JOINT

Journal of Informatics and Technology, Vol 2, No 3, Tahun 2013, p 113-124
http://ejournal-s1.undip.ac.id/index.php/joint

IMPLEMENTASI *OBJECT RELATIONAL MAPPING* PADA PENGEMBANGAN *E-COMMERCE* MENGGUNAKAN *FRAMEWORK* YII

Usva Dhiar Praditya [1], Ragil Saputra [2], Beta Noranita [3]

Jurusan Ilmu Komputer / Informatika Fakultas Sains dan Matematika Universitas Diponegoro

Email: dhiar.praditya@yahoo.co.id, ragil.saputra@undip.ac.id, betta@undip.ac.id

Abstrak

Objek Relational Mapping (ORM) merupakan sebuah ide atau teknik yang digunakan untuk memetakan basis data menjadi kelas yang digunakan dalam perangkat lunak. Terdapat sebuah toko komputer yang berupaya meningkatkan pemasaran komputer, tetapi memiliki kendala pada pembuatan perangkat lunak *e-commerce* sebagai alat untuk memasarkan produk dan penyimpanan data transaksi. Solusi yang cocok bagi permasalahan tersebut adalah digunakan teknik ORM untuk mengembangkan perangkat lunak. Alasan utama yang mendasari penggunaan ORM karena kemampuan ORM untuk menjaga integritas, validasi data, konsistensi dalam pengembangan berorientasi objek dan mendukung penggunaan komponen yang bisa digunakan ulang (*reusable*). ORM pada *framework* Yii memiliki kelebihan khusus untuk digunakan pada pengembangan perangkat lunak berbasis *web*. Proses pengembangan perangkat lunak menggunakan model pengembangan *unified process* (UP). UP mendukung penggunaan komponen yang juga terdapat pada ORM *framework* Yii. ORM yang digunakan diimplementasikan dengan pola arsitektur *active record* (AR). AR merupakan sebuah objek yang membungkus sebuah baris (*row*) dari sebuah tabel basis data atau *view*, merangkum (*encapsulate*) akses basis data, dan menambahkan logika domain pada data tersebut. Hasil akhir dari pengembangan perangkat lunak adalah dihasilkan sebuah perangkat lunak *e-commerce* yang memiliki integritas data, mampu menjaga validasi data, konsisten dalam pengembangan perangkat lunak berorientasi objek dan dapat digunakan ulang.

Kata kunci: *E-commerce*, Toko Ejcom, *Unified Process*, *Object Relational Mapping*, *Active Record*.

Abstract

Object Relational Mapping (ORM) is an idea or technique used to map the data base to used in software class. There is a computer shop work to increase marketing of computer, but have problems on the creation of e-commerce software as a tool to market the product and transaction data storage. Suitable solution to the problem is to use techniques ORM to develop software. The main reason underlying the use of ORM due to its ability to maintain its integrity, validation of data, consistency in the development of object-oriented and supports the use of reusable components. Software development process use the development model unified process (UP). UP supports the use of the components contained in the ORM on Yii framework. The used of ORM implemented by architecture pattern of active record (AR). AR is an object that wraps a row of a database table or view, encapsulates database access, and adds domain logic on that data. The final outcome of software development is an e-commerce software with data integrity, able to keep the data validation, consistent in the object-oriented software development and reusable.

Keywords: E-commerce, Ejcom Shop, Unified Process, Object Relational Mapping, Active Record.

1. Pendahuluan

Object Relational Mapping (ORM) merupakan teknik yang memetakan *persistent object* pada aplikasi menjadi tabel pada basis data serta memungkinkan untuk mengelola perbedaan tipe sistem pada basis data relasional dengan bahasa pemrograman berorientasi objek [6]. Implementasi ORM terletak pada lapisan *persistent* atau model yaitu lapisan yang paling dekat dengan sumber data. ORM dapat memiliki pola tertentu. Pola yang digunakan pada *framework* Yii adalah *Active Record* (AR).

AR adalah sebuah pendekatan untuk mengakses data dalam basis data. Di dalam *framework* Yii, AR merupakan kelas yang memiliki pola membungkus sebuah tabel basis data. *Framework* Yii mendukung pengembangan berorientasi objek dan juga mendukung penggunaan komponen sehingga cocok digunakan untuk model proses pengembangan perangkat lunak *Unified Process* (UP).

Model proses *Unified Process* (UP) merupakan proses pengembangan perangkat lunak yang secara konsisten mencoba beradaptasi dengan kompleksnya perangkat lunak [2]. Konsep *object oriented* (OO) yang diterapkan oleh UP memberikan kemudahan penggunaan kembali (*reusable*) [4]. Model UP juga mendukung penggunaan basis data relasional.

