Jurnal Informatika, Komputer dan Bisnis
Vol-1, Issue-1, 2021 (JIKOBIS)
E-ISSN: -
https://jurnal.itbaas.ac.id/index.php/jikobis

ANALISIS STUKTUR KECEPATAN AKSES DATA PADA OPTIMASI QUERY RUSHMORE

Darno*[1], Nendy Akbar Rozaq Rais[2]
[1]STMIK Dharma Negara
[2]Informatika ITB AAS Indonesia
E-mail: *[1]**darno.email@gmail.com**, [2]ab.terate@gmail.com

Abstrak

Analisa struktur kecepatan akses datapada query rushmore berhubungan dengan kecepatan saat memasukan data, melakukan pengolahan data serta pelaporan sebuah data menjadi informasi. Penelitian ini akan membahas tentang setruktur pada kondisi atau perintah serta kecepatan yang dihasilkan dari query rushmore sebagai dasar pengukuran kecepatan sebuah metode query database. Dalam hal kecepatan pengolahan data maka akan berhubungan langsung dengan penyajian data, sehingga apabila data semakin cepat dan tepat maka akan menjadi penyajian data yang baik. Dalam penelitian ini dilatar belakangi oleh perlunya informasi dalam penyajian data sebagai contoh dengan metode query rusmore. Penelitian ini akan menjelaskan beberapa poin sebagai tujuan penelitian yang diantaranya gambaran umum metode query rusmore, struktur dari metode query rushmore, perintah penggunaan query rushmore, pengujian query rasmore dan hasil query rushmore. Dengan adanya tujuan tersebut maka dapat dilihat hasil dari penelitian ini sebuah informasi untuk mengetahui struktur metode query rushmore. Hasil penelitian dapat memberikan sebuah informasidalam struktur kecepatan query rushmore yang dapat digunakan sebagai dasar pertimbangan dalam penelitian atau pembuatan sebuah database yang cepat dan tepat.

Kata Kunci : *data, struktur, database, query, rushmore*

Abstract

Analysis of the data access speed structure in queries related to the speed when entering data, processing data and reporting data into information. This research will discuss about the structure of the conditions or orders as well as the speed generated from the rushmore query as a basis for measuring the speed of a database query method. In terms of data processing speed, it will be directly related to data presentation, so that the data is faster and more precise, it will be a good data presentation. This research is motivated by the need for information in presenting data, for example by using the rusmore query method. This study will explain several points as research objectives, including an overview of the rusmore query method, the structure of the rushmore query method, the use of rushmore queries, testing the rasmore query and the results of the rushmore query. With this aim, it can be seen from the results of this study from an information to see the structure of the Rushmore query method. Research results can provide information in a rushed demand structure that can be used as a basis for consideration in research or creating fast and precise databases.

Keywords : *data, structure, database, query, rushmore*

1. PENDAHULUAN

1.1. Latar Belakang

Sebuah database yang berhubungan dengan penyajian data maka unsur kecepatan dan ketepatan menjadi unsur pokok dalam penyajian datanya. Penyajian data pasti akan dibutuhkan dari sebuah perusahaan maupun individu yang ingin melihat dan memiliki informasi tertentu untuk bahan pertimbangan ataupun bahan pengambilan keputusan. Dalam hal penyajian data perlunya kecepatan dan ketepatan data sebagai awal langkah mengetahui informasi. Dalam penyajian data terdapat beberapa metode untuk melakukanya, dalam penelitian ini mengambil contoh metode *query rushmore*. Setiap metode query pada database memiliki kelebihan dan kelemahan masing-masing, tetapi dalam hal ini metode *query rushmore* menjadi bahan penelitian. Metode *query rushmore* disini akan dibahas mengenai kecepatan akses data ke sebuah database. Pada penelitian ini memiliki dasar sebagai penelian berbasis analisis kecepatan dan ketepatan sebuah metode query database (rush more), metode *query rushmore* ini menjadi topik pembahasan nantinya yang mengenai kecepatan akses data ke sebuah database. Adanya penelitian analisis kecepatan akses data pada *query rushmore* ini juga dilatar belakangi permasalahan keingin tahuan lebih dalam tentang kelebihan dan kelemahan (kecepatan dan kelemahan) metode *query rushmore* untuk melakukan pengolahan data dari sebuah database, sehingga penelitian ini dilakukan untuk mengetahui seberapa cepat dan tepat metode ini saat penerapannya.

