ORM SEBAGAI PERSISTANCE LAYER UNTUK MEMETAKAN OBJECT DARI APLIKASI KE RDBMS

Desi Nurhayati[1], Dhinta Darmantoro[2], Hari Saptoadi[3]

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

Abstrak

Penulisan SQL secara handcoded dan berulang-ulang merupakan masalah bagi programmer dalam memaintain sebuah aplikasi. Salah satu cara untuk mengatasi masalah diatas yaitu dengan menerapkan ORM. Secara umum ORM (Object Relational Mapping) adalah cara atau teknik untuk memetakan object dari aplikasi ke tabel basisdata relasional. Object dari aplikasi akan dipetakan oleh ORM dan diubah ke dalam sintaks SQL(Structure Query Language). Setelah itu SQL dijalankan pada basisdata relasional dan hasilnya dikembalikan menjadi object ke aplikasi oleh ORM. Metoda yang digunakan untuk membuat SQL generator dengan menerapkan builder pattern. Sedangkan untuk menangani perpindahan database diterapkan strategi pattern. Adapun parameter yang diujikan pada implementasi ORM ini adalah fungsionalitas dari aplikasi ketika ORM diimplementasikan, kesesuaian antara metada dari class dan metadata di database, perpindahan database dan perbandingan jumlah kode baris antara aplikasi yang menerapkan ORM dengan yang tidak menerapkan ORM.

Berdasarkan hasil analisis, maka penerapan ORM ini dapat mengurangi ketergantungan database dan kode baris pada bagian model aplikasi.Kemudian jika ada perubahan metadata maka perubahan itu hanya dilakukan pada layer presentation dan database sedangkan ORM sendiri sebagai persistance layer tidak perlu melakukan perubahan metadata.

Kata Kunci : Object Oriented, ORM, persistance layer, LOC, RDBMS, PHP

Abstract

Writing SQL code through handcoded and repeatedly is being a problem for programmer to maintain application. One of solution is by implementing ORM. Generally, ORM is a way to map an object from application to relational database table. The object from application is mapped by ORM and then is changed to SQL. After that SQL is run in relational database and the result is returned to object in application by ORM (Object Relational Mapping). Method which is used to make SQL generator is builder pattern. To handle database changing, strategy pattern is used. There are some paramater which is tested in this ORM implementation : application functionality when ORM is working, similiarity between class metadata and database metadata, database changing and LOC (Line Of Code) comparation between application with ORM and without ORM. Based on result of analysis, ORM implementation can reduce database coupling and line of code in application model. Beside that, if there are changes on metadata database, it will be done in presentation layaer and database. ORM as persistence layer do not change.

Keywords : Object Oriented, SQL, ORM, persistance layer, LOC, RDBMS, PHP

1. Pendahuluan

1.1 Latar Belakang

Penulisan kode yang berhubungan dengan *database* merupakan hal yang membosankan bagi sebagian besar para *programmer*. Hal ini disebabkan karena kode yang sama sering ditulis berulang-ulang terutama pada *query* untuk proses *CRUD (create, retrieve, update, delete)*. Kode *SQL* dan *business logic* di dalam aplikasi (*hand coded*) juga menimbulkan kesulitan ketika ada perubahan dari sisi struktur *database* maka banyak pula perubahan kode di sisi aplikasi. Penulisan kode yang berulang-ulang mengakibatkan baris kodepun menjadi panjang. Sebaliknya baris kode yang lebih sedikit membuat program lebih mudah dipahami.

ORM sebagai persistance layer adalah salah satu solusi untuk mengatasi permasalahan diatas. Teknik mapping yang digunakan adalah c*lass* dipetakan pada tabel kemudian melakukan generate *SQL* dengan cara melakukan pengubahan dari tabel ke dalam sintaks *SQL*. Strategi yang digunakan menggunakan *builder pattern.* Sedangkan untuk menangani *database* yang berbeda disini akan menerapkan *strategy pattern.*

Tools *ORM* di Java sudah banyak dipakai seperti *iBatis* dan *Hibernate*. Namun di *PHP*, *ORM* merupakan sesuatu yang masih baru dan belum terlalu banyak dipakai. Oleh karena itu topik dari Tugas Akhir ini akan membahas bagaimana membangun *ORM (Object Relational Mapping)* sebagai *persistance layer* yang memiliki fungsi memetakan *object* dari aplikasi ke *RDBMS* kemudian bagaimana mengimplementasikannya ke dalam aplikasi.

1.2 Perumusan Masalah

Dalam tugas akhir ini penulis akan menitikberatkan pada bagaimana *ORM* itu dibangun. Dalam hal ini bagaimana cara melakukan mapping class ke tabel, mengatasi query dan bagaimana cara mengatasi perpindahan *database*. Setelah *ORM* dibangun bagaimana cara mengimplementasikannya di aplikasi kemudian membandingkan dengan aplikasi yag tidak mengimplementasikan *ORM*.

Ruang lingkup dalam tugas akhir ini adalah:

1. Studi kasus menggunakan submodul fixed asset dan hanya sebatas proses *CRUD* standar.
2. Tugas akhir ini tidak membahas mapping pada semua hubungan relation class. Fokusnya hanya pada mapping aggregation class.
3. Bentuk query yang diimplementasikan sederhana.
4. Desain ORM dan implementasi ORM dimodelkan menggunakan *UM*L.
5. Bahasa pemrograman yang digunakan adalah PHP 5.0.
6. Database yang digunakan adalah MySQL dan Oracle10g.