Basis data relasional merupakan sekumpulan data yang saling berhubungan, disimpan dengan minimum redundansi untuk melayani banyak aplikasi secara optimal [8]. Basis data relasional mempunyai karakteristik terintegrasi (*integrated*) dan pemakaian bersama (*shared*). Data di dalam basis data dianggap sebagai data yang menetap (*persistent data*). Basis data relasional sering digunakan pada pengembangan perangkat lunak yang menggunakan konsep *object oriented* (OO).

Pengembangan perangkat lunak menggunakan paradigma berorientasi objek memfokuskan penciptaan kelas yang merupakan cetak biru (*blueprint*) dari suatu objek. Objek menciptakan entitas yang memiliki batasan (*state*), sifat (*behavior*), dan identitas. Struktur dan *behavior* dari objek yang serupa didefinisikan di kelas mereka. Objek mempunyai siklus hidup: diciptakan, dimanipulasi dan dihancurkan.

Pengembangan perangkat lunak menggunakan basis data relasional dan pendekatan OO sering memunculkan ketidaksesuaian paradigma (*mismatch paradigm*). Ketidaksesuaian tersebut meliputi aspek kedetailan data (*granularity*), subtipe, identitas, asosiasi, dan navigasi data [6]. Persoalan ketidaksesuaian dapat diselesaikan dengen menerapkan teknik *Object Relational Mapping* (ORM).

Keberadaan teknologi informasi (TI) berupa perangkat lunak dapat dimanfaatkan sebagai media untuk menangani permasalahan yang terdapat dalam berbagai aspek kehidupan. TI mempunyai kekuatan dalam mengumpulkan, dan mengolah data menjadi informasi. Penulisan tugas akhir ini mengangkat salah satu topik teknologi informasi yaitu *Object Relational Mapping* (ORM) dam menerapkannya pada pengembangan perangkat lunak*e-commerce*.

2. Dasar Teori

2.1. Pengertian *E-commerce*

Electronic commerce (*E-commerce*) merupakan konsep baru yang digambarkan sebagai proses jual beli barang atau jasa menggunakan media internet maupun media elektronis [7]. Perdagangan utama dilakukan melalui berbagai media yang memadukan fungsi komputasi-telekomunikasi dan perpindahan data secara elektronik [1].

2.2. *Object Relational Mapping* (ORM)

Object Relational Mapping (ORM), merupakan sebuah teknik pemrograman yang menghubungkan SQL dengan konsep pemrograman berorientasi objek. ORM mampu menjembatani perbedaan tipe data pada konsep pemrograman yang berorientasi objek dengan konsep RDBMS [5].

Dengan mengimplementasikan ORM, pengembang bisa lebih berpikir secara objek dibanding dengan berpikir terhadap tabel dan kolom yang menjadi ciri dari model relasional. ORM muncul untuk menjaga kemurnian pola pikir pengembang atas pemrograman berorientasi objek. Implementasi ORM terletak pada layer *persistent* yaitu layer yang berhubungan dengan basis data[5].

2.3. *Object Relational Missmatch*

Object-relational missmatch adalah ketidaksesuaian yang muncul akibat penggunaan basis data yang menggunakan konsep relasional yang digunakan pada pengembangan aplikasi yang menggunakan konsep *object oriented* [6]. Ketidaksesuaian tersebut adalah sebagai berikut :

1. Tingkat kedetailan data (*granularity*)
   Tingkat kedetailan data pada sebuah atribut dapat memiliki pengaruh terhadap penggunaan tipe datanya. Satu atribut pada sebuah kelas dengan tipe data yang kompleks, dapat disesuaikan dengan atribut pada basis data.

2. Subtipe (*subtypes*)
   Subtypes yang dimaksud disini adalah pembeda antara *superclass* dan *subclass*. Pada pemrograman berorientasi objek dikenal istilah *inheritance* (pewarisan) dari *parent class* kepada *child class*. Pada basis data relasional, tidak dikenal istilah pewarisan antar tabel.
3. Identitas (*identity*)
   Pada objek, identitas pengenal menggunakan lambang *double equals* (==) yang menunjukkan nilai dan lojik yang sama. Pada basis data relasional, identitas pengenal ditunjukkan dengan *primary key*.
4. Asosiasi (*association*)
   Ketidaksesuaian pada aspek asosiasi meliputi hubungan dua entitas yang memiliki keterhubungan. Pada objek, penghubung dua entitas disebut dengan *references*, sedangkan pada tabel relasional disebut dengan *foreign key*.
5. Navigasi data
   Ketidaksesuaian pada aspek navigasi data meliputi proses pengaksesan suatu objek dari objek lain. Proses pencarian pada basis data menggunakan *join table.* Pada *framework* Yii, proses pemanggilan objek dari objek lain memanfaatkan *method relations* [6].

