ANALSIS PERFORMANSI REFACTORING DATABASE

Husna Alfiani[1], Kma[2], Kal[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Refactoring Database merupakan proses memperbaiki desain dari suatu skema basis data dengan memperhatikan faktor-faktor tertentu yang disebut dengan database smell. Perubahan desain dari skema database akan mempengaruhi waktu pemrosesan query dan biaya pemrosesan query. Dengan menggunakan pada sebuah representasi lojik, yaitu entity relationship diagram (ERD), dibuat desain dari suatu skema basis data yang telah di-refactor. Dalam menyelesaikan pemrosesan query pada proses bisnis yang telah ditentukan, skema refactoring dapat menyelesaikan semua pemrosesan query walaupun bila dibandingkan dari sudut pandang waktu pemrosesan query dengan skema yang tidak mengalami refactoring, tidak berbanding jauh. Namun terlihat bahwa terdapat kasus-kasus dimana biaya untuk pemrosesan query dapat diproses dengan biaya yang murah bila menggunakan skema refactoring.

Kata Kunci : Refactoring, skema basis data, query

Abstract

Database Refactoring is a process to improve design of a database schema by observing some factors called database smells. Design changes of a database schema will affect processing time and cost for processing the query itself.
Using a logic representation, which is entity relationship diagram (ERD), a design is made from a database schema that has been refactored. In compeleting query processing on a choosen bussines process, refactored schema can done all the query processing although if compared to schema without refactoring from query processing point of view, the result is not far difference. But proven that there is some case where the cost for query processing can be reduced using refactored schema.

Keywords : Refactoring, database schema, query

1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan zaman saat ini berpengaruh terhadap hampir semua aspek kehidupan. Teknologi juga mendapat perhatian khusus dalam hal ini. Pengelolaan data sebagai penyokong teknologi yang berkembang pesat saat ini semakin banyak dan kompleks. Keberadaan data yang sedemikian rupa, dengan *volume* data yang besar juga jumlah data yang semakin besar, berkaitan dengan bagaimana pengaksesan data tersebut. Maka dari itu, pengaksesan data diharapkan memiliki performansi yang tinggi.

Desain dari suatu basis data menjadi salah satu faktor signifikan dalam pengaksesan data. Dengan me-*refactor* skema yang kemudian akan mengasilkan skema yang baru diharapkan dapat memperbaiki kualitas dan mempercepat pengaksesan data, setelah itu akan diuji performansinya, seberapa besarkah pengaruh skema yang telah di-*refactor* dengan yang belum di-*refactor*.

Refactoring bukanlah proses mencari *bug* maupun menambah fungsionalitas baru, namun lebih ditekankan untuk memperbaiki desain agar lebih mudah dimengerti, sehingga memudahkan *maintenance* pada masa yang akan datang. Di samping itu, jika kode memiliki desain yang bagus, maka penambahan atau perubahan *requirement* tidak menjadi masalah besar [2]. *Refactoring database* tidak jauh berbeda dengan *refactoring* kode program, tujuannya sama, membuat desain baru yang diharapkan mudah dipakai. Namun, apakah performansi *refactoring* database yang bertujuan untuk memperbaiki desain cukup tinggi dalam pemakaiannya?

Refactoring database adalah pengubahan terhadap skema database untuk memperbaikin desain dari skema database tersebut tanpa mengubah perilaku dari skema database tersebut [1]. Dari data yang diperoleh akan dibuat skema yang baru yang telah di-*refactor*, yang tentu saja memiliki performansi yang berbeda dengan skema yang belum di-*refactor*. Setelah di-*refactor*, maka akan diuji dengan melakukan *query* pada dua buah model tersebut dan akan dianalisis performansinya, dan terakhir akan ditarik kesimpulan model manakah yang cocok atau layak diterapkan.

1.2 Perumusan Masalah

Dalam me-*refactor* suatu skema database kita dapat me-*refactor* dua aspek, yaitu aspek struktural seperti tabel atau view dan aspek fungsional seperti prosedur atau trigger. Sebelum melakukan *refactoring*, hal-hal yang harus dimiliki skema *database* adalah DDL (*Data Definition Language*), prosedur, trigger, view. Yang perlu diperhatikan dalam me-*refactor database* adalah kita tidak bisa menambah fungsi atau menghapus fungsi yang sudah ada ataupun mengubah makna dari *database* itu sendiri.

