***Research** : Journal of Computer, information system, & technology management* **Online** ISSN: 2615-7357
Vol. 1, **No. 2**. **April 2018, Pages 27-31** **Print** ISSN: 2615-7233
==========

OPTIMALISASI QUERY DALAM BASIS DATA MY SQL MENGGUNAKAN INDEX

Ridho Pamungkas
Jurusan Sistem Informasi, Fakultas Teknik, UNIPMA, Madiun
e-mail: **ridho.pamungkas@unipma.ac.id**

Abstrak—Kualitas sebuah sistem informasi sangat terpengaruhi oleh kualitas desain basis data. Terkadang seorang programmer melupakan optimalisasi sebuah query di MySQL, padahal jika suatu sistem yang telah berjalan opltimalisasi sangatlah penting ketika jumlah data menjadi besar. Di dalam MySQL menyediakan fungsi index pada table dan mempercepat proses pencarian dalam suatu basis data. Index pada intinya akan berguna apabila digunakan sesuai dengan kepentingan dan kebutuhan karena apabila tidak, index yang dibuat secara sembarangan dan banyak akan membuat lambatnya akses ke database. Sehingga harus diperhatikan lagi aturan – aturan dan analisis yang baik sebelum membuat index pada database.

***Kata kunci:** Optimalisasi, MySQL, Index.*

I. PENDAHULUAN

Basis data merupakan komponen yang penting dalam sebuah sistem informasi modern. Sebagian besar sistem informasi dewasa ini hampir semuanya menggunakan Relational Database Management System (RDBMS), Sistem Basis Data Relational. Software RDBMS yang umum digunakan dalam sistem informasi adalah Oracle, Ms SQL Server, PostgreSQL, DB2, FirebirdSQL atau MySQL[1].

Database sendiri merupakan kumpulan data yang pada umumnya menggambarkan aktivitas-aktivitas dan pelakunya dalam suatu organisasi, misalkan database universitas akan berisi: mahasiswa, dosen, kuliah dan lain-lain.[2]

SQL (*Structured Query Language*) merupakan bahasa yang terstruktur utuk menggunakan atau mengakses data pada *database* dan entitas – entitas yang ada pada *database* terserbut. SQL juga merupakan bahasa standar yang digunakan dalam berbagai *database* yang ada sehingga mudah untuk menggunakannya walaupun berpindah dari satu *database* ke *database* lainnya [3].

Index pada SQL-Server dikatakan sebagai objek database yang dibuat berdasarkan tabel dan kolom. *Index* pada *database* digunakan untuk mencari nilai kolom pada tabel tertentu dengan cepat karena tanpa menggunakan *index* maka *database* harus melakukan pencarian dari mulai tabel baris pertama hingga tabel baris terakhir dan akan memakan banyak waktu [1]. Selain itu juga, tanpa *index* akan membutuhkan kapasitas memori yang besar apabila dilakukan pencarian pada tabel yang memiliki baris yang banyak. Maka dapat dikatakan bahwa *index* ini mirip dengan daftar isi pada sebuah buku.

II. LANDASAN TEORI

Seperti dibahas sebelumnya, *index* pada *database* sama halnya seperti daftar isi pada sebuah buku untuk memudahkan dalam pencarian data. Karena apabila tidak menggunakan *index* pada suatu *database*, maka *database* akan melakukan *scanning* atau pencarian dari setiap baris pada tabel yang diinginkan. *Scanning* atau pencarian tersebut akan memakan waktu yang lama dan akan memperlambat kinerja suatu *database*.

Dalam SQL dapat membuat *index* dengan satu kolom atau beberapa kolom apabila *index* tersebut dipanggil. Jadi hasil dari *index* tersebut dapat berupa satu kolom (*column index*) dari sebuah tabel, atau juga beberapa kolom (*multiple-column*) dari sebuah tabel.

2.1. *Column Indexes* (*Index* Satu Kolom)

Semua tipe data MySQL dapat di-*index*-kan. Menggunakan *index* pada

kolom yang sesuai merupakan cara terbaik untuk meningkatkan performa dari operator *SELECT*.

Nomor maksimal *index* setiap tabel dan panjang *index* terdefinisi berdasarkan penyimpanan (*Storage Engine*).

2.2. *Multiple-Column Indexes* (*Index* Beberapa Kolom)

MySQL dapat membuat *index* gabungan (maksudnya membuat *index* dengan query memanggil dua kolom dari sebuah data atau tabel). *Multiple-Column index* yang dibuat maksimum 16 kolom.

