outlet= outlet.
id_outlet limit 1,1;

/*Nested Join Query Scalar :*/
select*from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet) limit 1,1;
```

```
/*Nested Join Correlated :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet)  limit 1,1;
```

Tabel 1. Hasil Eksekusi Query Relasi 1

No	Jumlah Data	Waktu											
		Hash Join Query				Nested Join Scalar				Nested Join Correlated			
		1	2	3	Rata-rata	1	2	3	Rata-rata	1	2	3	Rata-rata
1	1	0,171	0,172	0,156	0,166	0,141	0,125	0,156	0,141	0,14	0,156	0,171	0,156
2	2	0,203	0,172	0,11	0,162	0,203	0,328	0,187	0,239	0,156	0,125	0,187	0,156
3	4	0,172	0,218	0,172	0,187	0,125	0,124	0,172	0,140	0,125	0,141	0,156	0,141
4	8	0,14	0,203	0,156	0,166	0,11	0,156	0,188	0,151	0,203	0,203	0,187	0,198
5	16	0,218	0,188	0,141	0,182	0,14	0,172	0,124	0,145	0,203	0,234	0,187	0,208
6	32	0,14	0,109	0,141	0,130	0,14	0,125	0,156	0,140	0,141	0,218	0,125	0,161
7	64	0,187	0,125	0,14	0,151	0,202	0,125	0,109	0,145	0,281	0,219	0,187	0,229
8	128	0,172	0,187	0,187	0,182	0,203	0,203	0,14	0,182	0,141	0,14	0,156	0,146
9	256	0,14	0,141	0,14	0,140	0,109	0,125	0,125	0,120	0,187	0,172	0,172	0,177
10	512	0,156	0,14	0,172	0,156	0,156	0,156	0,156	0,156	0,172	0,188	0,156	0,172
11	1024	0,172	0,171	0,234	0,192	0,172	0,125	0,172	0,156	0,187	0,219	0,187	0,198
12	2048	0,187	0,109	0,187	0,161	0,141	0,172	0,124	0,146	0,234	0,219	0,156	0,203
13	4096	0,125	0,172	0,172	0,156	0,203	0,203	0,187	0,198	0,14	0,156	0,171	0,156
14	8192	0,156	0,171	0,109	0,145	0,141	0,203	0,203	0,182	0,234	0,125	0,188	0,182
15	16384	0,14	0,171	0,188	0,166	0,172	0,171	0,125	0,156	0,156	0,14	0,187	0,161
16	32768	0,156	0,141	0,188	0,162	0,14	0,109	0,156	0,135	0,125	0,156	0,203	0,161
17	65536	0,125	0,156	0,14	0,140	0,187	0,202	0,125	0,171	0,202	0,218	0,14	0,187
18	131072	0,156	0,125	0,124	0,135	0,187	0,25	0,109	0,182	0,171	0,125	0,156	0,151
19	262144	0,187	0,125	0,172	0,161	0,171	0,156	0,187	0,171	0,156	0,203	0,156	0,172
20	524288	0,156	0,125	0,171	0,151	0,156	0,172	0,141	0,156	0,203	0,202	0,218	0,208
21	1048576	0,141	0,129	0,218	0,163	0,218	0,234	0,188	0,213	0,124	0,171	0,172	0,156

```
/* Query 2 */
/*Hash Join Query :*/
select pegawai.* from pegawai, outlet,
jabatan where pegawai.id_outlet=
outlet. id_outlet and
pegawai.id_jabatan = jabatan.id_jabatan
limit 1,1;

/*Nested Join Query Scalar :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_jabatan in(select id_jabatan from
jabatan))
limit 1,1;
```