1.2. Tujuan penelitian

Analisis kecepatan akses data pada *query rushmore* ini memiliki tujuan yang sebagai dasar penelitian yang diantaranya :

1. Mengetahui struktur kecepatan akses data metode *query rushmore*
2. Kemampuan metode *query rushmore* dalam penyajian data
3. Mengetahui kapan metode rushmore tepat untuk digunakan

1.3. Batasan Penelitian

Dalam penelitian ini tentu memiliki batasan sebagai dasar penelitian supaya menjadi penelitian yang relevan terhadap topik penelitian, adapaun batasanya yang diantaranya:

1. Mencangkup sebagian perintah dari *query rushmore*
2. Menggabarkan struktur *query rushmore* dalam bentuk gambar secara umum
3. Mencangkup hasil sebuah query dapat dioptimasi atau tidak

1.4. Hasil Penelitian

Hasil dari penelitian ini akan berupa gambaran besaran kecepatan akses data dengan menggunakan metode *query rushmore* pada pengolahan data di database. Dalam penenelitian analisis kecepatan akses data dengan *query rushmore* ini akan mengetahui cara kerja dan struktur dari *query rushmore*.

1.5. Tinjauan Pustaka

Beberapa penelitian sebelumnya pernah memiliki topik ataupun tema yang berhubungan langsung ataupun tidak langsung dengan penelitian ini, dari penelian yang sebelumnya tersebut akan dijadikan sebagai referensi dalam penelian ini.

1.6. Query rushmore

Teknologi Optimasi *Query rushmore* sebagai salah satu teknologi untuk meningkatkan kecepatan akses data dapat dijadikan alternative pemecahan masalah kinerja sebuah aplikasi khususnya berhubungan dengan sebuah basis data. Teknologi Optimisasi *Query rushmore* adalah suatu teknik akses data yang menggunakan Index standart/baku untuk mengoptimalkan akses ke data. Teknologi Rushmore dapat digunakan dengan berbagai bentuk index termasuk Idx Index, compact (.idx) index, dan index campuran (.cdx). Tipe index compact(idx) dan campuran(cdx) menggunakan suatu teknik kompresi/pemampatan yang menghasilkan index lebih kecil 16 kali dari aslinya(tanpa kompresi). Visual Foxpro dapat memproses suatu index yang dikompresi dengan lebih cepat sebab memerlukan lebih sedikit akses disk, dan sebab lebih banyak index dapat disimpan di memori. Walaupun demikian teknologi optimisasi *Query rushmore*, seperti teknik akses file yang lain, bermanfaat bagi dari ukuran yang lebih kecil dari bentuk index compact, juga berfungsi baik pada index format lama [1].

Dari penjelasan diatas maka dapat diambil garis besarnya bahwa *query rushmore* merupakan sebuah teknologi ang digunakan untuk mempercepat akses data dalam databese dengan langkah dan perintah-perintah tertentu dan dikemas dalam bentuk informasi tertentu yang akan memudahkan dalamp penyajian data. Dari tinjauan pustaka tentang *query rushmore* tersebut memperlihatkan kelemahan dan kekurangan *query rushmore* yaitu hasil pemprosesan indek berupa akses disk yang kecil kalah dari pemprosesan visual foxpro sebagai kekuranganya tetapi memiliki kelebihan Rushmore dapat digunakan dengan berbagai bentuk index termasuk Idx Index, compact (.idx) index, dan index campuran (.cdx). Tipe index compact(idx) dan campuran(cdx).

1.7. Optimasi Query

Optimasi query dapat dilakukan dengan beberapa cara, dalam hal ini terdapat 2 pendekatan optimasi yang dapat digunakan menurut Chanowich (2001), 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:

- melakukan operasi *selection* di awal untuk mereduksi jumlah baris
- melakukan operasi projection di awal untuk mereduksi jumlah atribut
- mengkonversikan query dengan banyak join menjadi query dengan banyak subquery
- 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.