Multiple-Column index dapat dikatakan sebagai *array* yang mengandung nilai – nilai yang digabungkan dengan nilai pada kolom yang dibuat *index*.

MySQL menggunakan *index* untuk melakukan operasi-operasi sebagai berikut:

a. Untuk mencari baris-baris yang sesuai dengan klausa *WHERE* dengan cepat

b. Untuk mengeliminasi baris-baris dari pertimbangan. MySQL akan mencari *index* dengan nomor baris yang terkecil.

c. Untuk mengambil baris pada kolom yang telah digabungkan sebelumnya.

d. Dalam beberapa kasus, *query* dapat dioptimasi untuk mengambil nilai-nilai tanpa membuka tabelnya terlebih dahulu.

III. METODE

Dalam penelitian ini menggunakan metode eksperimental. Metode eksperimental merupakan salah satu dari jenis jenis metode penelitian. Metode eksperimental memungkinkan peneliti memanipulasi dan mengubah-ubah variabel dan meneliti akibat-akibatnya. Pada metode eksperimental ini variabel-variabel dikontrol sedemikian rupa, sehingga variabel luar yang mungkin dapat mempengaruhi dapat dihilangkan.

Metode eksperimental bertujuan untuk mencari dan mendapatkan hubungan sebab akibat dengan merubah atau memanipulasikan satu atau lebih variabel, pada satu atau lebih kelompok eksperimental dan kemudian membandingkan hasilnya dengan kelompok kontrol yang tidak mengalami manipulasi. Manipulasi adalah mengubah secara sistematis sifat-sifat atau nilai-nilai variabel bebas. Kontrol merupakan kunci metode eksperimental, sebab tanpa kontrol manipulasi dan observasi akan menghasilkan data yang diragukan kebenarannya.

IV. HASIL

Sebelum membuat *index* pada sebuah *database*, harus membuat dan memikirkan strategi yang tepat karena tidak semua data atau harus menggunakan *index*. Strategi dalam membuat *index* diantaranya:

a. Desain *index*

Desain *index* yaitu seperti menentukan kolom untuk digunakan, memilih jenis *index* yang akan dipakai (misalnya, *clustered* atau *non-clustered*), memilih opsi *index* yang tepat dan menentukan *filegroup* atau penempatan skema patisi.

b. Menentukan metode pembuatan yang terbaik

Index dibuat dengan cara:

1) Mendefinisikan *primary key* atau *constraint* yang unik pada kolom, dengan menggunakan *CREATE TABLE* atau *ALTER*. SQL secara otomatis membuat *index* yang unik untuk melaksanakan persyaratan keunikan dari *PRIMARY KEY* atau *constraint* yang unik, kecuali sebuah *clustered index* yang sudah ada ditabel atau menentukan *non-clustered* yang unik pada *index*. Secara *default*, *clustered* yang unik pada *index* dibuat untuk melaksanakan sebuah *PRIMARY KEY constraint* kecuali *clustered* yang unik pada *index* secara eksplisit ditentukan dan *clustered index* pada tabel tidak tersedia.
2) Membuat *index* yang terlepas dari batasan dengan menggunakan perintah *CREATE INDEX* atau Dialog ***New***

Index pada SQL. Yang dilakukan yaitu menentukan nama, tabel *index* dan kolom *index* yang berlaku. Pilihan *index* dan lokasi *index, filegroup* atau skema partisi, juga dapat ditentukan. Secara *default*, sebuah *non-clustered, non-unique index* dibuat apabila opsi *clustered* atau batasan unik tidak ditentukan. Untuk membuat *index* yang lebih spesifik (terfilter) gunakan klausa *WHERE*.

3) Membuat *index*

Membuat *index* di tabel yang kosong tidak memiliki implikasi performa pada saat *index* dabibuat, namun, performa akan terpengaruh ketika data dimasukan ke tabel.

Menciptakan *index* pada tabel besar harus direncanakan denga hati-hati sehingga performa database tidak terhambat. Cara yang lebih disukai untuk membuat *index* pada tabel besar yaitu mulai dengan *clustered index* dan kemudian membangun *non-clustered index* [8].