```
/*Nested Join Correlated :*/
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet and pegawai.id_jabatan
in(select id_jabatan
from jabatan where
pegawai.id_jabatan=jabatan.id_jabatan ))
limit 1,1;
```

Tabel 2.Tabel Hasil Eksekusi Query Relasi 2

No	Jumlah Data	Waktu											
		Hash Join Query				Nested Join Scalar				Nested Join Correlated			
		1	2	3	Rata-rata	1	2	3	Rata-rata	1	2	3	Rata-rata
1	1	0,187	0,172	0,156	0,172	0,125	0,218	0,219	0,187	0,156	0,171	0,156	0,161
2	2	0,156	0,125	0,265	0,182	0,281	0,188	0,219	0,229	0,188	0,203	0,297	0,229
3	4	0,125	0,141	0,234	0,167	0,140	0,187	0,256	0,194	0,156	0,187	0,203	0,182
4	8	0,187	0,156	0,171	0,171	0,172	0,172	0,218	0,187	0,140	0,172	0,156	0,156
5	16	0,187	0,140	0,202	0,176	0,110	0,109	0,188	0,136	0,187	0,156	0,219	0,187
6	32	0,125	0,203	0,109	0,146	0,188	0,209	0,312	0,236	0,203	0,281	0,287	0,257
7	64	0,141	0,125	0,187	0,151	0,234	0,125	0,124	0,161	0,187	0,265	0,250	0,234
8	128	0,109	0,187	0,187	0,161	0,203	0,156	0,171	0,177	0,218	0,188	0,156	0,187
9	256	0,249	0,171	0,187	0,202	0,141	0,156	0,171	0,156	0,250	0,156	0,202	0,203
10	512	0,187	0,219	0,265	0,224	0,266	0,109	0,124	0,166	0,171	0,188	0,202	0,187
11	1024	0,156	0,203	0,234	0,198	0,172	0,171	0,125	0,156	0,125	0,172	0,156	0,151
12	2048	0,124	0,203	0,172	0,166	0,187	0,125	0,125	0,146	0,140	0,172	0,219	0,177
13	4096	0,156	0,187	0,172	0,172	0,187	0,187	0,124	0,166	0,234	0,140	0,219	0,198
14	8192	0,203	0,172	0,202	0,192	0,203	0,156	0,187	0,182	0,172	0,218	0,203	0,198
15	16384	0,202	0,188	0,203	0,198	0,187	0,125	0,156	0,156	0,203	0,171	0,188	0,187
16	32768	0,125	0,141	0,218	0,161	0,187	0,110	0,171	0,156	0,187	0,109	0,141	0,146
17	65536	0,187	0,172	0,219	0,193	0,171	0,234	0,140	0,182	0,203	0,140	0,156	0,166
18	131072	0,218	0,187	0,187	0,197	0,156	0,156	0,187	0,166	0,203	0,219	0,171	0,198
19	262144	0,140	0,187	0,186	0,171	0,249	0,140	0,140	0,176	0,203	0,187	0,156	0,182
20	524288	0,187	0,188	0,203	0,193	0,172	0,140	0,141	0,151	0,250	0,171	0,141	0,187
21	1048576	0,171	0,234	0,172	0,192	0,156	0,203	0,218	0,192	0,172	0,219	0,140	0,177

```
/* Query 3 */
/*Hash Join Query :*/
select pegawai.* from pegawai, outlet,
jabatan, agama where pegawai.
id_outlet = outlet. id_outlet
and pegawai. id_jabatan = jabatan.
id_jabatan and pegawai.id_agama =
agama.id_agama
limit 1,1;

/*Nested Join Query Scalar :*/
select * from pegawai where
pegawai.id_outlet in(select
```

```
id_outlet from outlet where
pegawai.id_jabatan in(select id_jabatan from
jabatan)) and pegawai.id_jkel in
(select id_jkel from jkel)
limit 1,1;
```

/*Nested Join Correlated :*/

