PERBANDINGAN OPTIMASI QUERY MENGGUNAKAN QUERY SCALAR, CORRELATED DAN KOMBINASI

Jajang Nurjaman (za2ng2509@gmail.com)
Wahyuni (unniecandoit@gmail.com**)**
Fakultas Pascasarjana Teknik Informatika,
Universitas AMIKOM

ABSTRAK

Optimasi query merupakan suatu pola penulisan SQL yang mengacu pada standar SQL. Optimasi query ini sangat penting untuk dapat dipelajari karena dengan optimasi query ini kita dapat memaksimalkan kecepatan dan efisiensi dari eksekusi suatu program. Ada dua jenis pendekatan yang umum pada optimasi query ini yaitu pendekatan heuristic dan cost-base.dalam pendekatan cost base terdapat dua model optimasi query yaitu cross product dan subquery. Pada subquery sendiri terdapat tiga jenis query yaitu scalar, correlated dan kombinasi (cross product-scalar dan multi scalar). Ketiga jenis query dari subquery ini akan dibandingkan melalui studi kasus dimana pada studi kasus ini melibatkan banyak data dan relasi antar tabel. Dari hasil pengujian dapat diketahui bahwa jenis query scalar dan correlated adalah jenis query yang tepat untuk digunakan dalam jumlah data dan relasi antar tabel yang relatif sedikit. Sedangkan query multi scalar adalah jenis query yang paling tepat digunakan untuk mengeksekusi jumlah data yang banyak dan melibatkan relasi antar tabel yang banyak pula. Sebaliknya, jenis query yang paling tidak dianjurkan untuk digunakan adalah jenis query cross product-scalar.

Kata Kunci: optimasi query, scalar, correlated, cross product-scalar, multi scalar.

1. PENDAHULUAN

.Dalam mengelola sebuah database, dibutuhkan suatu DBMS yang merupakan jembatan dari user ke database tersebut. Untuk melakukan pengaksesan basis data diperlukan suatu bahasa manipulasi data yang biasa disebut SQL. Dengan SQL kita dapat membuat database, menambah, menghapus mengubah dan mencari suatu data. Namun permasalahannya adalah jika kita menggunakan bahasa tersebut tanpa mengikuti standar yang berlaku maka yang terjadi adalah waktu yang dibutuhkan untuk mengeksekusi query tersebut tidak optimal. Agar kecepatan dan efisiensi dari eksekusi program dapat berjalan dengan maksimal maka dibutuhkan suatu pola penulisan yang sesuai dengan standar SQL atau bisa disebut dengan optimasi query. Teknik optimasi query dapat dilakukan dengan berbagai cara dan pendekatan yaitu pendekatan secara *heuristic* dan *cost-based*. Dalam pendekatan *cost-based* terdiri dari tiga jenis bentuk query yaitu *scalar*, *correlated* dan kombinasi (*cost-product-scalar, dan multiscalar)*. Dari ketiga jenis bentuk query tersebut akan dibandingkan manakah jenis query yang paling optimal untuk digunakan dalam jumlah data yang kecil, sedang dan besar.

2. DASAR TEORI

A. Database Management System(DBMS)

DBMS adalah *software* yang menangani semua akses ke basis data. Secara konsep apa yang telah terjadi adalah sebagai berikut (Kusrini, 2007):

a. User melakukan pengaksesan basis data untuk informasi yang diperlukannya menggunakan suatu bahasa manipulasi data, biasanya disebut SQL.
b. DBMS menerima *request* dari user dan menganalisa *request* tersebut.

c. DBMS memeriksa skema eksternal user, pemetaan eksternal/konseptual, skema konseptual, pemetaan konseptual/internal, dan struktur penyimpanan.
d. DBMS mengeksekusi operasi-operasi yang diperlukan untuk memenuhi permintaan user.

Contoh dari DBMS ini yaitu antara lain Microsoft SQL Server 2000, Oracle, MySQL, Interbase, Microsoft Access, dan lain-lain.