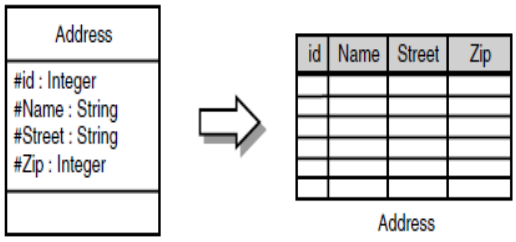
Permasalahan mengenai ketidaksesuaian di atas dapat diselesaikan dengan menggunakan prinsip *Object Relational Mapping* (ORM) seperti yang dipaparkan pada subbab 2.5.

2.4. Prinsip *Object Relational Mapping*

Masalah ketidaksesuaian antara paradigma berorientasi objek dengan tabel relasional muncul karena ada banyak pilihan setiap kelas yang mencakup masalah atribut, asosiasi, serta posisi kelas dalam sebuah hirarki pewarisan (*inheritance*). Solusi dari ketidaksesuaian dapat diselesaikan dengan prinsip ORM sebagai berikut [3] :

2.4.1. *Mapping Class and Attribute*

Pemetaan kelas dan atribut pada sebuah kelas tunggal dapat dilakukan dengan cara memetakan kelas menjadi tabel dan memetakan atribut menjadi *field*. Pemetaan tersebut ditunjukkan oleh gambar 2.1. Pemetaan ini menjadi solusi bagi ketidaksesuaian aspek kedetailan data (*granularity*) dan aspek identitas.



Gambar 2.1. Pemetaan Kelas Menjadi Tabel [3]

1. Solusi Terhadap Ketidaksesuaian Aspek Kedetailan Data (*Granularity*)

Masalah pada aspek tingkat kedetailan data (*granularity*) dapat muncul ketika merancang sebuah kelas ‘Address’. Kita dapat membuat atribut ‘address’ pada kelas ‘Address’ dengan tipe data ‘address’. Tetapi pada tabel relasional tidak mengenal tipe data ‘address’. Solusi dari hal tersebut adalah dengan memecah atribut ‘address’ menjadi beberapa atribut [6], misalnya atribut ‘name’, ‘street’, dan ’zip’ pada kelas ‘Address’ seperti yang ditunjukkan oleh kelas pada gambar 2.1.

Solusi pada gambar 2.1, diimplementasikan pada kelas dengan cara memetakan tabel ‘Address’ menjadi implementasi kelas ‘Address’. Implementasi kelas ‘Address’ memanfaatkan *Object-Relational Middleware* (*OR-Middleware*) *Framework* Yii dengan menggunakan fungsi tableName() dan rules() [9]. Fungsi tersebut ditunjukkan oleh tabel 2.1.

Tabel 2.1. Implementasi Kelas Address

```
class Address extends CActiveRecord{
  /* @return string the associated database
   table name */
  public function tableName(){
    return 'Address';
  }
  /* @return array validation rules for
  model attributes.*/
  public function rules(){
    array('id,Name,Street,Zip', 'required'
    ),
  }
}
```

2. Solusi Terhadap Ketidaksesuaian Aspek Identitas

Masalah pada aspek identitas muncul karena ada perbedaan antara baris (*row*) pada tabel dengan sebuah objek karena keduanya memiliki konsep

pengenal identitas yang berbeda. Dalam *object oriented* (OO), setiap objek memiliki pengenal atau *object identifier* (OID) yang unik. OID berguna sebagai properti tunggal yang tidak dapat diubah. Sedangkan pengenal pada baris (*row*) yaitu kunci primer (*primary key*) yang dapat diidentifikasi secara unik diantara semua baris tabel dan bisa dimodifikasi [3]. Masalah aspek identitas yang terjadi pada kelas gambar 2.1, solusinya adalah membuat sebuah OID unik yang memiliki sifat sebagai *primary key* ketika mengakses tabel.

Solusi membuat OID pada atribut 'id' kelas 'Address' gambar 2.1, diimplementasikan pada kelas dengan cara memetakan *field* 'id' menjadi implementasi atribut 'id' pada kelas 'Address'. Implementasi tersebut memanfaatkan *OR-Middleware Framework* Yii dengan menggunakan fungsi rules() [9]. Fungsi tersebut ditunjukkan oleh tabel 2.1. Proses tersebut dilakukan dengan syarat *field* 'id' pada tabel sudah bersifat *primary key*.

