INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS
Vol. 3, No. 2, Juni 2019, 173 - 178
E-ISSN: 2548-3331

Migrasi dari Sistem Relasional ke Basis Data Objek

Rahmadya Trias Handayanto [1,*], Nove Anggara Syah Sejati [1], Muhammad Aqil Emeraldi [1], Muhammad Ilham [1], Randika Purwadhana [1], Angga Fahreja [1], Aji Trisnantoro[1], Muhammad Ramadhan Fikri[1], Sella Alaida Syifa[1], Muhammad Irvan [1]

[1] Teknik Komputer; Universitas Islam 45 Bekasi; Jl. Cut Meutia No. 83 Bekasi Timur 17113 Indonesia Telp. (021 Telp: (021) 8808853 Fax: (021) 8808853; e-mail: rahmadya.trias@gmail.com, noveanggara1803@gmail.com, syahruli917@gmail.com, mi066285@gmail.com, prandikapurwadhana9@gmail.com, anggafahreza6@gmail.com, ajitrisnantoro10@gmail.com, mramadhanfikri@gmail.com, shellaalchernova1716@gmail.com, mirvan3107@gmail.com

* Korespondensi: e-mail: rahmadya.trias@gmail.com

Diterima: 24 Mei 2019; Review: 27 Mei 2019; Disetujui: 16 Juni 2019

Cara sitasi: Handayanto RT, Sejati NAS, Emeraldi MA, Ilham M, Purwadhana R, Fahreja A, Trisnantoro A, Fikri MR, Syifa SA, Irvan M. 2019. Migrasi dari Sistem Relasional ke Basis Data Objek. Information Management For Educators And Professionals. 3 (2): 173-178.

Abstrak: Sistem manajemen basis data relasional (RDBMS) merupakan sistem yang paling banyak digunakan di duni. Sistem ini menawarkan konsistensi, akurasi, keamanan dan aspek-aspek lain yang dibutuhkan dalam transaksi. Sistem manajemen basis data objek (ODBMS) di sisi lain menawarkan kesederhanaan dan struktur yang berbasiskan objek. Objek yang tersusun dalam kelas-kelas memiliki relasi yang berinteraksi dengan basis objek dengan kelas-kelas lain tanpa melihat hubungan atributnya (field dalam sistem relasional). Penelitian ini mencoba menyelesaikan masalah-masalah yang kerap terjadi dalam perubahan sistem relasional menjadi objek. Hasil uji coba mengindikasikan bahwa aspek-aspek tertentu harus diperhatikan ketika migrasi antara lain: perbedaan konsep penggunaan kunci utama, kunci tamu, dan metode penyimpanan.

Kata Kunci: *Java, Netbeans, DB4O, MySQL*

Abstract: *Relational Database Management System (RDBMS) is the most implemented system in the world. This system provides consistency, accuracy, security, and other aspects related to transaction. In the other hand, Object Database Management System (ODBMS) offers simplicity by object implementation. An object in a class has many relations to other objects without seeing their attributes (field in RDBMS). This study tried to solve problems that usually appears in converting relational into object system. Testing results indicated some aspects should be considered, i.e. the different in primary key, foreign key usages, and data store system method.*

Keywords: *Java, Netbeans, DB4O, MySQL*

1. Pendahuluan

Saat ini pemrograman berorientasi objek mulai banyak menarik minat programer-programer seiring dengan aplikasi permainan yang banyak beredar. Sistem ini mudah dimengerti karena mengadopsi cara berfikir objek yang sesuai dengan kondisi real sehari-hari [Arlow and Neustadt, 2004]. Untuk sistem transaksi atau sistem informasi kebanyakan masih menerapkan sistem relasional yang telah lama dikembangkan oleh vendor-vendor basis data yang dipelopori oleh International Business Machine (IBM) dilanjutkan oleh provider basis data seperti SQL Server, Oracle, MySQL, dan lain-lain [Hoffer et al., 2011; Valacich et al., 2009].