B. Structured Query Language (SQL)

SQL adalah salah satu dari sekian banyak bahasa pemrograman database yang paling popular dan menjadi standar perintah database (Wahana Komputer, 2010). SQL merupakan bahasa pemrograman yang banyak digunakan dalam aplikasi database seperti database JavaDB(Derby), MySQL, Postgree, MsSQL, dan lain sebagainya.

Dengan bahasa SQL memungkinkan kita membuat database, menambah, menghapus, mengubah dan mencari data.

C. Optimasi Query

Optimasi query adalah suatu pola penulisan SQL yang mengacu pada standar SQL yang ditujukan agar dapat memaksimalkan kecepatan dan efisiensi dari eksekusi program (Tomy, 2008).

Teknik optimasi dapat dilakukan dengan beberapa cara. Terdapat dua pendekatan optimasi yang umum digunakan sebagaimana diungkapkan oleh Chanowich dalam Ermatita (2009), yakni:

a. Heuristik atau rule-based

Teknik ini mengaplikasikan aturan heuristik untuk mempercepat proses query. Optimasi jenis ini mentransformasikan query dengan sejumlah aturan yang akan meningkatkan kinerja eksekusi, yakni:

1. Melakukan operasi *selection* di awal untuk mereduksi jumlah baris.
2. Melakukan operasi *projection* di awal untuk mereduksi jumlah atribut.
3. Mengkonversikan query dengan banyak join menjadi query dengan banyak subquery.
4. Melakukan operasi *selection* dan *join* yang paling kecil keluarannya sebelum operasi lain.

b. Cost-based

Teknik ini mengoptimasikan *cost* yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang menjadi *cost* terendah. Teknik ini mengoptimalkan urutan join terbalik yang dimungkinkan pada relasi-relasi r1 □ r2 □ ... rn. Teknik ini dipergunakan untuk mendapatkan pohon *left-deep join* yang akan menghasilkan sebuah relasi sebenarnya pada node sebelah kanan yang bukan hasil dari sebuah *intermediate join.*

Komponen-komponen harga yang digunakan untuk mengeksekusi query adalah (Santiputri & Kirana & Anni, 2010):

a. *Access cost* untuk *secondary storage*

Harga ini adalah harga untuk pencarian, pembacaan dan penulisan blok-blok data yang terletak pada secondary storage, terutama pada disk. Harga dari pencarian untuk record-record dalam sebuah file tergantung pada tipe dari bentuk-bentuk akses pada file tersebut, seperti pengurutan (*ordering*), *hashing* dan pengindeksan *primary* ataupun secondary. Sebagai tambahan, faktor-faktor seperti disediakan atau tidaknya blok-blok file yang berdekatan pada silinder disk yang sama atau tersebar pada disk juga dapat mempengaruhi harga akses.

b. *Storage cost*

Harga ini adalah harga dari penyimpanan file-file menengah yang dihasilkan oleh sebuah strategi eksekusi untuk query.

c. *Computation cost*

Harga ini adalah harga dari pelaksanaan operasi-operasi memori pada buffer data selama eksekusi query. Seperti operasi-operasi pencarian dan pengurutan record, penggabungan record untuk sebuah join dan melakukan perhitungan-perhitungan pada nilai-nilai field.

d. *Memory usage cost*

Harga ini adalah harga mengenai jumlah dari buffer memori yang diperlukan selama eksekusi query.

e. *Communication cost*

Harga ini adalah harga dari pengiriman query dan hasilnya dari tempat basisdata atau terminal dimana query berasal. Untuk melihat kinerja sistem basis data dalam pencarian data, ada 2 model query yang akan digunakan, yaitu *cross product* dan *subset query.*

D. Subset Query

Subset query dibagi atas tiga macam bentuk query, antara lain sebagai berikut (Santiputri & Kirana & Anni, 2010):

1. Scalar

Yaitu melakukan select sejumlah kolom dari satu tabel dimana kondisi suatu kolom terpenuhi pada satu sub query. Scalar query diwakili oleh query berikut:

