# PENINGKATAN UNJUK KERJA MYSQL DALAM INPUT DAN OUTPUT DATA

**Jaeni**[1]

**Abstraksi**

Proses input data dan output data pada mySQL membutuhkan waktu proses. Waktu proses yang baik adalah waktu sedikit yang mempercepat seorang administrator web dan pengguna internet ketika memasukkan dan menampilkan data.

**Kata Kunci**: mysql, input, output, sql.

## 1. Pendahuluan

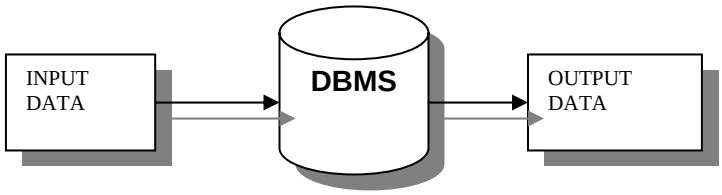
MySQL adalah basis data *multi-user* yang menggunakan bahasa *Sructured Query Language* (SQL). MySQL dalam operasi *client-server* melibatkan server daemon MySQL di sisi server dan berbagai macam program serta library yang berjalan di sisi client. MySQL mampu untuk menangani data yang cukup besar. Perusahaan yang mengembangkan MySQL yaitu Tcx, mengaku mampu menyimpan data lebih dari 40 database, 10.000 tabel dan sekitar tujuh juta baris, totalnya kurang lebih seratus gigabytes data (Manual MySQL Reference ).

SQL adalah bahasa standar yang digunakan untuk mengakses *database* server. Bahasa ini pada awalnya dikembangkan oleh IBM, namun telah diadopsi dan digunakan sebagai standar industri. Dengan menggunakan MySQL, proses akses database menjadi lebih *user-friendly*.

Dalam basis data proses input dan output adalah proses yang sangat mendasar. Data-data yang ada **dimasukkan** ke dalam suatu

[1] Staff Pengajar STMIK AMIKOM Yogyakarta

software basis data dan akan disimpan dalam software tersebut kemudian pada suatu waktu data itu akan **ditampilkan** kembali saat dibutuhkan. Begitulah terus-menerus suatu sistem basis data bekerja.




**Gambar 1. Proses input dan output data pada basis data**

Proses input dan output data memakan waktu yang tidak sedikit jika data tersebut berjumlah besar. Untuk memperoleh waktu seefisien mungkin dalam proses input dan output data diperlukan suatu teknik. Teknik inilah yang akan dibahas untuk menangani masalah seputar input dan output data pada basis data khususnya basis data berbasis *web*.

## 2. Pembahasan

### Efisiensi Input dan Output Data

Banyak sekali cara yang dapat ditempuh untuk meningkatkan unjuk kerja query khususnya dalam masalah input dan output data. Sangat lah sulit untuk menggambarkan luasnya cakupan masalah tersebut dalam satu tulisan saja. Berdasarkan hal itu, maka skripsi ini akan mencoba untuk menjelaskan dan memberi beberapa contoh untuk meningkatkan unjuk kerja MySQL dalam input dan output data. Yang ingin dicapai dalam bahasan ini bukan merupakan bahasan lengkap mengenai peningkatan unjuk kerja query pada MySQL sendiri, namun lebih kepada penjelasan pokok dan memberikan beberapa percontohan dari beberapa macam cara peningkatan unjuk kerja khususnya dalam masalh input dan output data.

**Peningkatan Unjuk Kerja Query dalam Input Data**

Peningkatan unjuk kerja query yang pertama adalah dalam masalah input data. Proses ini akan memakan waktu yang lama jika data yang dimasukkan dalam jumlah yang sangat banyak. Biasanya untuk memasukkan data pada MySQL digunakan perintah INSERT.

Untuk memasukkan data, MySQL menyediakan perintah lain yang lebih cepat yaitu menggunakan perintah ***load data infile***. Perintah load data infile membaca baris-baris dari sebuah file teks kemudian memasukkannya ke dalam tabel dengan kecepatan yang sangat tinggi (MySQL Manual Reference,2007). File teks tersebut berisi data yang akan dimasukkan. Tiap baris mewakili satu record dan tiap-tiap nilai dari field dipisahkan dengan tabulasi.

Sintak dasar dari perintah load data infile adalah sebagai berikut :

LOAD DATA INFILE ‘nama_file' INTO TABLE nama_tabel;

Perintah di atas hanya berlaku untuk satu buah tabel saja. Jika terdapat beberapa tabel maka input data tidak bisa dilakukan hanya dengan menggunakan perintah load data infile. Memasukkan data pada beberapa tabel yang saling berelasi juga dapat menghabiskan waktu yang sangat banyak. Lebih-lebih jika data tersebut dalam jumlah yang sangat besar.

