Meningkatkan Kecepatan Akses Data dengan Teknologi Optimasi Query Rushmore

Arief Jananto
Fakultas Teknologi Informasi, Universitas Stikubank Semarang
email : arief@unisbank.ac.id

Abstrak : Informasi sebagai keluaran(output) sebuah system informasi akan sangat bernilai guna tinggi jika memenuhi tiga hal kualitas yaitu relevan, akurat dan tepat waktu. Untuk mencapai kualitas tersebut dan menyesuaikan dengan tuntutan jaman yang serba cepat maka akses informasi yang cepat menjadi faktor utama yang harus dipenuhi. Data adalah deskripsi dari sesuatu dan kejadian yang dihadapi, dicatat dalam suatu media penyimpanan. Sebagai bahan baku dari informasi, maka kecepatan akses informasi sama dengan kecepatan akses data tersebut. Semakin cepat dan mudah akses terhadap data maka penyajian informasi sebagai kebutuhan dari manajemen untuk mengambil keputusan juga dapat ditingkatkan.
Kecepatan akses data dapat ditingkatkan dengan banyak cara, selain dari sisi perangkat kerasnya (hardware) dapat juga dilakukan dari sisi perangkat lunaknya (software) lebih khusus pada program aplikasinya. 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.

Kata kunci : data, informasi, kualitas, teknologi, rushmore, optimasi, query

PENDAHULUAN

Informasi dapat diibaratkan sebagai darah yang mengalir di dalam tubuh manusia, seperti halnya informasi di dalam sebuah perusahaan yang sangat penting untuk mendukung kelangsungan perkembangannya, sehingga terdapat alasan bahwa informasi sangat dibutuhkan bagi sebuah perusahaan.Akibat bila kurang mendapatkan informasi, dalam waktu tertentu perusahaan akan mengalami ketidakmampuan mengontrol sumber daya, sehingga dalam mengambil keputusan-keputusan strategis sangat terganggu, yang pada akhirnya akan mengalami kekalahan dalam bersaing dengan lingkungan pesaingnya.

Disamping itu, sistem informasi yang dimiliki seringkali tidak dapat bekerja dengan baik. Masalah utamanya adalah bahwa sistem informasi tersebut terlalu banyak informasi yang harua dikelola. Memahami konsep dasar informasi adalah sangat penting (*vital*) dalam mendesain sebuah sistem informasi yang efektif *(effective business system)*. Menyediakan informasi yang berkualitas dari sisi waktu atau kecepatan akses adalah tujuan dalam mendesain sistem baru yang lebih efektif dan efisien.

Data disimpan didalam suatu aplikasi dalam bentuk file tabel yang disertai dengan kunci index untuk mempermudah akses. Teknologi Optimasi query Rushmore merupakan salah satu teknologi procedural untuk meningkatkan kinerja dari aplikasi khususnya dari sisi kecepatan akses data dari file tabel dengan memanfaatkan index yang dimiliki atau telah ada.

DATA DAN INFORMASI

Data "Adalah deskripsi tentang benda, kejadian, aktivitas, dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai." (Abdul kadir : Pengenalan Sistem Informasi :29: 2002)

Representasi dari data dapat berupa : 1). Data yang terformat, yaitu data yang ditampilkan dalam kondisi yang sudah terformat seperti data tanggal, jam, suhu udara; 2). Teks,yaitu data yang ditampilkan dalam bentuk kumpulan karakter, huruf, angka maupun tanda baca seperti artikel Koran; 3). Citra yaitu data

yang ditampilkan dalam bentuk gambar atau lukisan seperti foto, tanda tangan; 4).Audio yaitu data yang ditampilkan melalui media Bantu lain berupa perlatan audio atau atau dalam bentuk suara yang dapat didengar oleh manusia seperti lagu(mp3 file); 5).Video yaitu data yang ditampilkan melalui media bantu lain yang berupa peralatan yang menampilkan gambar bergerak dan sekaligus suara yang dapat dinikmati oleh manusia seperti film.