```
select * from pegawai where
pegawai.id_outlet in(select
id_outlet from outlet where
pegawai.id_outlet=outlet.id_outlet
 and pegawai.id_jabatan in(select id_jabatan
from jabatan where
pegawai.id_jabatan=jabatan.id_jabatan ) and
pegawai.id_jkel in(select id_jkel
from jkel where
pegawai.id_jkel=jkel.id_jkel ))
limit 1,1;
```

Tabel 3.Tabel Hasil Eksekusi Query Relasi 3

No	Jumlah Data	Waktu											
		Hash Join Query				Nested Join Scalar				Nested Join Correlated			
		1	2	3	Rata-rata	1	2	3	Rata-rata	1	2	3	Rata-rata
1	1	0,171	0,187	0,234	0,197	0,203	0,203	0,125	0,177	0,171	0,187	0,171	0,176
2	2	0,234	0,156	0,125	0,172	0,187	0,188	0,14	0,172	0,203	0,171	0,188	0,187
3	4	0,172	0,187	0,171	0,177	0,218	0,188	0,14	0,182	0,125	0,188	0,187	0,167
4	8	0,202	0,172	0,187	0,187	0,171	0,172	0,218	0,187	0,156	0,171	0,172	0,166
5	16	0,171	0,187	0,156	0,171	0,171	0,14	0,125	0,145	0,156	0,203	0,124	0,161
6	32	0,156	0,187	0,203	0,182	0,156	0,172	0,125	0,151	0,187	0,11	0,14	0,146
7	64	0,234	0,218	0,203	0,218	0,172	0,109	0,172	0,151	0,187	0,234	0,141	0,187
8	128	0,14	0,171	0,141	0,151	0,14	0,219	0,172	0,177	0,202	0,187	0,218	0,202
9	256	0,172	0,156	0,125	0,151	0,187	0,234	0,171	0,197	0,187	0,156	0,218	0,187
10	512	0,141	0,218	0,156	0,172	0,187	0,187	0,156	0,177	0,171	0,156	0,14	0,156
11	1024	0,14	0,141	0,14	0,140	0,171	0,202	0,156	0,176	0,141	0,203	0,141	0,162
12	2048	0,171	0,187	0,14	0,166	0,141	0,202	0,218	0,187	0,156	0,141	0,312	0,203
13	4096	0,203	0,234	0,219	0,219	0,187	0,141	0,187	0,172	0,125	0,265	0,156	0,182
14	8192	0,187	0,203	0,141	0,177	0,187	0,156	0,202	0,182	0,187	0,14	0,188	0,172
15	16384	0,125	0,234	0,203	0,187	0,156	0,25	0,218	0,208	0,125	0,187	0,14	0,151
16	32768	0,172	0,124	0,187	0,161	0,156	0,188	0,187	0,177	0,249	0,187	0,171	0,202
17	65536	0,156	0,187	0,14	0,161	0,125	0,171	203	67,765	0,14	0,109	0,171	0,140
18	131072	0,172	0,187	0,125	0,161	0,14	0,25	0,203	0,198	0,171	0,172	0,14	0,161
19	262144	0,156	0,171	0,203	0,177	0,187	0,14	0,219	0,182	0,218	0,187	0,172	0,192
20	524288	0,156	0,203	0,141	0,167	0,187	0,202	0,172	0,187	0,156	0,297	0,203	0,219
21	1048576	0,109	0,203	0,156	0,156	0,202	0,109	0,172	0,161	0,203	0,234	0,141	0,193

```
/* Query 4 */
```

/*Hash Join Query : */

```
select pegawai.* from pegawai, outlet,
```

```
jabatan, klasifikasi, jenis where
pegawai.no_outlet =
outlet.no_outlet and
outlet.id_jabatan = jabatan.id_jabatan and
pegawai.id_jenis = jenis.id_jenis and
pegawai.no_klas = klasifikasi.no_klas;
```

/*Nested Join Query Scalar :*/

```
select * from pegawai where
pegawai.no_outlet in(select
no_outlet from outlet where
outlet.id_jabatan in(select id_jabatan from
jabatan)) and pegawai.id_jenis in
(select id_jenis from jenis) and
pegawai.no_klas in(select no_klas from
klasifikasi);
```

/*Nested Join Correlated :*/