Keuntungan dari optimizer adalah dapat mengakses semua informasi statistik dari sebuah database. Selain itu optimizer juga dapat dengan mudah untuk melakukan optimisasi kembali apabila informasi statistik sebuah database berubah dan optimizer dapat menangani strategi yang berbeda-beda dalam jumlah besar yang tidak mungkin dilakukan oleh manusia. Input dari optimizer adalah sebuah tree yang sudah mengalami proses parsing di dalam query parser. Tree tersebut biasanya disebut dengan *parse tree*. Sedangkan output dari optimizer adalah berupa rencana eksekusi (execution plan) yang siap untuk dikirimkan ke dalam query kode generator dan query processor untuk diproses untuk mendapatkan hasil akhir dari query tersebut. Proses optimisasi query dapat dianggap mempunyai dua tingkatan. Dua tingkatan tersebut adalah : *rewriting* dan *planning*. Hanya ada satu modul pada tingkat pertama yaitu *Rewriter*, dimana semua modul-modul lainnya berada pada tingkat kedua. Tahap penulisan dapat disebut sebagai level *declarative*, sedangkan tahap perencanaan dapat juga disebut sebagai level *procedural* [2].

1.8.Faktor Penentu Kecepatan Akses Data

Dalam melakukan query dalam hal percepatan akses data ada beberapa poin yang diperhatikan, dalam hal ini poin-poin tersebut pasti akan berkenaan langsung dengan data secara langsung, poin-poin yang harus diperhatikan yaitu sebagai berikut:

1. Optimasi Aplikasi

   Dalah hal optimasi aplikasi disini dimaksudkan bahwa Pembuatan aplikasi, perlu memperhatikan apakah akses terhadap data sudah efisien. Efisien dalam hal penggunaan obyek yang mendukung kecepatan akses, seperti index atau cluster. Kemudian juga bagaimana cara database didesain. Apakah desain database sudah melakukan normalisasi data secara tepat. Kadangkala normalisasi sampai level yang kesekian, tidak menjamin suatu desain yang efisien. Untuk membuat desain yang lebih tepat, kadang setelah melakukan normalisasi perlu dilakukan denormalisasi. Misalnya tabel yang hubungannya one-to-one dan sering diakses bersama lebih baik disatukan dalam satu tabel [2]

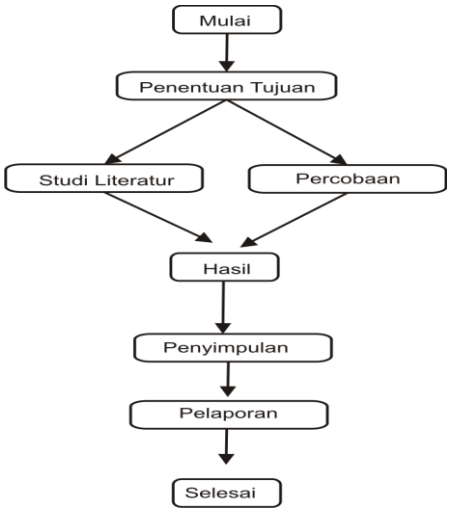
2. Cluster dan Index

   Cluster adalah suatu segment yang menyimpan data dari tabel yang berbeda dalam suatu struktur fisik disk yang berdekatan. Konfigurasi ini bermanfaat untuk akses data dari beberapa tabel yang sering di-query. Penggunaan cluster secara tepat dilaksanakan setelah menganalisa tabel-tabel mana saja yang sering di-query secara bersamaan menggunaan perintah SQL join. Jika aplikasi sering melakukan query dengan menggunakan suatu kolom yang berada pada klausa WHERE, maka harus digunakan index yang melibatkan kolom tersebut. Penggunaan index yang tepat bergantung pada jenis nilai yang terdapat dalam kolom yang akan diindex. Dalam RDBMS Oracle, index B-Tree digunakan untuk kolom yang mengandung nilai yang cukup bervariasi, sedangkan untuk nilai yang tidak memiliki variasi cukup banyak, lebih baik menggunakan index bitmap [2]

2. METODE PENELITIAN

Dalam hal metode penelitian maupun pengambilan data dilakukan pengamaatan dan melakukan studi literatur yang berkenaan langsung dengan tema dalam penelitian ini atau dengan kata lain studi literatur terhadap pustaka-pustaka yang berkaitan dengan analisis kecepatan akses data pada *query rushmore*.