Perkembangan teknologi yang pesat memacu tersedianya data-data berukuran besar dari aplikasi-aplikasi sosial media dengan karakter yang setengah terstruktur (semistructured) dengan format data yang banyak diterapkan pada sistem relasional [Yu, 2011]. Akhirnya beberapa pengembang basis data mulai membentuk sistem basis data objek, salah satunya

E-ISSN: 2548-3331; 173 – 178

adalah DB4O [Paterson et al., 2010]. Sistem basis data open source ini menyediakan fasilitas penyimpanan objek tanpa memecah menjadi tabel-tabel. Karena objek memiliki atribut sekaligus operasi, DB4O dapat menyimpan sekaligus atribut dan operasinya. Salah satu keunggulan basis data objek adalah kemampuan menyimpan atribut jamak (multivalue) yang tidak dapat diakomodir oleh sistem relasional (biasanya dengan menambah satu tabel baru).

Sistem berorientasi objek saat ini telah memiliki standar yang dikenal dengan nama Unified Modeling Language (UML) yang sudah masuk versi 2 dengan diagram-diagram yang diusulkan antara lain: use case, object, class, activity, package, deployment, communication, interaction diagram, timing diagram, etc [Jacobson et al., 1999; Sommerville, 2007; Pressman, 2001; Hoffer et al., 2011]. Tidak semua diagram UML tersebut harus dilibatkan dalan analisa dan disain. Dalam prakteknya model-model yang telah lama digunakan dalam pemrograman terstruktur seperti Entity Relationship Diagram (ERD) tetap boleh diterapkan dalam perancangan sistem berorientasi objek, terutama ketika menerapkan sistem objek relasional (ORDBMS).

Penelitian ini bermaksud menjawab permasalahan-permasalahan yang kerap muncul ketika mengkonversi sistem relasional menjadi objek, dikenal dengan istilah impedance mismatch [Paterson et al., 2010]. Fokus yang akan diteliti adalah ketika membuat transaksi yang melibatkan lebih dari satu kelas objek dimana peran kunci utama dan kunci tamu merupakan bagian sentral dari pemrograman terstruktur yang harus diselesaikan dengan baik saat migrasi ke sistem berorientasi objek.

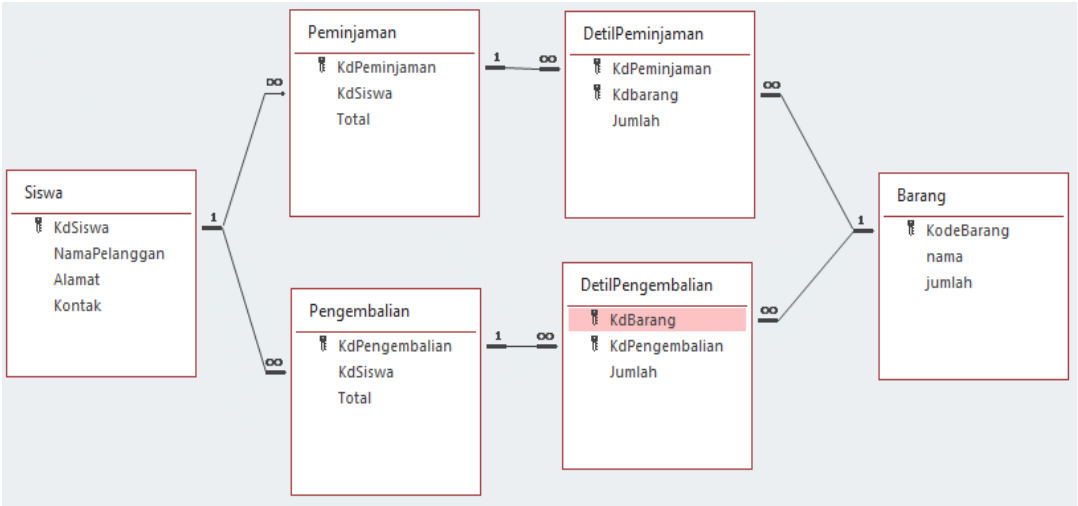
Artikel dimulai dengan penjelasan singkat sistem yang akan dimigrasi pada bab metode. Pada bab ini cara-cara konversi dijelaskan setelah tinjauan sistem. Bagian hasil dan pembahasan menjelaskan masalah-masalah yang berhasil diselesaikan disertai kemungkinan-kemungkinan kegagalan ketika melakukan migrasi. Artikel diakhiri dengan kesimpulan yang menjawab permasalahan penelitan.