**Peningkatan Unjuk Kerja Query dalam Output Data**

**a. Indeks pada Single-Table Query**

***Single-table query*** adalah proses query yang hanya melibatkan tabel tunggal saja. Bila suatu atribut atau kolom pada suatu tabel tidak mempunyai indeks, maka akan terjadi *full-table scan* ketika perintah untuk menampilkan data (SELECT) atau yang lainnya dilakukan. *Full-table scan* terjadi karena basis data membaca satu per satu *record* dalam tabel untuk menjalankan query. Dengan demikian akan tejadi suatu proses yang cukup lama untuk menjalankan suatu query.

*Full-tabel scan* dapat dihindari dengan cara membuat indeks pada kolom yang akan disebutkan dalam klausa WHERE pada query. Indeks akan menyediakan akses secara langsung ke dalam data.

Dengan penambahan indeks pada suatu data akan menambah kecepatan pengaksesan data tersebut.

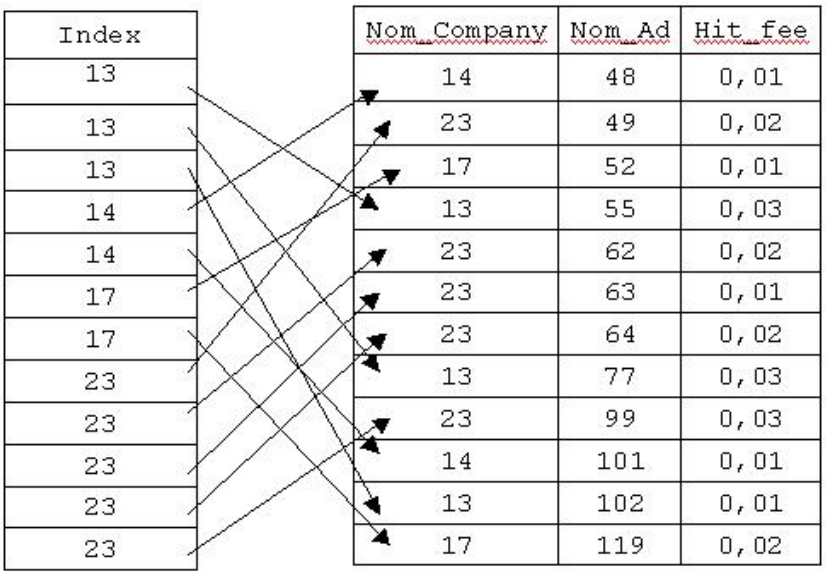
Untuk mengetahui keuntungan indeks yang dapat mempercepat query akan dilakukan perbandingan antara tabel yang berindeks dan tak berindeks. Sebagai contoh, gambar 3.1 menunjukkan tabel ad yang tak berindeks sehingga jika dicari nilai-nilai tertentu pada kolom company, maka nilai-nilai yang ada di masing-masing baris harus diperiksa. Ini lah yang disebut *full-tabel scan*.

Table ad

| Nom_Company | Nom_Ad | Hit_fee |
|---|---|---|
| 14 | 48 | 0,01 |
| 23 | 49 | 0,02 |
| 17 | 52 | 0,01 |
| 13 | 55 | 0,03 |
| 23 | 62 | 0,02 |
| 23 | 63 | 0,01 |
| 23 | 64 | 0,02 |
| 13 | 77 | 0,03 |
| 23 | 99 | 0,03 |
| 14 | 101 | 0,01 |
| 13 | 102 | 0,01 |
| 17 | 119 | 0,02 |

**Gambar 2. Tabel Ad yang tak berindeks**

Gambar 2 menunjukkan tabel yang sama, tetapi dengan tambahan sebuah indeks pada kolom nom_company. Indeks berisi sebuah masukan untuk masing-masing baris dalam tabel ad, akan tetapi nilai nom_company pada indeks telah terurut.




**Gambar 3 Tabel Ad yang berindeks**

Misal akan dicari semua baris di mana nom_company bernilai 13. Pencarian dimulai dengan memeriksa indeks awal sehingga didapat tiga baris pertama yang sesuai dengan kriteria. Selanjutnya pemeriksaan akan mencapai nilai 14. Nilai ini lebih tinggi dari kriteria yang ditetapkan. Nilai indeks adalah nilai yang sudah terurut sehingga setelah nilai yang disyaratkan yaitu 13 telah selesai, pencarian akan berhenti. Dengan teknik ini tentu saja akan mempercepat perolehan data dan menghemat waktu dalam pencarian.

**b. Indeks pada Multiple-Table Query**

***Multiple-table query*** adalah proses query yang melibatkan lebih dari satu tabel yang saling berelasi. Pada *single-table query* indeks dapat mempercepat pencarian data dengan menghilangkan *full table scan*. Begitu pula pada *multiple-table query*, indeks bahkan lebih berarti ketika query yang melibatkan perintah join dijalankan.