Data merupakan bentuk jamak dari bentuk tunggal data-item. Data merupakan bentuk yang belum dapat memberikan manfaat yang besar bagi penerimanya, sehingga perlu suatu model yang nantinya akan dikelompokkan dan diproses untuk menghasilkan informasi.

Menurut *Davis: 1999,* informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau saat mendatang. Dan menurut *Mc Fadden,dkk:1999,* Informasi dapat didefinisikan sebagai data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakannya.

Data akan melalui suatu tahapan pengolahan data yang terdiri atas *1.) data input (Recording, Coding, Storing); 2.) Data transformation (Calculating, Summarizing, Classifying); 3.) Information output (Displaying result, Reproducing , Telecommunicating),* akan dihasilkan informasi yang sangat dibutuhkan oleh manajemen untuk mengambil keputusan. Informasi yang dihasilkan harus memiliki suatu ketentuan kualitas agar keputusan yang diambil dapat dipertanggungjawabkan dan mendatangkan suatu keuntungan bagi perusahaan.

Kualitas informasi (*quality of information*) sangat dipengaruhi atau ditentukan oleh 6 hal, yaitu :

a. Relevan *(relevancy),*

   Informasi harus memberikan manfaat bagi pemakainya. Relevansi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda. Misalnya informasi mengenai sebab-musabab kerusakan mesin produksi kepada akuntan perusahaan adalah kurang relevan dan akan lebih relevan bila ditujukan kepada ahli teknik perusahaan. * *How is the message used for problem solving (decision masking) ?*

b. Akurat *(accuracy)*

   Informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan, dan harus jelas mencerminkan maksudnya. Ketidakakuratan dapat terjadi karena sumber informasi (data) mengalami gangguan atau kesengajaan sehingga merusak atau merubah data-data asli tersebut. Komponen akurat meliputi : b.1) Completeness(*Are necessary message items present ?)*. Berarti informasi yang dihasilkan atau dibutuhkan harus memiliki kelengkapan yang baik, karena bila informasi yang dihasilkan sebagian-sebagian tentunya akan mempengaruhi dalam pengambilan keputusan atau menentukan tindakan secara keseluruhan, sehingga akan berpengaruh terhadap kemampuannya untuk mengontrol atau memecahkan suatu masalah dengan baik.; b.2) Correctness (*Are message items correct ?);* b.3) Security (*Did the message reach all or only the intended systems users ?)*

c. Tepat waktu (*timeliness*)

   Informasi yang dihasilkan atau dibutuhkan tidak boleh terlambat (usang). Informasi yang usang tidak mempunyai nilai yang baik, sehingga kalau digunakan sebagai dasar dalam pengambilan keputusan akan berakibat fatal atau kesalahan dalam keputusan dan tindakan. Kondisi demikian menyebabkan mahalnya nilai suatu informasi, sehingga kecepatan untuk mendapatkan, mengolah dan mengirimkannya memerlukan teknologi-teknologi terbaru. * *How quickly is input transformed to correct output ?*

d. Ekonomis *(Economy)* * *What level of resources is needed to move information through the problem-solving cycle ?*

e. Efisien *(Efficiency)* * *What level of resources is required for each unit of information output ?*

f. Dapat dipercaya *(Reliability)*

Pada kondisi saat ini,dimana segala sesuatu serba cepat dan instant maka permasalah atau faktor yang ketiga yaitu tepat waktu(timelines) menjadi utama. Bahkan banyak system informasi faktor kecepatan mencari dan menampilkan informasi yang diinginkan menjadi suatu daya tarik paling kuat saat ini. Sebagai contoh untuk mencari informasi melalui website atau internet, pengguna akan banyak menggunakan search engine dapat cepat dalam menampilkan informasi dengan tingkat keakuratan yang juga tidak ditinggalkan.