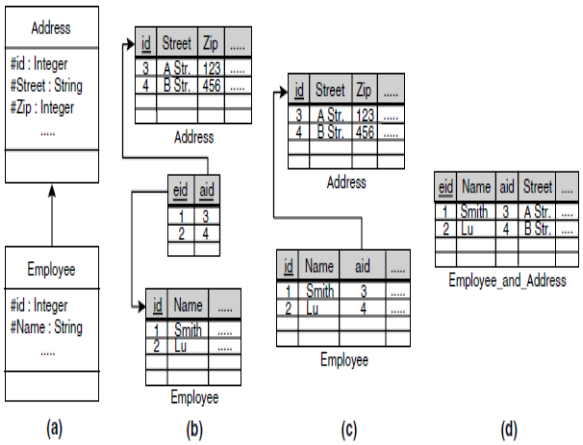
2.4.2. *Association Mappings*

Pembahasan *association mappings* berfokus pada pemetaan keterhubungan *one-to-one*, *one-to-many* dan *many-to-many* [3]. Sebuah pemetaan asosiasi, mampu menjadi solusi bagi ketidaksesuaian pada aspek asosiasi dan aspek navigasi data.

1. Solusi Terhadap Ketidaksesuaian Aspek Asosiasi dan Navigasi Data

1.1. Asosiasi *One-to-one*

Asosiasi *one-to-one* yang ditunjukkan pada gambar 2.2(a). dapat diselesaikan dengan tiga cara. Cara yang pertama yaitu memetakan kedua kelas yang berasosiasi menjadi hubungan yang terpisah seperti yang ditunjukkan oleh gambar 2.2(b). Keuntungan dari pendekatan ini adalah mudah dipahami, dan dapat memelihara skema. Modifikasi kardinalitas dalam model objek dapat dipetakan ke dalam skema relasional dengan mudah [3].



Gambar 2.2. Pemetaan Asosiasi *One-to-one* [3]

Cara yang kedua adalah memetakan kedua kelas menjadi dua tabel yang terpisah. Kunci tamu disisipkan ke dalam tabel sumber seperti yang ditunjukkan oleh gambar 2.2(c) [3].

Cara yang ketiga adalah menggabungkan kedua kelas yang berasosiasi menjadi sebuah tabel seperti yang ditunjukkan oleh gambar 2.2(d).

Apabila diinginkan sebuah solusi pemetaan yang memiliki kesesuaian antara kelas dengan tabel, maka asosiasi *one-to-one* pada kelas dipetakan menjadi *one-to-one* pada asosiasi tabel. Asosiasi tabel tersebut ditunjukkan oleh gambar 2.2(c).

Solusi dari ketidaksesuaian asosiasi *one-to-one* diimplementasikan ke dalam kelas dengan memanfaatkan *OR-Middleware Framework* Yii. Implementasi dilakukan dengan menggunakan fungsi relations() dan atribut jenis relasi [9]. Atribut jenis relasi 'HAS_ONE' diimplementasikan pada kelas 'Address' dan ditunjukkan oleh tabel 2.2. Atribut jenis relasi 'BELONGS_TO' diimplementasikan pada kelas 'Employee' dan ditunjukkan oleh tabel 2.3.

Solusi dari ketidaksesuaian navigasi data adalah dengan memanfaatkan *OR-Middleware Framework* Yii berupa atribut relasi pada fungsi relations() [9]. Atribut relasi 'employeeRelation' diimplementasikan pada kelas 'Address' dan ditunjukkan oleh tabel 2.2. Atribut relasi 'addressRelation' diimplementasikan pada kelas 'Employee' dan ditunjukkan oleh tabel 2.3. Dengan adanya atribut relasi, sebuah objek 'Address' dapat dipanggil melalui objek 'Employee'.