Peleitian ini akan dilakukan dalam beberapa tahap untuk menghasilkan tujuan utama dari penelian ini, dalam penelitian ini ada beberapa tahap yang diantaranya dapat diliihat pada gambar 1 tentang alur penelitian.



Gambar 1 Alur Penelitian

Pada Gambar 1 dapat dilihat alur proses penelitian yang diantaranya memulai dengan langsung menentukan tujuan karena penelitian bersifat analisi yang akan dilaterbelakangi permasalahan, berlanjut ke proses studi literatur untuk menemukan pustaka-pustaka yang mendukung penelitian analisis struktur kecepatan akses datap pada optimasi *query rushmore* dan percobaan dengan mengggunakan kondisi serta perintah-perintah pada *query rushmore* dapatkah dilakukan optimasi, selanjutnya tahap hasil yaitu tahap mengetahui hasil dari studi litelatur dan percobaan, selanjutnya penyimpulan dari hasil sebuah penelitian dan selanjutnya dokumentasi atau membuat daftar penelitian dan poin-poin penelitian sampai selesai.

3. HASIL DAN PEMBAHASAN

Query rushmore adalah suatu metode percepatan akses data dengan perintah query tertentu, *query rushmore* sebagai pengoptimalan akses data tergantung pada banyaknya tabel dan data yang akan diolah. Ketika mengakses tabel-tabel tunggal penggunaan *query rushmore* memiliki kemudahan dan keuntungan yaitu menggunakan *query rushmore* dengan perintah FOR dan ketika pengolahan data dan tabel banyak maka dibutuhkan teknik lain dari query yaitu dengan perintah SELECT- SQL query untuk sebagai pengganti optimasi kecepatan akses data dari *query rushmore* dengan perintah FOR menjadi SELECT-SQL.

Fungsi dan cara kerja *query rushmore* ialah sebagai pemercepat kinerja dari perintah-perintah untuk akses tabel tunggal yang menggunakan perintah “FOR “ yang akan mendeklarasikan sebuah set record dalam kaitan dengan index yang ada. *Query rushmore* ini digunakan sebagai percepatan record tetapi juga dapat digunakan juga untuk mempercepat operasi dari perintah tertentu seperti LOCATE dan INDEX.

3.1. Penggunaan Perintah Query rushmore

Pada intinya penggunaan *query rushmore* yaitu apa bila data tunggan lebih baik menggunakan perintah FOR dan apabila data banyak makan menggunakan perintah SELECT-SQL. Adapun contoh perintah dasar pada *query rushmore* yang dapat dioPtimasi untuk percepatan akses data diantanya seperti pada tabel 1.

Tabel 1 Perintah Query Rusmore Yang Dapat di Optimasi Untuk Percepatan Akses Data [3]

AVERAGE	BLANK
BROWSE	CALCULATE
CHANGE	COPY TO
COPY TO ARRAY	COUNT
DELETE	DISPLAY
EDIT	EXPORT TO
INDEX	JOIN WITH
LABEL	LIST
LOCATE	RECALL
REPLACE	REPLACE FROM ARRAY
REPORT	SCAN
SET DELETED	SET FILTER
SORT TO	SUM
TOTAL TO	

Pada tabel 1 diatas dapat dijelaskan fungsi dari perintah tersebut sebagai contoh penggunaan perintah berikut:

AVERAGE : Digunakan untuk membuat rata-rata suatu data yang memiliki nilai.
BROWSE : Fungsi dari perintah browse untuk perintah proses cari sebuah data
DELETE : Perintah untuk melakukan menghapus sebuah data
EDIT : Perintah yang digunakan untuk merubah sebuah data atau melakukan manipulasi dari sebuah data
REPLACE : perintah yang digunakan untuk melakukan pergantian nama atau pergaintian data dengan data yang lain.
SORT TO : perintah ini digunakan untuk melakukan tindakan untuk menampilkan sebuah data berdasarkan kriteria-kriteria tertentu perintah SORT TO ini memiliki fungsi peran sama pada SQL / MYSQ dengan perintah SORT BY yang juga berfungsi untuk menampilkan data sesuai kriteria tertentu.