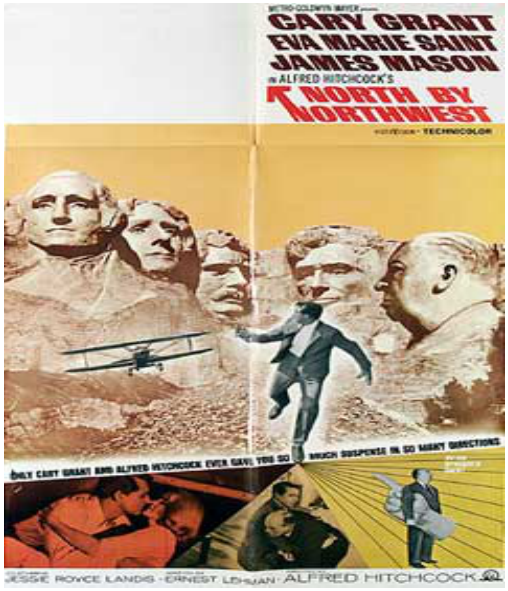
Sebuah system informasi berbasis computer (CBIS=Computer Base Information System) secara umum merupakan sebuah aplikasi pengelola database. Jadi dalam sebuah aplikasi system informasi data akan disimpan dalam bentuk tabel-tabel data. Kecepatan pengaksesan data dari tabel-tabel tersebut akan sangat mempengaruhi kecepatan dan keakuratan hasil yang akan ditampilkan.

Dengan demikian agar informasi dapat dihasilkan secara cepat dan akurat maka perlu adanya peningkatan kinerja dari aplikasi system informasi yang dapat ditopang dengan meningkatkan kecepatan akses data. Salah satu aplikasi basis data yang paling umum dan banyak digunakan sejak dahulu adalah Visual Foxpro yang merupakan pengembangan lebih lanjut dari Dbase dan FoxBase.

Untuk membantu mengoptimalkan kinerja sebuah aplikasi, visual foxpro menambahkan teknologi akses data yang disebut dengan Rushmore Query Optimization. Dengan menggunakan Teknologi ini, aplikasi dapat menjalankan ratusan operasi manipulasi tabel yang komplek atau bahkan beribu kali lebih cepat dari tanpa itu.

MENGAPA DINAMAKAN *RUSHMORE*

Bersumber dari **The Heart of The Fox** berikut adalah sejarah mengapa teknologi ini dinamakan dengan Rushmore. http://www.foxprohistory.org/***rushmore****.htm*



The Fox Software design team came up with the "code-name" Rushmore after spending a night watching the Alfred Hitchcock's movie "North by Northwest".

"North by Northwest" (1959) is an Alfred Hitchcock classic, suspenseful comic thriller - one of the most entertaining movies ever made and one of his most famous suspense/mystery stories.

The film's theme includes elements typical of many Hitchcock films (especially "The 39 Steps" (1935) and "Saboteur" (1942)) - predominantly the themes of mistaken identity for the innocent ordinary hero, false pretenses and survival in 20th Century America.

A light-hearted and complacent hero/bystander (Cary Grant in the role of Roger Thornhill, a Manhattan advertising executive), totally vulnerable, isolated and victimized (and mistaken for a government agent by a group of spies), is on-the-run as a murder suspect (framed for murder at the UN), and pursued (across the US) by a seeming conspiratorial group of spies, the police, and the FBI. The American is eventually forced to assume another man's identity, while confronted with murder, mayhem, a world of spies and counterspies, a domineering mother, and an untrustworthy lover. His final salvation occurs on the Presidential faces carved on Mount

Rushmore.

The original title was "The Man In Lincoln's Nose", which was replaced by a reference to a line from William Shakespeare's "Hamlet" (in which Hamlet says, "I am but mad north-north-west.").

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.

Pada saat Visual Foxpro memproses tabel sangat besar pada komputer dengan hanya RAM yang kecil, Rushmore tidak mengalami masalah. Pada kondisi memori terbatas, Visual Foxpro akan menampilkan pesan (" Tidak cukup memori untuk optimisasi"). Walaupun program berfungsi dengan tepat dan tanpa kegagalan data, tetapi optimasi menjadi sia-sia.

Dalam bentuk sederhana , Rushmore mempercepat kinerja dari perintah akses tabel tunggal yang menggunakan klausa FOR yang menspesifikasikan suatu set record dalam kaitan dengan index yang ada. Rushmore juga dapat mempercepat operasi dari perintah tertentu seperti LOCATE dan INDEX.