Untuk mengetahui keuntungan indeks pada *multiple-table query* adalah dengan memisalkan adanya tiga tabel yang tak berindeks. Sebut saja t1, t2, dan t3. Masing-masing tabel secara berurutan berisi kolom c1, c2, dan c3. Masing-masing kolom terdiri dari seribu baris. Query untuk mencari semua kombinasi dari baris-baris tabel yang mana mempunyai nilai yang sama adalah sebagai berikut :

```
SELECT c1, c2, c3
FROM t1, t2, t3
WHERE c1=c2 AND c1=c3
```

Hasil dari query di atas seharusnya seribu baris, di mana masing-masing baris memiliki kolom dengan tiga nilai yang sama. Jika query diproses tanpa indeks, maka sangat lama untuk mencari baris yang berisi nilai yang sama. Dengan kata lain, ini akan berkonsekuensi harusnya mencoba semua kombinasi untuk mencari kombinasi yang sesuai dengan klausa WHERE. Jumlah kombinasi yang mungkin adalah 1000x1000x1000 atau **satu milyar**. Hal ini tentu saja membuang waktu yang sangat banyak sekali lebih-lebih jika data yang dicari sangat sedikit dari jumlah kombinasi.

Jika masing-masing tabel diberi indeks, pencarian akan menjadi sangat cepat karena indeks mengijinkan query untuk memproses seperti ini :

1. Menyeleksi baris pertama dari tabel t1 dan melihat nilai baris tersebut.

2. Menggunakan indeks pada tabel t2 yang secara langsung (tanpa *full table scan*) akan mencek baris yang nilainya sama dengan nilai dari tabel t1. Dengan cara yang sama, indeks pada tabel t3 akan mencek secara langsung pula baris yang sesuai nilanya dengan nilai dari tabel t1.
3. Memproses baris berikutnya pada tabel t1 dan melakukan pengecekan kembali pada tabel t2 dan t3. Hal ini berulang terus-menerus sampai semua baris yang ada pada tabel t1 selesai diperiksa.

Pada kasus ini *full table scan* masih dilakukan pada tabel t1, akan tetapi indeks dapat dikerjakan pada tabel t2 dan t3 sehingga dalam kasus ini secara teori query dapat berjalan kira-kira satu juta kali lebih cepat dari pada tidak memakai indeks sama sekali.

## 3. Penutup

Dari pembahasan, percobaan, dan penjelasan di depan dapat diambil beberapa kesimpulan, yaitu :

a. Efisiensi input data menggunakan MySQL pada basis data berbasis web untuk *single-table query* dapat diperoleh dengan memakai perintah LOAD DATA INFILE. Keuntungannya adalah seorang administrator cukup sekali saja menjalankan *submit* sehingga dapat memperhemat tenaga dan kerja serta membutuhkan waktu proses yang tidak terlalu lama dibandingkan memakai INSERT.
b. Efisiensi input data menggunakan MySQL pada basis data berbasis web untuk *multiple-table query* dapat diperoleh dengan memakai perintah LOAD DATA INFILE untuk penyimpanan data sementara dan INSERT INTO TABLE … SELECT … untuk mendistribusikan data ke tabel sebenarnya. Keuntungan yang diperoleh sama dengan keuntungan pada efisiensi input data pada

satu tabel dengan tambahan program yang dibuat dapat berfungsi sebagai mesin normalisasi tabel.

c. Efisiensi output data menggunakan MySQL baik pada *single-table query* atau *multiple-table query* pada basis data berbasis web dapat diperoleh dengan memakai indeks. Indeks juga harus dibantu pada kejadian-kejadian di mana indeks tidak berfungsi sama sekali.

## 4. Daftar Pustaka


Atkinson, L., 1999, *Core PHP Programming*, Prentice Hall PTR.

Arbie, 2004 , Manajemen Database dengan MySQL, Andi Offset

Buther, Tony, 2000, Teach Youself MySQL in 21 Days, SAMS .

Bakken, S.S., Aulbach, A., Schmid, E., Winstead, J., Wilson, L.T., Lerdorf, R., Zmievski, A., Ahto, J., 2002, *PHP Manual*, PHP Documentation Group.

Date, C.J., 1986, *An Introduction To Database System*, Addison-Wesley Publishing Company..

Ramakhrisnan, R., 1998, *Database Management System*, Mc Graw-Hill International Edition.

Silberschatz, A., Korth, H.F., Sudarsan, F., 1997, *Database System Concepts*, McGraw-Hill in Series in Computer Science.

_____, 2007, *MySQL Reference Manual*, TcX AB, Detron HB, and MySQL Finland AB, **www.mysql.com** .

_____, 2007, *Dictionary Websters*, **www.websters.com** .

_____, 2007, PHP **www.php.net**