Perumusan masalah dalam penelitian ini adalah sebagai berikut :

- Seberapa besarkah pengaruh *refactoring database* dan seberapa tinggi performansi *refactoring database*.
- Apakah *refactoring database* cocok atau layak diterapkan pada suatu kasus.

1.3 Tujuan

Tujuan dari dilakukannya penelitian ini adalah

1. Mengimplementasikan *refactoring database.*
2. Menganalisis performansi *refactoring* database dengan parameter waktu pemrosesan *query* dan biaya pemrosesan *query*.

1.4 Batasan Masalah

Batasan masalah untuk penelitian ini adalah :

1. Menggunakan skema tertentu dan mengubah skema tersebut dengan cara melakukan proses *refactor* serta membandingkan performansinya.
2. Jenis aspek yang akan di-*refactor* adalah aspek struktural.
3. Skema yang digunakan adalah skema SH (*Sales History*), yaitu skema bawaan dari Oracle 10g.
4. Kedua skema yang digunakan, baik untuk skema asli maupun skema *refactoring* menggunakan pemodelan *entity relationship diagram*.
5. Terbatas pada studi kasus untuk proses bisnis transaksi jual beli.
6. Pembangunan system menggunakan Java jdk 6.10 dan Oracle 10g R2 sebagai DBMS.

1.5 Metodologi Penyelesaian Masalah

Penelitian ini akan mengimplementasikan *refactoring database*. Dan sebagai pembanding dalam pengukuran performansi digunakan skema sebelum di-*refactor*. Parameter yang akan menjadi perbandingan pada kedua model adalah waktu pemrosesan *query* dan biaya pemrosesan *query*.

a. Studi Literatur :
   Pencarian referensi dan sumber-sumber lain yang dapat digunakan sebagai acuan dalam penelitian *refactoring* database.
b. Pengumpulan dan pengolahan data :
   Mencari studi kasus yang akan digunakan sebagai bahan *refactor*.
c. Analisis dan Desain :
   Analisis terhadap database yang belum di-*refactor* yang kemudian akan di-*refactor* sesuai dengan *database smell*.
d. Implementasi :
   Membangun skema yang baru, lalu dijalankan *query* tehadap skema yang belum di-*refactor* dan yang sudah di-*refactor*, kemudian di ukur performansi keduanya.
e. Analisis hasil :
   Analisa sejauh mana performansi *refactoring* database yang mencakup waktu pemrosesan *query* dan biaya pemrosesan *query*.
f. Pengambilan kesimpulan dan penyusunan laporan tugas akhir :

Pengambilan kesimpulan dari hasil analisis yang telah dilakukan pada tahap sebelumnya untuk kemudian disusun laporan terhadap analisis yang telah dilakukan

1.6 Sistematika Penulisan

Tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I **PENDAHULUAN**

Berisi pemaparan mengenai latar belakang permasalahan, tujuan yang ingin dicapai dengan adanya penelitian ini, perumusan masalah, batasan masalah, metodologi tugas akhir, dan sistematika penulisan.

BAB II **LANDASAN TEORI**

Berisi uraian mengenai landasan teori yang akan digunakan, meliputi teori tentang bagaimana skema yang harus di-*refactor* dan teori-teori lain yang berkaitan dengan penelitian tugas akhir ini

BAB III **ANALISIS DAN PERANCANGAN SISTEM**

Berisi tentang analisa dan perancangan terhadap skema database yang telah di-*refactor*

BAB IV **ANALISIS DAN PENGUJIAN SISTEM**

Berisi implementasi dari hasil analisa dan perancangan sistem yang dibuat, serta pengujian kehandalan sistem.

BAB V **PENUTUP**

Berisi kesimpulan dan saran-saran untuk pengembangan lebih lanjut terhadap hasil penelitian ini.