Selain penjelasan perintah diatas tentu masih banyak perintah-perintah yang lain tentunya memiliki fungsi dan kegunaan masing-masing yang pada intinya penggunaan perintah *query rushmore* diatas tersebut digunakan sebagai metode optimasi kecepatan akses data pada sebuah database.

3.2. Ketepatan Penggunaan Query rushmore

Dalam *query rushmore* tidak semua stetement/kalimat perintah dapat dioptimasi secara kecepatan akses. Tetepi *query rushmore* dapat digunakan saat kondisi stetement tidak terdapat ekpresi FOR dan NOT. Berikut contoh perintah *query rushmore* yang dapat dioptimasi kecepatanya dan yang tidak dapat dioptimasi kecepatan aksenya:

Index on not deleted() tag; notdel

Perintah diatas dengan maksud tujuan akan mendeklarasikan sebuah data dalam sebuah database untuk tidak dapat dihapus, dengan pendeklarasian perintah tersebut maka kondisi tersebut tidak dapat dilakukan optimasi dikarenakan terdapat kondisi NOT atau dalam kondisi

"tidak dapat" tetapi beda halnya apabila kkondisi perintah untuk melakukan optimasi kecepatan pada perintah berikut

Index on deleted() tag del

Dengan perintah "Index on deleted() tag del" tersebut maka sebuah data dapat dioptimasi kecepatan akses datanya sehingga perbedaan perintah dengan penambahan sebuah kondisi (FOR dan NOT) tidak memenuhu syarat sebagai perinttah optimasi. Lain halnya dengan perintah kedua yang tidak mengandung unsur kondisi (FOR dan NOT) sehinggga dapat dilakukan optimasi kecepatan akses datanya. Dari kedua contoh perintah diatas walaupun sebenarnya perintah tersebut bertujuan untuk melakukan percepatan akses data yaitu dengan menambahkan ekperesi ON pada perintahnya tetapi tidak semua dapat dilakukan optimasi kecepatan dengan penjelasan diatas.

3.3. Kondisi Tanpa Query rushmore

Kondisi dimana pada saat melakukan *query* pada saat-saat tertentu yang tidak memerlukan *query rushmore* yang dimana kkondisi tertentu adalah proses melakukan *query* data yang biasa ataupun jumlah data tidak terlalu banyak sehingga *query rushmore* tidak diperlukan sementara. Kondisi yang tidak memerlukan *query rushmore* yaitu pada kondisi sebagai berikut:

1. Ketika *query rushmore* tidak bisa mengoptimalkan ekspresi klausa FOR dan NOT dalamsuatu perintah yang dapat mengoptimasi.
2. Ketika suatu perintah atau kondisi mengandung perintah WHILE atau DO WHILE sehingga deklarasi stetemen secara satu persatu.
3. Ketika memori penyimpanan terbatas/kecil, sehingga mengganggu penyimpanan data hasil optimasi.

Dengan pertimbangan ketiga hal tersebut maka tidak semua kondisi dapat dilakukan optimasi *query rushmore*. Dengan demikian bahwa ada beberapa aspek yang hasus diperhatikan dalam menentukan ketepatan penggunaan *query rushmore*.