2. Metode Penelitian

2.1. Sistem Relasional

Sistem yang akan dijadikan uji coba migrasi adalah sistem informasi peminjaman alat laboratorium sebuah sekolah. Di sini peminjam yang merupakan siswa di sekolah tersebut dicatat barang yang dipinjam dengan jumlah tertentu. Setiap peminjaman akan mengurangi stok barang yang ada. Tentu saja jika stok barang habis, siswa tidak bisa meminjam alat tersebut.

Operator merupakan staf yang harus terdaftar namanya dalam tabel operator. Operator terlebih dahulu login sebelum melakukant transaksi baik peminjaman, pengembalian, maupun pelaporan. Gambar 1 memperlihatkan struktur basis data sistem peminjaman.



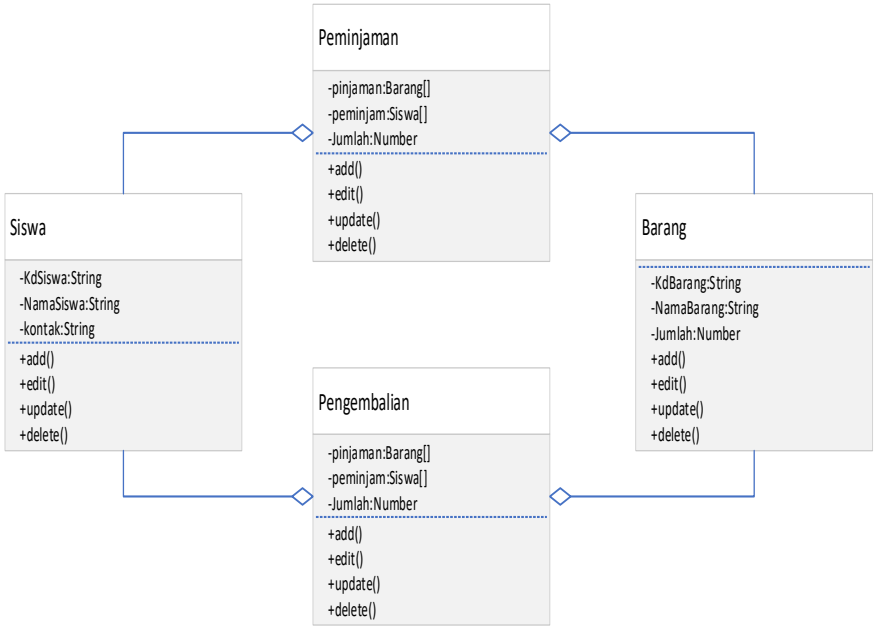
Sumber: Hasil Penelitian (2019)

Gambar 1. Relasi Antar Tabel

2.2. Sistem Berorientasi Objek

Untuk mengkonversi sistem relasional menjadi berorientasi objek perlu terlebih dahulu mendefinisikan objek-objek yang terlibat dalam sistem. Objek-objek tersebut antara lain siswa,

barang, peminjaman, pengembalian dan laporan. Hubungan antar objek tersebut disederhanakan dalam sebuah kelas diagram (Gambar 2).

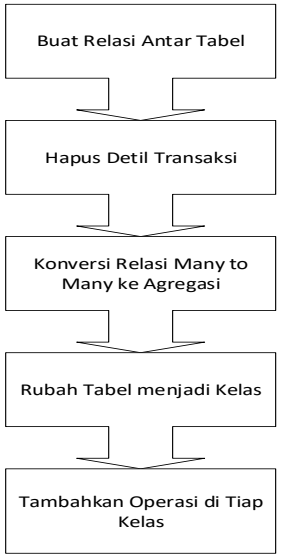


Sumber: Hasil Penelitian (2019)

Gambar 2. Diagram Kelas

Berbeda dengan tabel yang hanya memiliki field, objek memiliki atribut (setara dengan field) sekaligus operasi. Dengan operasi yang hanya dapat dilakukan oleh kelas yang bersangkutan, prinsip enkapsulasi (prinsip yang menjaga intervensi kelas lain terhadap suatu kelas) dapat diterapkan. Kelas yang memiliki operasi juga memudahkan pengembang dalam melakukan penggunaan kembali (reuse) jika ingin membuat aplikasi-aplikasi sejenis.