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1]
where [nama_tabel1].[nama_kolom1] in
(select[nama_kolom1]from [nama_tabel2])
```

2. Correlated

Yaitu melakukan select sejumlah kolom pada satu tabel dimana kondisi suatu kolom terpenuhi pada satu sub query, dan sub query tersebut pengkondisiannya masih berhubungan dengan super query. Correlated query diwakili oleh query berikut:

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1]
where [nama_tabel1].[nama_kolom1] in
(select [nama_kolom1] from [nama_tabel2]
where [nama_tabel1].[nama_kolom2] =[nama_tabel2].[nama_kolom2])
```

3. Kombinasi

Bentuk kombinasi ini adalah mengabungkan bentuk cross product dengan subset query. Kombinasi tersebut adalah:

• Cross Product dengan Scalar

Melakukan select sejumlah kolom pada beberapa tabel dimana pengkondisian antar tabel menggunakan join dan pengkondisian suatu kolom dipenuhi oleh satu sub query. Query tersebut diwakili oleh:

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1], [nama_tabel2]
where [nama_tabel1].[nama_kolom1]
=[nama_tabel2].[nama_kolom2]
AND [nama_tabel1].[nama_kolumn1] in
(select [nama_kolom1]
from [nama_tabel3])
```

• Multi Scalar

Melakukan select sejumlah kolom pada satu tabel dimana kondisi suatu kolom terpenuhi pada satu sub query dan sub query tersebut terdapat satu sub query lagi. Query ini diwakili oleh:

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1]
where [nama_tabel1].[nama_kolom1] in
(select [nama_kolom1] from [nama_tabel2]
where [nama_tabel2].[nama_kolom2] in
(select [nama_kolom2]
from [nama_tabel3]))
```

3. PENGUJIAN

A. Batasan Pengujian

Adapun batasan pengujian pada penelitian ini adalah sebagai berikut:

- Query yang digunakan untuk pengujian adalah model *subset query* yaitu jenis *scalar, correlated* dan kombinasi (*cross product-scalar* dan *multi scalar*).
- Melakukan pengujian dengan menggunakan 1 relasi sampai dengan 5 relasi dimana hasil yang didapat melalui ketiga jenis query tersebut adalah sama.
- Yang menjadi parameter dalam membandingkan ketiga jenis query tersebut adalah jumlah waktu yang dibutuhkan untuk mengeksekusi query tersebut.
- Jumlah data yang diuji bertahap dari 100 data, sampai dengan 1.000.000 data.
- DBMS yang digunakan adalah SQL server.

B. Basis Data Yang Digunakan

Basis data yang digunakan adalah *database* demo SQL Server yang dibagikan secara bebas untuk keperluan pembelajaran, basis data tersebut penulis dapatkan dari https://www.sqlskills.com/resources/conferences/creditbackup80.zip, *sqlskills.com* merupakan sebuah web yang menyediakan jasa *training* dan *consulting* khusus DBMS SQL Server.

Nama basis data adalah *Credit* yang terdiri dari 10 table yang saling berelasi. Adapun jumlah data pada tiap Table adalah sebagai berikut :

Tabel 1. Spesifikasi Basis Data

No	Nama Table	Jumlah Baris
1	Category	10
2	Charge	1.600.000
3	Corporation	500
4	Member	1.000
5	Member2	1.000
6	Payment	15.554
7	Provider	500
8	Region	9
9	Statement	2.000
10	Status	1

C. Skenario Pengujian

Skenario pengujian yang dilakukan adalah sebagai berikut:

- Hal yang dilakukan pertama kali adalah menentukan jumlah data yang akan diuji. Pada penelitian kali ini telah ditentukan jumlah data yang akan di uji yaitu 100,1000,10.000,100.000 dan 1.000.000 data.
- Mengeksekusi 4 macam query untuk tiap-tiap jumlah data yang sudah ditentukan di atas. Sehingga sebanyak 100 query yang akan dieksekusi.
- Membandingkan jumlah waktu pengeksekusian masing-masing query untuk masing-masing data.
- Setelah itu dapat dilihat query jenis apa yang lebih optimal untuk digunakan pada pencarian data dalam jumlah kecil dan besar.