1.3 Tujuan

Tujuan penelitian dari tugas akhir ini adalah :

1. Membuat *API(Application Programming Interface)* untuk menangani operasi-operasi *CRUD*.

2. Membuat *API* untuk melakukan *query* yang mengacu pada *class* dan propertinya.
3. Menghilangkan ketergantungan antara aplikasi dan *database.*
4. Menerapkan *ORM* dengan cara mengubah struktur kode pada bagian model dan mengubah bentuk penulisan kode *SQL* pada bagian controller aplikasi.
5. Menganalisis hasil uji coba aplikasi yang menerapkan dan tidak menerapkan *ORM* dengan parameter ukuran jumlah baris kode program.

1.4 Metodologi Penyelesaian Masalah

Untuk mencapai tujuan yang dimaksud, maka metodologi yang digunakan dalam penelitian penyusunan tugas akhir ini adalah :

1. Studi Literatur, langkah ini bertujuan untuk mempelajari dan memahami teori-teori dasar tentang konsep *object, UML, SQL, ORM, PHP* dan *OOP*.
2. Menguraikan analisis terhadap *ORM* yang akan dibangun beserta input outputnya.
3. Mendesain *ORM* dan implementasi ORM menggunakan notasi *UML.*
4. Menerapkan ORM pada submodul fixed asset menggunakan bahasa pemrograman PHP sesuai dengan rancangan submodul fixed asset.
5. Melakukan pengujian dari aplikasi yang telah dibangun berdasarkan parameter ukuran jumlah baris kode
6. Pengambilan kesimpulan dan menyusun laporan.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Beberapa kesimpulan yang dapat diambil dari analisa sistem Sistem yang dibangun adalah:

1. Aplikasi tetap bisa berjalan dengan benar dan tidak mengalami error ketika *ORM* diimplementasikan.
2. *ORM* bisa diterapkan pada *database* yang berbeda tanpa harus mengubah script. Dalam hal ini support *MySQL* dan *Oracle*. Proses perpindahan *database* yang dilakukan adalah hanya dengan mengubah koneksi saja.
3. Penambahan metadata kolom tidak akan mempengaruhi kerja *ORM*. Perubahan tipedata tidak akan mempengaruhi kerja *ORM* selama tidak dilakukan operasi pada data. Yang mempengaruhi kerja *ORM* tidak bisa bekerja adalah jika nama tabel atau nama *class* diubah karena salah satu kondisi yang harus dipenuhi agar *ORM* bisa bekerja adalah memadankan nama class dan tabel terlebih dahulu. Kemudian perubahan format data *date* dan *binary* pada *ORM* tidak disupport.
4. Dari fakta dapat disimpulkan bahwa implementasi *ORM* bisa menjadikan baris kode lebih berkurang terutama berpengaruh pada bagian model dari aplikasi sehingga tidak perlu mengubah pada bagian view.
5. Pada bagian controller aplikasi kita cukup mengubah kode program yang berhubungan dengan query dan mengganti sintaksnya dengan mengacu pada method-method *ORM*.
6. *ORM* didesain untuk mengurangi ketergantungan aplikasi terhadap *database* akan tetapi perubahan metadata tidak bisa dilakukan sembarangan tetap harus disesuaikan dengan rancangan aplikasi.

5.2 Saran

Beberapa saran yang dapat dikembangakan dari tugas akhir ini diantaranya:

1. Menerapkan *ORM* untuk studi kasus *SQL* yang lebih komplek
2. Mengembangkan *ORM* untuk studi kasus *object oriented* yang mempunyai asosiasi lebih komplek.
3. *ORM* bisa support banyak *database.*

DAFTAR PUSTAKA

[1] Ambler, Scott. 2003. W. *The Elements of UML Style*. Cambridge University Press

[2] Azis, Farid M,Ir,M.Kom.2005.*Object Oriented Programming dengan PHP5*. PT.Elex Media Komputindo

[3] Bagui, Shika; Richard Earp. 2003. *Database Design Using Entity-Relationship Diagram*

[4] Bauer, Christian and King, Gavin.2005. *Hibernate in Action .Manning*. Publication.Co

[5] Craig , Larman.1998. *Applying UML and Pattern*

[6] Fowler, Martin; David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford. November 2002. *Patterns of Enterprise Application Architecture*, Addison Wesley

[7] Fussel, Mark L. 1997. *Foundations of Object Relational Mapping*. ChiMu Corporation. www.chimu.com/publications/objectRelational/

[8] Gamma, Erich; Richard Helm, Ralph Johnson, John Vlissides. 1996. *Design Pattern Elemen Of Reusable Object Oriented Software*

[9] Kriegel, Alex; Boris M. Trukhnov. 2000. *SQL Bible*, John Wiley & Sons.

[10] Nugroho, Widodo. *Tip dan Trik Pemograman Delphi.* PT.Elex Media Komputindo, 2002

[11] Sperko, Richard. 2003. Java *Persistence for Relational Databases*. Apress

[12] Sweat, Jason E. 2005. *PHP Architect Guide to PHP Design Patterns.* Apress

[13] Wolfgang, Keller. 2004. *Mapping Object To Tables*