Perintah SQL Visual FOXPRO menggunakan Rushmore sebagai tool utama di dalam optimisasi query multi-table, penggunaan index yang ada dan bahkan menciptakan index baru tersendiri untuk mempercepat query.

PENGGUNAAN RUSHMORE DENGAN BANYAK TABEL

Penggunaan Rushmore untuk mengoptimalkan akses data tergantung dari banyaknya tabel yang terlibat. Ketika mengakses tabel-tabel tunggal, didapat keuntungan dari Rushmore sepanjang menggunakan klausa FOR. Ketika mengakses banyak tabel, SELECT- SQL query menggantikan semua Optimisasi Rushmore. Dalam suatu perintah SQL, Visual Foxpro memutuskan apa yang diperlukan untuk mengoptimalkan suatu query dan mengerjakannya. Tidak harus membuka tabel atau index,jika SQL memerlukan maka akan dibuat index sementara di memori.

Cara menggunakan Optimisasi Query Rushmore

Ada dua bentu, pilih salah satu dari pilihan berikut ini :

1. Untuk mengakses data dari tabel tunggal, gunakan klausa FOR dalam perintah seperti AVERAGE, BROWSE, atau LOCATE, atau menggunakan perintah SQL untuk memperbaharui tabel. Untuk perintah lengkap penggunaan klausa FOR, perhatikan tabel 1. - atau-
2. Untuk mengakses data dari tabel lebih dari satu, gunakan perintah-perintah SELECT-SQL DELETE-SQL, dan UPDATE- SQL

Tabel 1. perintah yang menggunakan klausa FOR yang dapat dioptimasi

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	

Jika menggunakan suatu klausa sebagai tambahan terhadap suatu optimasi ekspresi klausa FOR, maka scopenya harus diset ALL atau REST untuk mendapatkan manfaat dari Rushmore. Klausa NEXT dan RECORD tidak dapat digunakan. Sebab scope defaultnya adalah ALL untuk semua perintah, Rushmore bekerja ketika scope klausa dihilangkan. Rushmore dapat menggunakan beberapa index terbuka kecuali untuk memfilter dan Index UNIQUE..

Catatan : untuk kinerja optimal, jangan mengaktifkan index dari tabel

Buat index atau tags secara otomatis menetapkan order. Jika ingin mengambil keuntungan maksimum dari Rushmore data skalabesar maka harus ditetapkan order yang spesifik, SET ORDER TO untuk mematikan kendali index, kemudian menggunakan perintah SORT.

Index efektif untuk Optimisasi Query Rushmore

Rushmore tidak bisa mengambil keuntungan dari dari semua index. Jika menggunakan klausa FOR dalam perintah INDEX, Rushmore tidak bisa menggunakan index itu untuk optimisasi. Sebagai contoh, karena berisi suatu klausa FOR, statemen berikut ini tidak bisa dioptimasi :

INDEX ON ORDNUM FOR DISCOUNT > 10 TAG ORDDISC

Dengan cara yang sama, Rushmore tidak bisa menggunakan index tanpa dengan suatu Kondisi. Sebagai contoh, ekspresi berikut ini dapat dioptimasi :

INDEX ON DELETED() TAG DEL

Tetapi yang ini tidak bisa:

INDEX ON NOT DELETED() TAG; NOTDEL

Dalam kasus tertentu untuk mengeluarkan record yang dihapus dari suatu query, dengan menggunakan suatu index, seperti di contoh yang pertama lebih awal, akan mempercepat operasi dengan cara SET DELETE diaktifkan(ON)

Operasi Tanpa Optimisasi Query Rushmore

Operasi Perolehan kembali Data(*Data retrieval*) dilakukan tanpa Optimisasi Rushmore dalam situasi sebagai berikut:

1. Ketika Rushmore tidak bisa mengoptimalkan ekspresi klausa FOR dalam suatu perintah yang dapat mengoptimasi.
2. Ketika suatu perintah yang mungkin bermanfaat dari Rushmore berisi sebuah klausa WHILE.
3. Ketika memori terbatas/kecil. Perolehan kembali Data berlanjut, tetapi tidaklah dioptimalkan.