Untuk menyimpan objek pada penelitian ini sebuah basis data objek, DB4O, digunakan. Database ini tidak perlu diinstal melainkan hanya didaftarkan pada library Java Netbeans. Lokasi file basis data perlu dijaga aksesnya karena file tersebut sangat penting. Sebaiknya backup rutin perlu dijalankan terhadap file basis data tersebut (secara default DB4O menggunakan ekstensi *.yap pada file basis datanya walaupun dapat menggunakan nama ekstensi lainnya).



Sumber: Hasil Penelitian (2019)

Gambar 3. Proses Migrasi

E-ISSN: 2548-3331; 173 – 178

Gambar 3 memperlihatkan proses migrasi dari sistem relasional ke sistem berorientasi objek. Relasi antar tabel dan ERD menjadi patokan dasar dilanjutkan dengan mencari kandidat-kandidat kelas objek. Jika kelas sudah dipastikan sesuai dengan sistem yang ada maka dilanjutkan dengan membuat diagram kelas (class diagram) sebelum membuat kelas-kelasnya dalam Integrated Development Environment (IDE) Netbeans. Untuk menggantikan DBMS MySql diperlukan JAR dari DB4O untuk media penyimpanan kelas-kelas yang terlibat. Pada dasarnya tabel dapat dikonversi menjadi kelas real (bukan kelas abstrak). Namun beberapa tabel seperti tabel detil dapat dimerger dengan tabel transaksi dengan mengijinkan adanya multivalue. Kelas juga bisa ditambahkan dengan kelas abstrak yang berupa interface untuk input-output program rancangan dan yang terpenting untuk kelas real harus ditambahkan operasi yang sesuai, terutama operasi dasar basis data (CRUD). Operasi lainnya misalnya konstruktor boleh tidak dituliskan dalam diagram kelas yang dihasilkan.

3. Hasil dan Pembahasan

Dalam proses migrasi banyak kendala-kendala yang dihadapi. Kendala-kendala tersebut akan coba diselesaikan pada penelitian ini.

3.1. Impedance Mismatch

Kendala pertama adalah masalah kompatibilitas yang dikenal dengan istilah “impedance mismatch”. Masalah impedansi ini terkait teknik yang berbeda antara sistem relasional dengan yang berorientasi objek. Selain itu masalah kultur dan sudut pandang pemrogram juga menjadi bahan pertimbangan jika ingin bermigrasi dari sistem relasional ke objek dan juga sebaliknya. Perlu diketahui, sistem berorientasi objek memandang objek tidak hanya atribut melainkan juga proses yang ada di dalamnya. Misalnya pada objek barang tidak hanya kode barang, nama, jumlah dan atribut-atribut lainnya saja yang tersimpan dalam sistem basis data melainkan juga operasi-operasi yang terlibat di dalamnya seperti penambahan, pembacaan, perubahan, dan penghapusan yang dikenal dengan istilah create, read, update, dan delete (CRUD). Jadi tiap kelas objek terdapat operasi-operasi dasar (CRUD) tersebut yang memastikan sebuah kelas terenkapsulasi. Dengan enkapsulasi itu maka untuk memanipulasi sebuah objek harus lewat proses yang diijinkan oleh kelas tersebut. Akibatnya keamanan terjaga. Selain itu enkapsulasi dapat dimanfaatkan untuk digunakan kembali dalam sistem lainnya yang dikenal dengan istilah reuse. Tentu saja dengan sedikit modifikasi terhadap lokasi dan nama databasenya. DB4O yang digunakan dalam penelitian ini memiliki kemampuan untuk menyimpan secara permanen sebuah objek, yang dikenal dengan istilah persistence.

3.2. Multivalue

Dalam sebuah transaksi terkadang melibatkan jumlah item yang lebih dari satu. Untuk kasus ini misalnya seorang siswa meminjam lebih dari satu alat. Sistem relasional tidak mengijinkan satu field memiliki lebih dari satu nilai (dikenal dengan istilah multivalue). Sehingga perlu dibuatkan satu atau lebih tabel baru yang biasanya diberi nama detil transaksi seperti detil penjualan, detil pembelian, detil peminjaman dan lain-lain. Sistem berorientasi objek ternyata dapat menyederhanakan sebuah transaksi tanpa melibatkan satu tabel detil transaksi. Kode berikut contoh bagaimana multivalue diimplementasikan.

```
private Peminjaman(){
  this.pinjaman=new Barang[5];
}
```

Perhatikan di sini objek peminjaman dalam satu pinjaman boleh diisi lebih dari satu, misalnya 5 barang. Hal ini dapat digunakan juga untuk membatasi barang yang dipinjam, dalam hal ini maksimal 5 orang. Format yang digunakan adalah Array satu dimensi pada kelas Siswa. Jadi dalam satu kelas tertentu (dengan indikasi k.