3.4. Cara Mengaktifkan Dan Menonaktifkan Optimasi Queri Rushmore

Untuk melakukan sebuah kondisi perlu atau tidaknya *query rushmore* tergantung dari beberapa aspek seperti pada pembahasan sebelumnya. Apabila sebuah kondisi memerlukan tindakan untuk melakukan optimasi *query rushmore* maka *query rushmore* dapat diaktifkan dan apabila *query rushmore* tidak diperlukan maka dapat dilakukan tindakan menonaktifkan query rusmore tersebut. Berikut contoh perintah untuk melakukan aktivasi dan menonaktifkan *query rushmore*:

Aktivasi *Query rushmore*: Set optimaze on

Menonaktifkan *Query rushmore*: Set optimaze off

Dari perintah diatas sudah jelas kelihatan bahwa sebuah perintah ataupun kondisi untuk melakukan aktivasi maupun mennonaktifkan sebuah kondisi melalui perintah (ON atau OFF). Dari perintah diatas terlihat sebuah kondisi dengan menggunakan perintah SET OPTIMAZE maka dalam perintah tersebut sebuah kondisi dimana sebuah data akan dilakukan sebuah optimasi kecepatan aksesnya.

Setelah melakukan aktivasi optimasi *query rushmore* dapat dilakukan kondisi mendeklarasikan perintah-perintah dasar sebagai berikut:

eIndex relOp eExp

atau

eExprrelOp eIndex

Pada *query rushmore* memiliki kriteria dan karakteristik yang berikut:

1. Pada ekpresi query index yang dibuat memerlukan eIndex sesuai dengan ekspresi yang ada.
2. eExpr ekspresi yang digunakan pada saat variable dan *fields* dari tabel yang tidak berelasi.
3. relOp dari operator relasi (<,>,=,<=,>=,<>,#,==, atau!=.) Dapat juga menggunakan fungsi-fungsi ISNULL (), BETWEEN(), atau INLIST() (atau SQL padanannya seperti IS NULL, dan sebagainya).

Query rushmore dapat mengoptimasi data dalam hal akses data dalam ekspresi hanya jika mencari data kembali terhadap penggunaan ekspresi yang tepat dalam suatu index. Sebagai contoh bayangkan telah dibuat suatu tabel dan ditambahkan index yang pertama dengan menggunakan suatu perintah seperti berikut:

USE USER
INDEX ON UPPER(cu_name) TAG name

1. Pada perintah diatas terlihat bahwa perintah tersebut terlihat tidak tersetruktur tetapi jelas maksud dan tujuan, maka dari itu perintah diatas tidak dapat dioptimasi kecepatan akses datanya. Lain halnya pada perintah SQL yang mempunyai struktur query dengan kondisi yang bagus sebagai contoh menggunakan sehingga dapat dilakukan optimasi akses datanya dengan perintah berikut:

*SELECT * FROM User WHERE;*
cu_name ="ACME"

Perintah maka akan menghasilkan penyajian data diatas tidak hanya melakukan pencarian pada field cu_name saja melaainkan mengoptimasi semua data pada sebuah tabel.

Sebuah data dapat di indexkan dengan berbagai kombinasi untuk menjadi sebuah expresi, dapat dilakukan dengan kombinasi operator FOR atau WHERE selainitu juga dapat dilkukan kombinasi dengan operator OR, END dan NOT. Jika suatu ekspresi yang dasarnya tidak optimal, maka ekspresi yang kompleks boleh jadi secara parsial optimizable atau tidak optimizable sama sekali, dalam penggunaan kombinasi operator tersebut memiliki aturan-aturan kombinasi ekpresi dasar yang ada pada *query rushmore* yang diantaranya dapat dilihat pada Tabel 2 dan aturan kombinasi ekspresi kompleks dapat dilihat pada Tabel 3.

Tabel 2 Aturan *Query rushmore* Dengan Kombinasi Dasar Menggunakan Operator

Basic Expression	Operator	Basic Expression	Query Result
Optimizable	AND	Optimizable	Fully Optimizable

Optimizable	OR	Optimizable	Fully Optimizable
Optimizable	AND	Not Optimizable	Not Optimizable
Optimizable	OR	Not Optimizable	Not Optimizable
Not Optimizable	AND	Not Optimizable	Not Optimizable
Not Optimizable	OR	Not Optimizable	Not Optimizable
-	NOT	Optimizable	Fully Optimizable
-	NOT	Not Optimizable	Not Optimizable

Tabel 2 diatas adalah ekpresi dah hasil apakah perintah dengan opertor tertentu dapat dioptimasi dan menghasilkan data yang optimasi pada akses datanya.