Mennonaktifkan (disable) Optimisasi Query Rushmore

Rusmore dapat dinonaktifkan, meskipun hal ini jarang dilakukan, Ketika akan digunakan suatu perintah yang menggunakan Rushmore, Visual Foxpro dengan seketika menentukan record-record yang memenuhi ekspresi klausa FOR. Record-record ini kemudian akan dimanipulasi oleh perintah-perintah.

Jika suatu perintah optimasi berpotensi untuk memodifikasi kunci index dalam klausa FOR, recordset yang sedang dioperasikan dengan Rushmore menjadi tidak berguna. Dalam hal ini, rushmore dapat dinonaktifkan untuk memastikan mempunyai banyak informasi dari tabel

Menonaktifkan Rushmore untuk perintah tunggal menggunakan klausa NOOPTIMIZE, sebagai contoh, perintah LOCATE ini tidak dioptimalkan:

LOCATE FOR Duedate< {^ 1998-01-01}; NOOPTIMIZE

Rushmore dapat dinonaktifkan secara global dengan memberi berintah *SET OPTIMIZE* : Untuk menonaktifkan Rushmore secara global, gunakan kode yang berikut:

SET OPTIMIZE OFF

Untuk mengaktifkan Rushmore secara global, gunakan kode yang berikut:

SET OPTIMAZE ON

Secara default Optimasi Rushmore adalah aktif/ON

Mengoptimasi Ekspresi-ekspresi Rushmore

Teknologi Rushmore tergantung pada keberadaan dari a *basic optimizable expression* dalam suatu klausa *FOR* atau dalam suatu klausa *SQL WHERE*. a *basic optimizable expression* dapat berbentuk keseluruhan ekspresi atau terlihat sebagai bagian dari suatu ekspresi. Dapat juga dikombinasikan ekspresi dasar membentuk suatu ekspresi optimimasi yang kompleks.

Membuat Ekpresi Dasar(*Basic*) Optimizeable

Basic optimizable expression, dapat berbentuk sebagai berikut :

eIndex relOp eExp

- atau-

eExpr relOp eIndex

Suatu Ekpresi Basic Optimizeable mempunyai karakteristik yang berikut:

1. eIndex sesuai dengan ekspresi yang ada pada index yang dibuat
2. eExpr ekspresi yagn dapat berisi variable dan fields dari tabel yang tidak berelasi.
3. relOp salah satu operator relational yang berikut:<,>,=,<=,>=,<>,#,==, atau!=. Dapat juga menggunakan fungsi-fungsi ISNULL (), BETWEEN(), atau INLIST() (atau SQL padanannya seperti IS NULL, dan seterusnya).

Disini dapat menggunakan BETWEEN() atau INLIST() dalam dua format berikut ini :

BETWEEN(eIndex, eExpr, eExpr)
atau

INLIST(eIndex, eExpr [, eExpr, eExpr, ...])

Catatan : ISBLANK() dan EMPTY() tidak dioptimasi oleh Rushmore.

Jika dibuat index firstname, custno, UPPER(LASTNAME), dan hiredate, masing-masing ekspresi berikut ini adalah optimizable:

firstname= "Fred"
custno>= 1000
UPPER(LASTNAME)= "SMITH"
hiredate< {^ 1997-12-30}

Suatu ekspresi yang dapat dioptimasi dapat berisi variabel dan fungsi yang mengevaluasi pada suatu nilai spesifik. Sebagai contoh, menggunakan index addr, jika menggunakan perintah *STORE " WASHINGTON AVENUE" TO cVar*, kemudian statemen berikut ini adalah juga ekpresi dasar yang teroptimasi:

ADDR= cVar
ADDR= SUBSTR(CVAR,8,3)

KAPAN SUATU QUERY DIOPTIMASI ?