4. HASIL DAN PEMBAHASAN

A. Query Yang Digunakan

a. Query yang digunakan pada pencarian data dengan relasi dua tabel adalah sebagai berikut:

```
    --Query Scalar
Select charge.*
from charge
where charge.member_no
in(select member_no from member    )
    --Query Corelate
select charge.*
from charge
where charge.member_no
in(select member_no from member where charge.member_no=member.member_no)
```

b. Query yang digunakan pada pencarian data dengan relasi tiga tabel adalah sebagai berikut:

```
    --Query Scalar
select charge.*
from charge, member
where charge.member_no in (select member_no from member)
and member.corp_no in (select corp_no from corporation)

    --Query Corelate
select charge.*
from charge, member
where charge.member_no in (select member_no from member where
charge.member_no=member.member_no) and
member.corp_no in(select corp_no from corporation where
member.corp_no=corporation.corp_no)

    --Query cross-product dan scalar
select charge.*
from charge,member
where charge.member_no=member.member_no and member.corp_no in
(select corp_no from corporation)

    --Query multi scalar
select charge.*
from charge
where charge.member_no in (select member_no from member where member.corp_no in
(select corp_no from corporation))
```

c. Query yang digunakan pada pencarian data dengan relasi empat tabel adalah sebagai berikut:

```
    --Query Scalar
select charge.*
from charge, member,corporation
where charge.member_no in (select member_no from member)
and member.corp_no in (select corp_no from corporation)
and corporation.region_no in (select region_no from region)
```

```
    --Query Corelate
select charge.*
from charge, member, corporation
where charge.member_no in
(select member_no from member where charge.member_no=member.member_no) and
member.corp_no in
(select corp_no from corporation where member.corp_no=corporation.corp_no) and
corporation.region_no in
(select region_no from region where corporation.region_no=region.region_no)

    --Query cross-product dan scalar
select charge.*
from charge,member,corporation
where charge.member_no=member.member_no and
member.corp_no=corporation.corp_no and
corporation.region_no in
(select region_no from region)


    --Query multi scalar
select charge.*
from charge
where charge.member_no in
(select member_no from member where member.corp_no in
(select corp_no from corporation where corporation.region_no in
(select region_no from region)))
```

d. Query yang digunakan pada pencarian data dengan relasi lima tabel adalah sebagai berikut:

```
      --Query Scalar
select charge.*
from charge, member,corporation
where charge.member_no in (select member_no from member)
and member.corp_no in (select corp_no from corporation)
and corporation.region_no in (select region_no from region)
and member.region_no in (select region_no from region)

      --Query Corelate
select charge.*
from charge, member, corporation
where charge.member_no in
(select member_no from member where charge.member_no=member.member_no) and
member.corp_no in
(select corp_no from corporation where member.corp_no=corporation.corp_no) and
corporation.region_no in
(select region_no from region where corporation.region_no=region.region_no and
member.region_no=region.region_no)

      --Query cross-product dan scalar
```

```
select charge.*
from charge,member,corporation
where charge.member_no=member.member_no and
member.corp_no=corporation.corp_no and
corporation.region_no in
(select region_no from region) and
member.region_no in (select region_no from region)

    --Query multi scalar
select charge.*
from charge
where charge.member_no in
(select member_no from member where member.corp_no in
(select corp_no from corporation where corporation.region_no in
(select region_no from region)) and member.region_no in (select region_no from region))
```