Sebuah *index* dapat dibuat dalam sebuah tabel untuk mencari data lebih cepat dan efisien. Pengguna tidak dapat melihat *index,* merek hanya menggunakan untuk kecepatan dalam pencarian / *query*. Berikut ini adalah sintak yang digunakan untuk membuat *index* pada sebuah tabel. Diperbolehkan nilai ganda:

```
CREATE INDEX index_name
ON table_name(column_name)
```

Query diatas ditujukan untuk membuat *index* dengan nama *index* yang akan dibuat pada tabel apa dan kolom keberapa.

Berikut ini adalah sintak untuk membuat *index* yang unik pada sebuah tabel. Tidak diperbolehkan nilai ganda pada *index* ini:

```
CREATE UNIQUE INDEX index_name
ON table_name(column_name)
```

Query diatas ditujukan untuk membuat *index* yang unik yang tidak diperbolehkan nilai ganda dengan nama *index* yang akan dibuat pada tabel apa dan kolom keberapa

.

Selain dapat membuat sebuah *index,* dapat juga melakuakan *drop* atau menghapus sebuah *index* yang telah dibuat sebelumnya, berikut ini adalah sintaksnya:

```
DROP INDEX index_name
ON table_name
```

Query diatas ditujukan untuk menghapus sebuah *index* yang telah dibuat sebelumnya dengan menggunakan operator *DROP INDEX* lalu dilanjutkan dengan nama *index* yang akan dihapus beserta nama tabel yang akan dihapus.

Selain membuat *index* dan menghapus *index*, dapat juga dilakukan penonaktifan sebuah *index* yang telah dibuat. Maksud dari penonaktifan ini adalah mencegah pengguna untuk mengakses *index, clustered index* atau tabel pokok. Metadata dan data statistik *index* tetap ada pada *non-clustered index*. Menonaktifkan *clustered index* atau *non-clustered index* hanya menghapus data fisik dari *index*.

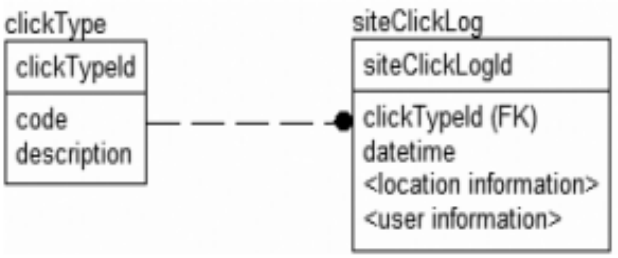
Menonaktifkan *clustered index* mencegah akses ke data; data tetap ada pada tabel tetapi tidak tersedia untuk operasi DML hingga *index* tersebut dihapus atau dibuat kembali. Untuk membuat kembali dan mengaktifkan kembali *index* dapat menggunakn operasi dengan statemen *ALTER INDEX REBUILD* atau *CREATE INDEX WITH DROP_EXISTING*

c. Keuntungan Menggunakan Skenario *Indexed View*

1) *Foreign Key Indexes*

Kolom *Foreign key* merupakan *special case* dimana membutuhkan *index* dari

beberapa urutan. Ini karena pembuatan *foreign key* sehingga bisa menyesuaikan baris dalam sebuah tabel ke baris di tabel lainnya. Hal ini penting untuk memastikan bahwa setiap mendeklarasikan *foreign key constraint*, ada potensi untuk keperluan *index* bilamana memiliki paren tabel dan ingin melihat anak dari baris. Kasus penting dan *special* dimana tipe aksesnya *essential* (penting) ketika menghapus *parent row* di beberapa relasi, bahkan salah satu jenis domain.



Gambar 1. Relasi yang Menggunakan Foreign Key

2) *Indexed Views*

Meng-*index*-an sebuah view pada dasarnya mengambil struktur virtual dari *view* dan membuatnya menjadi entitas fisik. Data untuk menyelesaikan *query-query* dengan *view* adalah hasil sebagai data yang dimodifikasi dalam tabel. *Indexed view* memberikan kemampuan untuk membangun rangkuman tabel tabel tanpa beberapa macam operasi manual atau *trigger*.

Keuntungannya dua kali lipat ketika menggunakan index view. SQL secara otomatis mempertimbangkan penggunaan sebuah indexed view setiap kali mengeksekusi beberapa query, bahkan jika belum menetukan view tertentu untuk digunakan dan bahkan jika query tidak mereferensikan sebuah view.

d. Indeks *Oracle*: Kinerja Database

Indeks *Oracle* adalah pengindeksan database yang tepat merupakan faktor penting bagi kinerja database. Kebanyakan *Oracle* database memiliki ratusan atau bahkan ribuan indeks. Ini sejumlah besar indeks dan kompleksitas mereka membuat penyetelan indeks dan memantau tugas yang sulit untuk DBA. Seiring dengan berjalannya waktu, bahkan indeks awalnya efisien dapat menjadi tidak efisien karena distorsi indeks disebabkan oleh berbagai perubahan data dalam tabel diindeks.