Adalah penting untuk memahami ketika query akan dioptimasi atau tidak. Visual Foxpro mengoptimalkan kondisi-kondisi pencarian dengan melihat pada suatu perbandingan pasti antara sisi sebelah kiri dari suatu ekspresi yang difilter dan suatu ekspresi kunci index. Oleh karena itu, Rushmore dapat mengoptimasi suatu ekspresi hanya jika mencari 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 CUSTOMERS
INDEX ON UPPER(cu_name) TAG name

Perintah berikut ini tidak optimizable, sebab kondisi pencarian didasarkan hanya pada field cu_name saja, bukan pada suatu ekspresi yang diindexkan:

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

Sebagai ganti, perlu dibuat suatu ekspresi optimizable dengan menggunakan suatu perintah sebagai berikut, di mana ekspresi untuk pencarian sama dengan ekspresi indexed:

*SELECT * FROM customers WHERE; UPPER(cu_name) = "ACME"*

Tabel 2. Tiga level rushmore optimization

Optimization Level	Deskripsi
None	Query tidak bisa dioptimasi dengan Teknologi Rushmore
Partial	Beberapa ekspresi dalam query bisa dioptimasi dengan teknologi Rushmore. Index

	Tag yang digunakan untuk Rushmore Optimisasi didaftarkan/ditampilkan.
Full	Query dioptimasi secara penuh dengan teknologi Rushmore. Index Tag yang digunakan untuk Rushmore Optimisasi didaftarkan/ditampilkan.

Mengkombinasikan Ekspresi Dasar yang dapat diOptimasi

Ekspresi dapat dikombinasikan sederhana atau kompleks berdasar pada klausa FOR atau klausa WHERE untuk meningkatkan kecepatan perolehan kembali data, jika ekspresi FOR mempunyai karakteristik ekspresi dasar optimizable

Ekspresi dasar boleh jadi optimizable. Ekspresi dasar juga dapat dikombinasinasikan dengan menggunakan operator logika AND, OR, dan NOT untuk membentuk suatu ekspresi klausa FOR yang kompleks yang mungkin juga optimizable. Suatu ekspresi dibuat dengan suatu kombinasi dari ekspresi dasar optimizable secara penuh optimizable. Jika satu atau lebih ekspresi yang dasar tidak optimizable, ekspresi yang kompleks boleh jadi secara parsial optimizable atau tidak optimizable sama sekali.

Satu set aturan menentukan jika suatu ekspresi terdiri atas ekspresi non-optimizable atau optimizable dasar secara penuh, secara parsial optimizable, atau tidak optimizable. Tabel 3 berisi aturan-aturan Rushmore Optimisasi Query.

Tabel 3. Kombinasi Ekspresi dasar

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

Operator AND dapat digunakan untuk berkombinasi dua ekspresi optimizable ke dalam satu ekspresi yang optimizable:

FIRSTNAME= "FRED" AND HIREDATE< ; {^ 1997-12-30}&& Optimizable

Dalam contoh ini, operator OR mengkombinasikan suatu ekspresi dasar optimizable dengan suatu ekspresi yang tidak optimizable untuk membuatn suatu ekspresi yang tidak optimizable:

FIRSTNAME = "FRED" OR "S" $;LASTNAME && Not optimizable

Penggunaan operator NOT pada suatu ekspresi optimizable membuat suatu ekspresi menjadi optimizable secara penuh :

NOT FIRSTNAME = "FRED" && Fully optimizable

Tanda kurung dapat juga digunakan untuk menggolongkan kombinasi dari ekspresi dasar.

Kombinasi Ekspresi Kompleks

Sama halnya dengan mengkombinasikan ekspresi dasar, dapat juga mengkombinasikan ekspresi kompleks untuk membuat suatu ekspresi lebih rumit yang optimizable, secara parsial optimizable, atau tidak optimizable. Kemudian bisa mengkombinasikan ekspresi yang lebih rumit ini untuk menciptakan ekspresi yang boleh jadi secara penuh atau secara parsial optimizable, atau tidak optimizable sama sekali. Tabel 4 menguraikan hasil kombinasi ekspresi kompleks. Aturan ini juga berlaku untuk ekspresi yang dikelompokkan dengan tanda kurung.