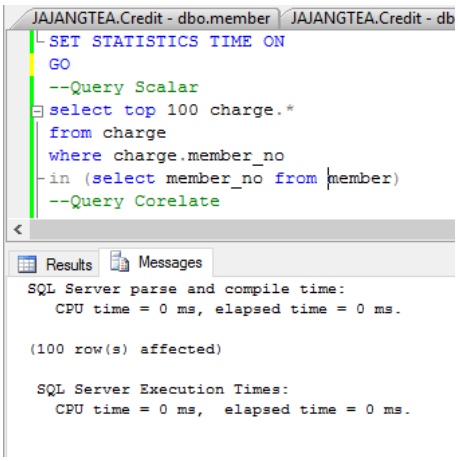
e. Query yang digunakan pada pencarian data dengan relasi lima tabel adalah sebagai berikut:

```
      --Query Scalar
select charge.*
from charge, member,corporation,provider
where charge.member_no in (select member_no from member)
and member.corp_no in (select corp_no from corporation)
and corporation.region_no in (select region_no from region)
and member.region_no in (select region_no from region)
and provider.region_no in(select region_no from region)

      --Query Corelate
select charge.*
from charge, member, corporation,provider
where charge.member_no in
(select member_no from member where charge.member_no=member.member_no) and
member.corp_no in
(select corp_no from corporation where member.corp_no=corporation.corp_no) and
corporation.region_no in
(select region_no from region where corporation.region_no=region.region_no and
member.region_no=region.region_no and provider.region_no=region.region_no)

      --Query cross-product dan select charge.*
from charge,member,corporation,provider
where charge.member_no=member.member_no and
member.corp_no=corporation.corp_no and
corporation.region_no in
(select region_no from region) and
member.region_no in (select region_no from region) and
provider.region_no in (select region_no from region)

    --Query multi scalar
select charge.*
from charge,provider
```

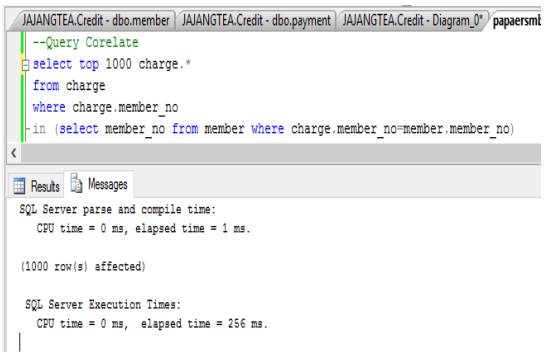
where charge.member_no in
(select member_no from member where member.corp_no in
(select corp_no from corporation where corporation.region_no in
(select region_no from region)) and member.region_no in (select region_no from region))and provider.region_no in (select region_no from region)

B. Tampilan User Interface

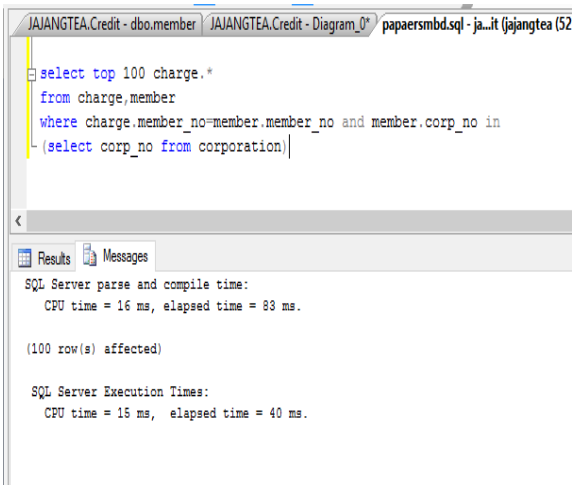
Adapun contoh tampilan user interface pada pengeksekusian beberapa query adalah sebagai berikut:



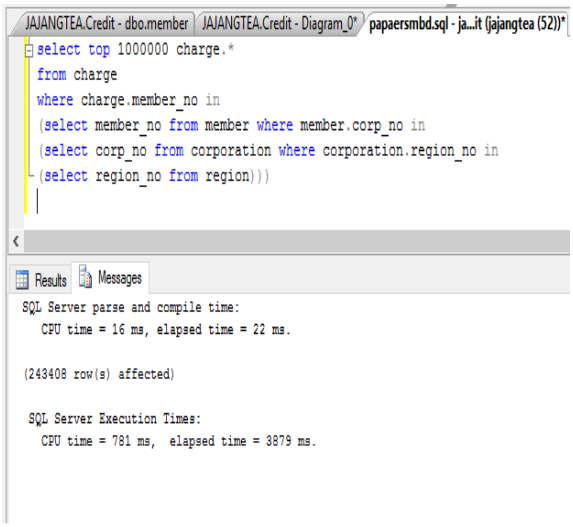
Gambar 1. Contoh Pengeksekusian query scalar dengan jumlah data 100.