5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian, dapat ditarik kesimpulan sebagai berikut:

1. Skema baru dapat diterapkan dan memiliki performansi yang baik. Dari segi waktu pemrosesan *query* maupun biaya pemrosesan *query*, untuk kasus tertentu yaitu adanya *query* yang mengakses pada tabel partisi dengan kriteria yang tepat.
2. Dekomposisi tabel yang menjadi bagian dari proses *refactor* menyebabkan waktu pemrosesan *query* pada skema baru menjadi lebih lama karena *join* yang digunakan lebih banyak karena melibatkan banyak tabel akibat proses dekomposisi.
3. Skema baru memiliki struktur tabel yang baik, sehingga unggul dalam waktu pemrosesan *query*. Hal ini terbukti dari kasus-kasus dimana skema baru memiliki waktu pemrosesan *query* yang lebih baik dari skema lama.
4. Untuk biaya pemrosesan *query*, tabel yang dipartisi menyebabkan biaya yang dibutuhkan menjadi minimal. Dari penelitian ini, untuk kasus pemrosesan *customer income level* untuk produk tertentu biaya yang dibutukan pada skema lama adalah 5443 dan untuk skema baru adalah 1442. Keunggulan untuk skema baru sebesar 73,81%
5. Kriteria untuk tabel partisi juga berpengaruh pada waktu pemrosesan *query* dan juga biaya pemrosesan *query*.

5.2 Saran

Setelah penelitian ini selesai dilakukan dan telah dilakukan dianalisis, penulis memiliki bebrapa saran, antara lain :

1. Pada studi kasus yang diuji pada penelitian ini hanya ditemukan 2 *database smell*. Penelitian bisa dikembangkan dengan menggunakan studi kasus yang mencakup keseluruhan *database smell*.
2. Partisi yang dilakukan dapat dikembangkan dengan melakukan penyebaran pada *tablespace*.
3. *Refactoring* dilakukan dengan pemodelan *relational*, dapat juga dikembangkan dengan menggunakan pemodelan lain untuk proses *refactor*, seperti ORDBMS, OODB, *dimensional modeling*, dan lain sebagainya.

6 Daftar Pustaka

[1] Ambler S.W dan Sadalage P.J. 2006. *Refactoring Databases: Evolutionary Database Design*. Boston, Addison Wesley Professional

[2] Ambler S.W. 2008. *The Process of Database Refactoring : Strategies for Improving Database Quality*. Dikutip : 8 Agustus 2008, [online]. Available : http://www.agiledata.org/essays/database*Refactoring*.html

[3] Amikom. 2007. *Refactoring Database*. Dikutp, 8 Agustus 2008, [online]. Available : http://blog.amikom.info/sukrisno/archives/date/2007/07/23

[4] Cummunity Server. 2006. *Refactoring Database*. Dikutip : 8 Agustus 2008, [online]. Available : http://geeks.netindonesia.net/tags/Diary/.NET+Technology/default.aspx?PageIndex=1.

[5] Fatansyah. 2004. *Basis Data*. Informatika : Bandung.

[6] Forum Amikom.2007. *Refactoring Database*. Dikutip : 8 Agustus 2008, [online]. Available : http://forum.amikom.ac.id/main/viewpost_bebas.php?fid=9&tid=1295

[7] Fowler, Martin. 2003. *Evolutionary Database Design*. Dikutip 9 Agustus 2008, [online]. Available : http://www.martin fowler.com

[8] Marlinda L. 2004. *Sistem Basis Data. Jakarta, Penerbit Andi*

[9] Mulyanto. 2007. Partisi Tabel pada Oracle 10g. Dikutip : 30 Januari 2009, [online]. Available : http://mulyanto.net

[10] Rohmat. 2007. *Partitioning Table: Definisi dan Contoh*. Dikutip : 18 Februari 2009, [online]. Available http://rohmat.net

[11] Tim Asisten Kerja Lab OOT. 2007. *Refactoring. STT Telkom Bandung*

[12] Tim Asisten Kerja Lab Sistem Basis Data.2007. *Security*. STT Telkom Bandung

[13] Tim Asisten Praktikum Teknik Informatika II. 2007. *Modul Praktikum Teknik Informatika II dan Basis Data*. STT Telkom Bandung