Bagaimana mengelola indeks *Oracle* dan apa pilihan yang berbeda yang tersedia untuk menggunakannya?

Indeks yang logis dan fisik independen dari data dalam tabel yang terkait. DBA dapat membuat atau menjatuhkan indeks kapan saja tanpa mempengaruhi tabel dasar atau indeks lainnya. Jika DBA turun indeks, semua aplikasi terus bekerja. Namun, akses ke data yang sebelumnya diindeks mungkin lebih lambat. Indeks, karena strukturnya independen, membutuhkan ruang penyimpanan.

Oracle secara otomatis menjaga dan menggunakan indeks setelah mereka diciptakan. *Oracle* secara otomatis mencerminkan perubahan data, seperti menambahkan baris baru, memperbarui baris, atau menghapus baris, di semua indeks yang relevan dengan tidak ada tindakan tambahan oleh pengguna.

Teks *Oracle* mendukung terciptanya tiga jenis indeks *Oracle* tergantung pada aplikasi *Oracle* dan sumber teks. DBA menggunakan pernyataan CREATE INDEX untuk membuat semua jenis teks indeks *Oracle*.

V. KESIMPULAN

Index pada *database* sangat berguna untuk meningkatkan performa SQL khususnya bagi *database* yang memiliki banyak tabel dan memiliki banyak baris pada setiap tabel. Karena apabila tidak menggunakan *index* pada *database* tersebut, maka pada saat pencarian sebuah data pada *database* akan memakan banyak waktu karena harus melakukan pencarian atau *scanning* pada semua baris pada tabel. Selain itu

juga akan memakan banyak memori pada *storage*.

Penggunaan *index* di SQL pada *Oracle* sangat berguna karena *Oracle* mendukung lebih banyak tipe *index* dibandingkan pada MySQL. Seperti contoh, pada *Oracle* dapat pencarian frase dengan menggunakan *text index*.

Tetapi penggunaan *index* ini harus tepat karena tidak semua *database* memerlukan *index* sehingga apabila akan membuat sebuah *index* pada *database*, diperlukan perencanaan dan strategi yang cermat. Karena apabila tidak direncakan terlebih dahulu dan tidak memiliki strategi yang baik, *index* yang dibuat akan memperumit pengguna dalam mencari sebuah data yang mereka cari. Sehingga akan menghasilkan data yang salah bagi pengguna lain yang membutuhkan data atau informasi dari *database* tersebut.

Jadi, *index* pada *database* sangat berguna bagi pengguna yang menggunakannya tetapi harus hati – hati dalam menentukan tabel mana saja yang akan dibuat *index* dan perlukah tabel tersebut dibuat *index* atau tidak, agar tidak memperlambat performa dari sebuah SQL. Karena terlalu banyak *index* pun akan memperlambat kinerja sebuah *database* pada SQL.

DAFTAR PUSTAKA

[1] Raharjo, Suwanto, "Integrity Constraint Basis Data Relasional Dengan Menggunakan pl/pgsql Dan Check Constraint " in *Seminar Nasional Teknik dan Manajemen Industri*, Malang, 2015, pp. IV-22 - IV 27.

[2] Bambang Hariyanto, *Rekayasa Sistem Berorientasi Objek*. Bandung: Informatika, Bandung, 2004.

[3] Z.A. Hasibuan, *Metodologi Penelitian pada Bidang Ilmu Komputer dan Teknologi Informasi*. Jakarta, Infonesia: Fakultas Ilmu Komputer, Universitas Indonesia, 2007.

[4] B.Rassameethes, "Analysis and Integrastion of Thailand ICT Master Plan," *International Journal of Synergy and Research*, p. 77, 2012.

[5] Firdayanti Restika, "Persepsi Resiko Melakukan E-Commerce dengan Kepercayaan Konsumen dalam Membeli Produk Fashion Online," *Journal of Social and Industrial Psychology*, vol. I, no. 1, pp. 1-7, 2012.

[6] Robinson Pearce, *Manajemen Strategis, Formulasi Implementasi dan Pengendalian*. Jakarta: Salemba Empat, 2008.