Gambar 2. Contoh Pengeksekusian query correlated dengan jumlah data 1000.



Gambar 3. Contoh Pengeksekusian query cross-product-scalar dengan jumlah data 100.



Gambar 4. Contoh Pengeksekusian query multi scalar dengan jumlah data 1.000.000.

C. Hasil Perbandingan

Dari hasil percobaan seperti contoh di atas, maka di dapat hasil perbandingan antara keempat query tersebut. Adapun hasil perbandingannya adalah sebagai berikut:

Tabel 2. Tabel Perbandingan Antar Dua Tabel.

query	Jumlah data dan kecepatan (ms)				
	100	1.000	10.000	100.000	1000.000
Scalar	0	248	353	2036	13331
Corelate	1	256	386	1938	15211
Tercepat	Scalar	Scalar	Scalar	Correlate	Scalar
Terlama	Correlate	Correlate	Correlate	Scalar	Correlate

Untuk perbandingan antar dua tabel hanya bisa menggunakan dua jenis query yaitu scalar dan correlated. Dari tabel di atas dapat dilihat bahwa waktu yang dibutuhkan untuk mengeksekusi query scalar rata-rata lebih cepat dibandingkan untuk mengeksekusi query correlated.

Tabel 3. Tabel Perbandingan Antar Tiga Tabel.

query	Jumlah data dan kecepatan (ms)				
	100	1.000	10.000	100.000	1000.000
Scalar	0	200	415	1628	15000
Corelate	0	202	316	1593	12911
Cross product-scalar	40	350	408	1723	3691
Multi scalar	8	322	480	1746	3535
Tercepat	**Scalar, correlated**	**Scalar**	**Correlated**	**Correlated**	**Multi scalar**
Terlama	**Cross product-scalar**	**Cross product-scalar**	**Multi scalar**	**Multi scalar**	**Scalar**

Dari tabel di atas jenis query yang paling cepat dieksekusi dalam range jumlah data 100 sampai 100.000 adalah scalar dan correlated. Sedangkan pada jumlah data lebih dari 100.000 sampai 1.000.000 jenis query multi scalar yang paling cepat dieksekusi.

Tabel 4. Tabel Perbandingan antar empat tabel.

query	Jumlah data dan kecepatan (ms)				
	100	1.000	10.000	100.000	1000.000
Scalar	0	261	515	1939	12528
Corelate	0	194	419	2029	17229
Cross product-scalar	13	290	452	5567	3978
Multi scalar	10	238	372	2229	3879
Tercepat	**Scalar, correlated**	**Correlated**	**Multi scalar**	**Scalar**	**Multiscalar**
Terlama	**Cross product-scalar**	**Cross product-scalar**	**Scalar**	**Cross product-scalar**	**Correlated**

Dari tabel di atas jenis query yang paling cepat dieksekusi dalam range jumlah data 100 sampai 100.000 rata-rata adalah scalar dan correlated. Sedangkan pada jumlah data lebih dari 100.000 sampai 1.000.000 jenis query multi scalar yang paling cepat dieksekusi.

Tabel 5. Tabel Perbandingan Antar Lima Tabel

query	Jumlah data dan kecepatan (ms)				
	100	1.000	10.000	100.000	1000.000
Scalar	0	290	465	1669	12495
Corelate	86	260	415	1458	12148
Cross product-scalar	9	237	336	1640	3446
Multi scalar	15	208	371	1488	3373
Tercepat	**Scalar**	**Multi scalar**	**Cross product-scalar**	**Correlated**	**Multi scalar**
Terlama	**Correlated**	**Scalar**	**Scalar**	**Scalar**	**Scalar**

Dari tabel di atas jenis query multi scalar lah yang rata-rata dapat dieksekusi paling cepat, walaupun dari tabel di atas masih nampak ketidak stabilan dalam waktu pengeksekusian masing-masing query.