3.3. Kunci Utama dan Kunci Tamu

Ketika satu transaksi melibatkan tabel lain, misalnya transaksi peminjaman melibatkan barang yang dipinjam, maka tabel tersebut membutuhkan informasi mengenai barang lewat kunci utamanya (foreign key). Kunci utama yang terletak pada tabel lain (misal tabel transaksi) dikenal dengan istilah kunci tamu (foreign key). Berbeda dengan sistem berorientasi objek yang

hanya memiliki kunci berupa identitas tertentu yang diberikan secara otomatis oleh sistem, mekanisme kunci tamu diganti dengan mekanisme relasi yang terdiri dari asosiasi, agregasi, komposisi, dan generalisasi. Untuk transaksi peminjaman, diperlukan data barang yang dipinjam dan diambil dari kelas Barang. Kelas Peminjaman membutuhkan satu atribut utama yaitu barang yang dipinjam, selain dari kode peminjaman. Kode berikut memperlihatkan cara menjadikan objek Barang menjadi atribut pada objek peminjaman.

```
public class Peminjaman {
public String kode,tanggal;
private Siswa[] peminjam;
private Barang[] pinjaman;
private int i;
```

Perhatikan pada kode di atas tidak sebuah atribut berasal dari kelas itu sendiri. Atribut peminjam dan pnjamannya berasal dari kelas lain berturut-turut Siswa dan Barang. Perlu diketahui penghapusan sebuah objek pada kelas Peminjaman tidak merubah isi kelas Barang sebaliknya, penghapusan Barang perlu dijaga karena masih dimiliki oleh kelas Peminjaman. Jadi hubungan yang berlaku adalah agregasi yang berbeda dengan asosiasi (tidak saling memiliki) sitasi. Ada satu permasalahan lain yang perlu diperhatikan dalam migrasi sistem relasional ke objek yaitu masalah kunci utama pada sistem berorientasi objek. Kunci utama pada sistem berorientasi objek sangat sulit diakses karena diberikan secara sistemik oleh DBMS-nya. Basisnya adalah “searching”, misalnya yang dipinjam adalah sebuah tang, maka sistem harus mencari terlebih dahulu barang tersebut, tidak bisa hanya dengan mengetik “tang”. Bahkan walaupun memiliki kode barang “011” untuk tang tidak serta merta ketika memasukan barang yang dipinjam dengan kode “011” akan terekam tang karena bisa saja barang baru dengan kode yang sama tetapi indeks/identitas yang berbeda. Oleh karena itu untuk memasukan barang yang dipinjam adalah tang maka perlu dilakukan proses searching terhadap sebuah kode barang yang unik (pada sistem relasional dalam bentuk kunci utama) dan hasil pencarian itulah yang dijadikan patokan barang yang dipinjam. Silahkan lakukan modifikasi lain, misalnya ketika sebuah tang dipinjam maka dilakukan proses pengurangan jumlah tang di kelas Barang, begitu pula jika proses pengembalian maka operasi penambahan dipanggil dari kelas Barang untuk menambah barang yang ada.

```
    public void add(Barang b){
     if(i<pinjaman.length){
        pinjaman[i]=s;
        System.out.println("Barang Pinjaman "+i);
        i++;
     }
        System.out.println("Barang Pinjaman: "+pinjaman);
  }
```

3.4. Relasi Many-to-Many

Relasi many to many biasanya muncul di transaksi yang melibatkan barang dan peminjam. Jika pada sistem relasional dibutuhkan satu tabel detil yang berfungsi mendata beberapa barang yang dipinjam pada sistem berorientasi objek tidak diperlukan. Hal ini terjadi karena sistem berorientasi objek mampu menyimpan data yang *multivalue*.