Tabel 4. Ekpresi kompleks

Expression	**Operator**	**Expression**	**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

Ekspresi dapat dikombinasikan secara optimizable penuh dengan operator OR untuk membuat satu ekspresi yang adalah juga optimizable secara penuh :

* Fully-optimizable expression

  (FIRSTNAME = "FRED" AND HIREDATE < {^1997-12-30})OR (LASTNAME = "" AND HIREDATE > {^1996-12-30})

  Untuk membuat ekspresi optimizable secara parsial, kombinasikan suatu ekspresi optimizable secara penuh dengan suatu ekspresi yang tidak optimizable. Di dalam contoh, operator AND digunakan untuk mengkombinasikan ekspresi tersebut:

* Partially-optimizable expression

  (FIRSTNAME = "FRED" AND HIREDATE; < {^1997-12-30}) AND "S" $ LASTNAME

Ekspresi optimizable secara parsial dapat dikombinasikan untuk membuat satu ekspresi yang juga optimizable secara parsial :

* Partially-optimizable expression

  (FIRSTNAME = "FRED" AND "S" $ LASTNAME) OR (FIRSTNAME = "DAVE"; AND "T" $ LASTNAME)

  Kombinasikan ekspresi yang tidak optimizable untuk membuat suatu ekspresi yang juga tidak optimizable :

* Expression that is not optimizable

  ("FRED" $ FIRSTNAME OR "S" $ LASTNAME) OR ("MAIN" $ STREET OR; "AVE" $ STREET)

Dalam paper yang berjudul "*Rushmore Query Optimization in the Jet Database Engine Version 2.0*" yang disusun oleh Neil Black, disebutkan Operasi-operasi yang mungkin dengan Rushmore

Index Intersection

Pemecahan suatu query yang menggunakan Index Intersection melibatkan membaca sekilas berbagai index untuk record yang sesuai dengan kriteria seperti:

column1=<expr>AND column2=<expr>

Penggunaan Rushmore melibatkan penggiunaan index on column1 and the index on column2 untuk menemukan record yang cocok. Record yang diasilkan dari masing-masing index adalah intersection untuk menemukan record yang cocok dengan kriteria keduanya

Sebagai contoh, suatu tabel yang berisi informasi demografis tentang semua orang-orang di dalam Kota besar New York. Tabel;Meja berisi informasi umur tiap orang, tingginya, berat/beban, warna rambut, dan warna mata. Misalnya ditulis suatu query untuk temukan semua orang-orang dengan rambut wanita berambut pirang(blonde) dan mata biru. Kriteria akan terlihat sebagai berikut :

hair_color = "blonde" AND;
eye_color= "blue"

Query ini akan diselesaikan menggunakan Rushmore index intersection. Index pada

hair_color akan digunakan untuk menemukan semua wanita berambut blode(pirang) dan kemudian index pada eye_color akan digunakan untuk menemukan semua orang-orang yang bermata biru(blue). Hasil dari tiap index pencarian akan diinterseksikan untuk menemukan semua orang-orang dengan karakteristik keduanya.

Index Union

Penyelesaian suatu query yang menggunakan Union Index melibatkan pembacaan beberapa index untuk mencocokan recods sesuai kriteria seperti :

column1=<expr>OR column2=<expr>

Penggunaan Rushmore melibatkan penggunaan index on column1 and the index on column2 untuk menemukan record yang cocok. Record hasil dari masing-masing index kemudian diunionkan bersama-sama untuk menemukan record yang sesuai dengan salah satu kriteria atau lainnya.

Sebagai contoh, mempertimbangkan kembali tabel yang berisi informasi demografis tentang semua orang-orang di Kota New York. Bagaiman suatu query ditulis untuk menemukan semua orang-orang dengan rambut hitam(black) atau bermata hijau. Kriteria akan terlihat seperti:

hair_color = "black" OR;
eye_color = "green"

Query ini akan diselesaikan menggunakan Rushmore Index Union. Index pada hair_color akan digunakan untuk menemukan semua orang-orang yang berambut hitam(black) dan kemudian index pada eye_color akan digunakan untuk menemukan semua orang-orang yang bermata hijau. Hasil dari pencarian tiap index kemudian digabungkan(unioned) bersama-sama untuk menemukan semua orang dengan karakteristik lainnya.

SEBERAPA CEPATKAH ?

