

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 menghasilkan 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 memperbaiki 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* terhadap 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.