Untuk relasi antar kelas, pada sistem berorientasi objek terdapat empat jenis relasi antara lain: generalisasi, asosiasi, agregasi, dan komposisi. Mengingat ada atribut dari kelas peminjaman dan pengembalian yang berasal dari objek kelas lain yaitu pinjaman dan peminjam yang berturut-turut berasal dari kelas Barang dan Siswa. Maka relasi yang tepat adalah agregasi. Berbeda dengan komposisi yang objek anak akan hilang jika objek induk dihapus, pada agregasi hanya memiliki saja tetapi objek anak tidak hilang jika objek induk dihapus. Relasi lain yang digunakan dalam penelitian ini adalah asosiasi yang menghubungkan antara interface (kelas abstrak) dengan kelas realnya (Siswa, peminjaman, pengembalian, dan Barang). Asosiasi hanya memberikan pesan kepada kelas lain untuk menjalankan suatu operasi tertentu seperti misalnya pada frame (form) peminjaman yang memiliki tombol “Pinjam” ketika ada barang yang dipinjam. Tombol tersebut akan mengirim pesan ke kelas peminjaman untuk menjalankan operasi insert/create transaksi peminjaman yang baru. Berbeda denga sistem relasional yang fungsi manipulasinya di luar tabel dengan instruksi structured query language (SQL), *insert, update, delete,* pada sistem berorientasi objek operasi terletak pada kelas yang bersangkutan. Kelas lain hanya memberi pesan (*message*) ketika akan mengaktivasi operasi/metode tertentu. Hubungan pemberian pesan ini termasuk dalam kategori asosiasi. Manfaat dengan diletakannya operasi manipulasi objek di kelas yang bersangkutan

E-ISSN: 2548-3331; 173 – 178

adalah terjaganya prinsip enkapsulasi (*encapsulation*) dimana suatu kelas tidak dapat diutak-atik dari luar dan hanya dirubah lewat operasi yang diijinkan/ada pada kelas yang bersangkutan. Interface yang digunakan masih tetap yang lama hanya saja akses basis data sudah beralih ke basis data objek. Beberapa aspek masih tetap dipertahankan mengingat ketangguhan sistem relasional dalam hal security misalnya untuk akses login petugas bagian peminjaman (Gambar 4). Pengujian yang dilakukan pada penelitian ini dengan mengikuti alur program (*white box*) dan menguji terhadap fungsi-fungsi yang tersedia pada aplikasi yang dibuat (*black box*) dan hasilnya menunjukan hasil yang sama dengan sistem relasional sebelum migrasi.

Sumber: Hasil Penelitian (2019)

Gambar 4. Tampilan Menu Utama

4. Kesimpulan

Hasil uji coba menunjukan migrasi dari sistem relasional ke sistem berorientasi objek memiliki banyak kendala mendasar yang harus diselesaikan oleh pemrogram. Pada prinsipnya untuk sistem yang tidak memerlukan keterikatan antar field masalah yang muncul juga tidak terlalu kompleks. Sebenarnya database relasional dapat juga digunakan untuk sistem berorientasi objek yang dikenal dengan istilah sistem object relational (ORDBMS). Namun tidak efektif karena perlu merakit kembali tiap objek yang akan dioperasikan oleh sistem, berbeda dengan sistem berorientasi murni objek yang ketika menjalankan objek hanya memanggil saja objek yang tersimpan.

Referensi

Arlow J, Neustadt I. 2004. UML and the unified process. 81–82 p.

Hoffer JA, F. GJ, Valacich JS. 2011. Modern Systems Analysis and Design - Sixth Edition. New Jersey: Pearson Education Limited.

Jacobson I, Booch G, Rumbaugh J. 1999. The Unified Software Development Process. Addison-Wesley Publishing.

Paterson J, Edlich S, Horning H, Horning R. 2010. The Definitive Guide to db4o. United States: Appress.

Pressman RS. 2001. Software Engineering - A Practitioner's Approach, Fifth Edit. New York: McGraw Hill.

Sommerville I. 2007. Software Engineering - Eighth Edition. London: Pearson Education Limited.

Valacich J s., George JF, Hoffer JA. 2009. Essentials of Sistems Analysis and Design. New Jersey: Pearson.

Yu L. 2011. A Developer's Guide to the Semantic Web. New York: Springer.