Tabel 3 Aturan *Query rushmore* Dengan Kombinasi Kompleks Menggunakan Operator

Basic Expression	Operator	Basic Expression	Query Result
Fully Optimizable	AND	Fully Optimizable	Fully Optimizable
Fully Optimizable	OR	Fully Optimizable	Fully Optimizable
Fully Optimizable	AND	Partially Optimizable	Partially Optimizable
Fully Optimizable	OR	Partially Optimizable	Partially Optimizable
Fully Optimizable	AND	Not Optimizable	Partially Optimizable
Fully Optimizable	OR	Not Optimizable	Not Optimizable
-	NOT	Fully Optimizable	Fully Optimizable
Partially Optimizable	AND	Partially Optimizable	Partially Optimizable
Partially Optimizable	OR	Partially Optimizable	Partially Optimizable
Partially Optimizable	AND	Not Optimizable	Partially Optimizable
Partially Optimizable	OR	Not Optimizable	Not Optimizable
-	NOT	Partially Optimizable	Not Optimizable
Not Optimizable	AND	Not Optimizable	Not Optimizable
Not Optimizable	OR	Not Optimizable	Not Optimizable
-	NOT	Not Optimizable	Not Optimizable

Pada Tabel 3 dapat dilihat ada berbagai kombinasi antar beberapa expresi dang operator dan menghasilkan optimasi data apakah data dapat dioptimasi maupun tidak dapat dioptimasi.

3.5. Hasil Percobaan

Dengan melakukan percobaan menggunakan jet machine sp.2 *product microsoft* bahwa dilakukan dengan hasil dapat dilihat pada Tabel 4.

Tabel 4 Percobaan Query Rushmore

Jumlah Record	Jumlah Data Percobaan	Waktu Dibutuhkan
46	100	0,07
46	1000	0,12
46	10000	0,37
46	100000	0,55
490	100	0,15
490	1000	0,26
490	10000	0,41
490	100000	0,75

Dari Tabel 4 tersebut di ketahui bahwa gambaran tabel dalam bentuk perkiraan percobaan pada optimasi *query rushmore.*

4. KESIMPULAN

Dari tujuan penelitian yang ingin mengetahui struktur optimasi kecepatan akses data pada *query rushmore*, bahwa penelitian ini telah mampu menjawab dari tujuan yang dimaksud dengan bukti bahwa:

1. Dapat diketahuai bagaimana struktur dari *query rushmore*
2. Perintah-perintah untuk optimasi dengan *query rushmore*
3. Kemampuan apa yang dimiliki *query rusmore* dalam hal penyajian data
4. kapan atau kondisi yang tepat penggunaan *query rushmore*.

Dari beberapa poin dari kesimpulan diatas dapat disimpulkan bahwa *query rushmore* dapat diterapkan untuk melakukan optimasi *query* data untuk penyajian data dengan struktur yang benar dan tepat penggunaanya.

5. SARAN

Adapun saran dari penelitian ini karena penelitian ini hanya mencangkup analisis maka percobaan yang dilakukan hanya sebatas untuk pembuktian dan tidak atau belum dilakukan pada kondisi nyata dengan kondisi tertentu, sehingga penelitian ini pasti masih banyak kekuranganannya untuk itu perlu sebuah evaluasi dan pengembangan lebih lanjut, adapun poin-poin saran untuk dapat ditindak lanjuti :

1. Penggunaan *query rushmore* pada saat queri data kompleks data *real*
2. Hasil ststistika kecepatan akses data menggunakan *query rushmore*
3. Ketepatan *query rushmore* dalam penyajian data

DAFTAR PUSTAKA

[1] Jananto, A, 2006, Meningkatkan Kecepatan Akses Data Dengan Teknologi Optimasi Query Rushmore, *Jurnal Teknologi Informasi DINAMIK*, vol. XI, No. 1, hal 47-56

[2] Ermatita, 2009, Analisis Optimasi Pada Data Mining, *Jurnal Sistem Informasi (JSI)*, vol 1, no 1, hal 47-54

[3] Learning resaurce of microsoft, Using Rushmore Query Optimization To Speed Data Access, http//msdn.microsoft.com/en-us/library/, diakses tanggal 27 Maret 2021