Sebuah Set Query Benchmarks adalah suatu satuan pengukuran(benchmark) standard industri yang digunakan untuk menentukan kecepatan query dengan menggunakan sekumpulan operasi relasional pada database berskala besar. Berikut adalah beberapa contoh percobaan yang telah dilakukan oleh Neil Black dengan menggunakan Set Query Benchmark dengan mesin Jet 1.x dan jet 2.0 untuk mengukur kecepatan query untuk menunjukkan bagaimana Rushmore dapat mempercepat pengolahan query.

Q2a - K1K:

SELECT Count(*)
FROM bench
WHERE (bench.K2 = 2) AND;
(bench.K1K = 3)

Query ini melaksanakan perhitungan dari 46 record yang sesuai dengan kriteria khusus yang ditetapkan dari total 100,000 record dalam tabel. Menggunakan Mesin Jet 1.X, query ini menggunakan 1.59 detik untuk eksekusi. Menggunakan Mesin Jet 2.0, query yang sama menggunakan 0.55 detik untuk eksekusi. Terjadi peningkatan performansi/kecepatan 300%.

Q2a - K100:

SELECT Count(*)
FROM bench
WHERE (bench.K2=2) AND;
(bench.K100 =3)

Query ini melaksanakan perhitungan dari 490 recordyang sesua dengan kriteria khususyang ditetapkan dari total 100,000 record dalam tabel. Menggunakan Mesin Jet 1.X, query ini menggunakan 11.53 detik untuk eksekusi. Menggunakan Jet 2.0, query yang sama mengambil 0.77 detik untuk melaksanakan. Terjadi peningkatan sebanyak 1500% kali.

Q24 - 2-4:

SELECT bench.KSEQ, bench.K500K
FROM bench
WHERE (bench.K100 > 80) AND;
(bench.K10K Between 2000 And 3000) AND;
(bench.K5=3)

Query ini menemukan 379 record yang cocok dengan kriteria yang ditetapkan dari total 100,000 record dalam tabel. Menggunakan Jet 1.X, query ini menggunakan 223.93 detik untuk eksekusi. Menggunakan Jet 2.0, query yang sama mengambil 1.7 detik untuk eksekusi. Terjadi peningkatan 13100% kali.

TIPE DATA YANG DAPAT DIGUNAKAN RUSHMORE

Rushmore Query akan bekerja dengan Native Jet Data yaitu MDB file, seperti halnya data Foxbase dan dBase yaitu DBF file. Rushmore tidak bisa digunakan pada datasource ODBC, ketika suatu query ini dikirim ke data source ODBC untuk diproses, maka tidak akan diproses secara local oleh Jet.

KESIMPULAN

Salah satu keuntungan optimisasi query Rushmore adalah bahwa mesin jet Query sekarang dapat menggunakan index lebih dari satu tiap tabel untuk menyelesaikan suatu query. Didalam Jet 1.X, hanya satu index pada satu waktu yang bisa digunakan untuk menyelesaikan suatu query. Kemampuan index dari tabel besar dapat menghasilkan kinerja besar ketika Rushmore Optimisasi Query digunakan.

Dengan kinerja akses data yang lebih cepat maka diharapkan selanjutnya informasi yang dibutuhkan dapat dihasilkan dan digunakan dengan baik oleh pihak manajemen untuk mengambil keputusan dalam meraih keuntungan yang besar.

DAFTAR PUSTAKA

1. Black, Neil W., 1993, *Rushmore Query Optimization in the Jet Database Engine Version 2.0*, Microsoft Corporation
2. http://www.foxprohistory.org/*rushmore.htm*
3. http://msdn2.microsoft.com/en-us/library/*1f5d2sa3(VS.80).aspx*
4. Jogiyanto, HM., 1999, *Analisis dan Desain Sistem Informasi*, Andi Offset,Yogyakarta
5. Kadir A., 2002, *Pengenalan Sistem Informasi*, Andi Offset,Yogyakarta
6. Martina I., *Ir, 2002, 36 Jam Belajar Komputer Visual Foxpro 6.0*, PT. Elexmedia Komputindo, Jakarta