```
select * from pegawai where
pegawai.no_outlet in (select
no_outlet from outlet where
pegawai.no_outlet
=outlet.no_outlet and
outlet.id_jabatan in (select id_jabatan
from jabatan where
outlet.id_jabatan=jabatan.id_jabatan))
and pegawai.id_jenis in (select
id_jenis from jenis where
pegawai.id_jenis=jenis.id_jenis) and
pegawai.no_klas in(select no_klas from
klasifikasi where pegawai.no_klas =
klasifikasi.no_klas )
```

Tabel 4.Tabel Hasil Eksekusi Query Relasi 4

No	Jumlah Data	Waktu											
		Hash Join Query				Nested Join Scalar				Nested Join Correlated			
		1	2	3	Rata-rata	1	2	3	Rata-rata	1	2	3	Rata-rata
1	1	0,124	0,187	0,140	0,150	0,156	0,171	0,203	0,177	0,140	0,234	0,187	0,187
2	2	0,171	0,141	0,172	0,161	0,172	0,140	0,140	0,151	0,156	0,156	0,172	0,161
3	4	0,110	0,125	0,110	0,115	0,156	0,156	0,141	0,151	0,156	0,140	0,172	0,156
4	8	0,140	0,171	0,187	0,166	0,140	0,188	0,141	0,156	0,187	0,171	0,125	0,161
5	16	0,124	0,172	0,171	0,156	0,171	0,156	0,187	0,171	0,202	0,125	0,171	0,166
6	32	0,203	0,124	0,203	0,177	0,187	0,187	0,140	0,171	0,219	0,187	0,187	0,198
7	64	0,187	0,140	0,188	0,172	0,218	0,172	0,141	0,177	0,187	0,156	0,156	0,166
8	128	0,156	0,172	0,124	0,151	0,203	0,250	0,234	0,229	0,187	0,156	0,234	0,192
9	256	0,203	0,234	0,187	0,208	0,203	0,187	0,156	0,182	0,203	0,187	0,140	0,177
10	512	0,171	0,219	0,172	0,187	0,109	0,171	0,187	0,156	0,187	0,171	0,172	0,177
11	1024	0,256	0,202	0,140	0,199	0,172	0,187	0,172	0,177	0,203	0,125	0,203	0,177
12	2048	0,172	0,141	0,218	0,177	0,220	0,187	0,234	0,214	0,266	0,234	0,156	0,219
13	4096	0,234	0,187	0,203	0,208	0,140	0,187	0,265	0,197	0,125	0,203	0,202	0,177
14	8192	0,188	0,203	0,172	0,188	0,188	0,156	0,141	0,162	0,188	0,172	0,171	0,177
15	16384	0,172	0,171	0,141	0,161	0,156	0,188	0,171	0,172	0,218	0,187	0,187	0,197
16	32768	0,156	0,171	0,109	0,145	0,203	0,125	0,125	0,151	0,203	0,219	0,141	0,188
17	65536	0,187	0,187	0,219	0,198	0,203	0,203	0,171	0,192	0,187	0,156	0,203	0,182
18	131072	0,234	0,203	0,202	0,213	0,140	0,125	0,172	0,146	0,218	0,202	0,218	0,213
19	262144	0,203	0,218	0,202	0,208	0,125	0,203	0,125	0,151	0,187	0,188	0,140	0,172
20	524288	0,218	0,140	0,187	0,182	0,219	0,188	0,125	0,177	0,140	0,187	0,187	0,171
21	1048576	0,172	0,125	0,140	0,146	0,140	0,187	0,249	0,192	0,280	0,141	0,156	0,192

4. KESIMPULAN

Dari hasil pengujian 4 query relasi, dapat disimpulkan bahwa query yang memiliki kecepatan eksekusi paling cepat adalah nasted join scalar dengan rata-rata kecepatan 0,172 per second.

Gambar 2. Hasil Akhir Penelitian

Kelebihan dari penelitian ini adalah, pengujian dilakukan dengan menggunakan software XAMPP dan SQLyog sehingga kita memahami alur kerja proses optimasi query dan cara perhitungan eksekusinya. Sedangkan kelemahan dari penelitian ini adalah proses penginputan menggunakan metode pengujian secara manual, sehingga untuk melakukan pengujiannya membutuhkan waktu yang cukup lama dan proses penginputan datanya juga lebih rentan kesalahan.

5. SARAN

Untuk penelitian berikutnya disarankan untuk membuat aplikasi agar bisa memudahkan dalam mengtahui kecepatan eksekusi optimasi query.Membandingkan optimasi query dengan algoritma lain.

DAFTAR PUSTAKA

[1] Berry M, A. Haidar M, Maria U, 2015, Analisis Perbandingan Optimasi Query Nested Join Dan Hash Join Pada Aplikasi Pencarian Data Berbasis Web, http://digilib.binadarma.ac.id/files/disk1/144/123-123-berrymaula-7151-1-jurnal_p-n.pdf, diakses 27 Desember 2016

[2] Junus Sinuraya, 2013, Analisis Query Pencarian Data Menggunakan Algortima Hash Join dan Nested Join, blob:https://web.whatsapp.com/dc575335-cf8d-4a6a-92c9-97b9ff9c4082 diakses 27 Desember 2016