Tabel 6. Tabel Perbandingan Antar Enam Tabel

query	Jumlah data dan kecepatan (ms)				
	100	1.000	10.000	100.000	1000.000
Scalar	1	206	324	1582	12517
Corelate	0	207	343	1475	12666
Cross product-scalar	1	211	380	1479	12617
Multi scalar	0	228	332	1471	11933
Tercepat	Correlate, multi scalar	Scalar	Scalar	Multi scalar	Multi scalar
Terlama	Scalar, cross product-scalar	Multi scalar	Cross product-scalar	Scalar	correlated

Dari tabel di atas dapat dilihat bahwa waktu yang dibutuhkan untuk mengeksekusi query scalar dan multi scalar rata-rata lebih cepat dari jenis query yang lain.

5. KESIMPULAN

Setelah melakukan pengujian seperti yang telah dibahas sebelumnya, maka ditarik kesimpulan sebagai berikut:

1. Untuk relasi antar dua table, hanya bisa menggunakan jenis query scalar dan correlated.
2. Pada perbandingan antar dua table query scalar lah yang lebih baik dibandingkan dengan correlated walaupun waktu pengeksekusian tidak terlalu jauh berbeda.
3. Untuk jumlah data yang kecil dan jumlah relasi yang sedikit, jenis query scalar lah yang lebih optimal digunakan.
4. Jenis query correlated juga tidak menutup kemungkinan bisa menjadi query yang optimal untuk digunakan dalam jumlah data dan relasi yang sedikit karena membutuhkan waktu yang tidak jauh berbeda denagn jenis query scalar.
5. Range jumlah data yang bisa dieksekusi secara optimal menggunakan jenis scalar dan correlated berkisar pada 100 sampai 100.000 data.
6. Untuk jumlah data yang besar kisaran 100.000 sampai 1.000.000 data, jenis query multi scalar adalah jenis query yang paling tepat untuk digunakan.
7. Jenis query multi scalar pun bisa digunakan dalam jumlah data yang sedikit namun memiliki banyak relasi antar table.
8. Untuk relasi antar yang banyak penggunaan jenis query scalar tidak dianjurkan.
9. Tidak dianjurkan pula untuk menggunakan jenis query cross product-scalar dalam optimasi query karena rata-rata waktu yang dibutuhkan untuk mengeksekusi adalah waktu yang paling lambat di antara jenis query yang lain.
10. Struktur database, penggunaan dbms, penggunaan browser, spesifikasi computer juga merupakan factor-faktor yang mempengaruhi optimasi query.

6. DAFTAR PUSTAKA

Ermatita. (2009). Analisis Optimasi Query Pada Data Mining. *Jurnal Sistem Informasi (JSI), Vol.1, No.1, April 2009, Hal 47-54.*

Kusrini. (2007). Strategi Perancangan dan Pengelolaan Basis Data. Penerbit Andi:Yogyakarta.

Maulani, B., A. Haidar M., Maria U. (2015). Analisis Perbandingan Optimasi Query Nested Join dan Hash Join Pada Aplikasi Pencarian Data Berbasis Web. *Jurnal Ilmiah Berry Maulani Vol.1 No.1, September 2015 : 1 – 16.*

Santiputri, M., M. C. Kirana, Anni. (2010). Perbandingan Cross Product dan Subset Query Pada Multiple Relasi Dengan Metode Cost-Based. *Seminar Nasional Informatika 2010 (semnasIF 2010) UPN "Veteran" Yogyakarta, 22 Mei 2010.*

Tomy. (2008). Tip dan Trik Profesional MySQL 5. Elex Media Komputindo.

Wahana Komputer. (2010). Pengembangan Aplikasi Database Berbasis JavaDB dengan Netbeans: Shortcourse Series. Penerbit Andi:Yogyakarta.

W, Tri W. (2008). Pengoptimasian Pencarian Data Dengan Algoritma Subset Query. *Jurnal Artificial, ICT Research Center UNAS Vol.2 No.1 Januari 2008.*