Tabel 2.2. Fungsi Relasi pada Kelas Address

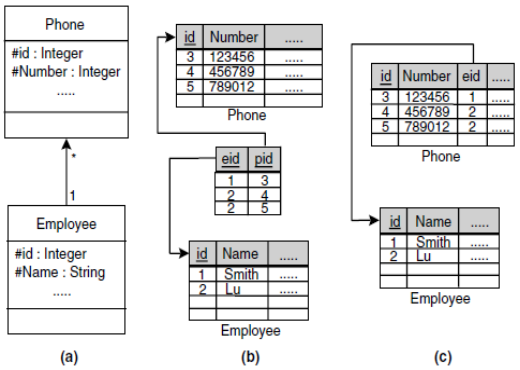
```
class Address extends CActiveRecord{
  public function relations(){
    return array(
```

```
  'employeeRelation'=>array  (self::HAS_ONE,
     'Employee,'aid'),);}}
```

Tabel 2.3. Fungsi Relasi pada Kelas Employee

```
class Employee extends CActiveRecord{
  public function relations(){
    return array(
     'addressRelation'=>array(
      self::BELONGS_TO,'Address','aid'),);}}
```

1.2. Asosiasi *One-to-many*



Gambar 2.3. Pemetaan Asosiasi *One-to-many* [3]

Asosiasi *one-to-many* yang ditunjukkan pada gambar 2.3(a) dapat diselesaikan dengan dua cara. Cara yang pertama yaitu memetakan kedua kelas yang berasosiasi menjadi hubungan yang terpisah seperti pada gambar 2.3(b).

Cara yang kedua yaitu menyisipkan kunci tamu ke dalam tabel target seperti yang ditunjukkan oleh gambar 2.3 (c). Apabila diinginkan sebuah solusi pemetaan yang memiliki kesesuaian antara kelas dengan tabel, maka asosiasi *one-to-many* pada kelas dipetakan menjadi asosiasi *one-to-many* antar tabel seperti yang ditunjukkan oleh gambar 2.3 (c).

Solusi *one-to-many* diimplementasikan ke dalam kelas dengan memanfaatkan *OR-Middleware Framework* Yii. Implementasi dilakukan dengan menggunakan fungsi relations() dan atribut jenis relasi [9]. Atribut jenis relasi `'HAS_MANY'` diimplementasikan pada kelas 'Employee' dan ditunjukkan oleh tabel 2.4. Atribut jenis relasi `'BELONGS_TO'` diimplementasikan pada kelas 'Phone' dan ditunjukkan oleh tabel 2.5.

Solusi dari ketidaksesuaian navigasi data adalah dengan memanfaatkan *OR-Middleware Framework* Yii berupa atribut relasi pada fungsi relations() [9]. Atribut relasi `'phoneRelation'` diimplementasikan pada kelas 'Employee' dan ditunjukkan oleh tabel 2.4. Atribut relasi `'employeeRelation'` diimplementasikan pada kelas 'Phone' dan ditunjukkan oleh tabel 2.5. Dengan adanya atribut relasi, sebuah objek 'Employee' dapat dipanggil melalui objek 'Phone'.

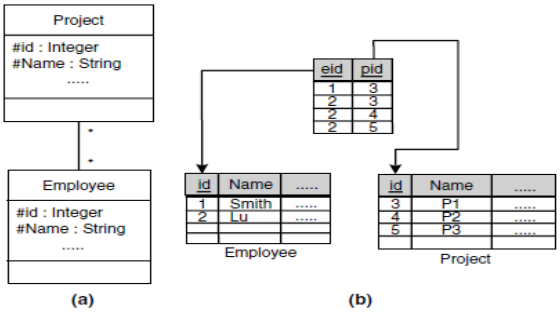
Tabel 2.4. Fungsi Relasi pada Kelas Employee

```
class Employee extends CActiveRecord{
  public function relations(){
    return array('phoneRelation'=>array
      (self::HAS_MANY,'Phone','eid'),);}}
```

Tabel 2.5. Fungsi Relasi pada Kelas Phone

```
class Phone extends CActiveRecord{
  public function relations(){
    return array(
      'employeeRelation'=>
       array(self::BELONGS_TO,'Employee',
       'eid' ),); }}
```

1.3. Asosiasi *Many-to-many*



Gambar 2.4. Pemetaan Asosiasi *Many-to-many* [3]

Asosiasi *many-to-many* pada gambar 2.4(a) dapat diselesaikan dengan cara membuat tabel terpisah yang menyimpan kunci primer dari kedua tabel ditunjukkan pada gambar 2.4(b). Tabel yang menyimpan kunci primer dari kedua tabel tersebut dapat dimisalkan dengan nama tabel 'Contract'.

Solusi dari ketidaksesuaian asosiasi *one-to-many* diimplementasikan menjadi kelas dengan memanfaatkan *OR-Middleware Framework* Yii. Implementasi dilakukan dengan menggunakan fungsi relations() dan atribut jenis relasi [9]. Atribut jenis relasi 'HAS_MANY' diimplementasikan pada kelas 'Project' dan ditunjukkan oleh tabel 2.6. Atribut jenis relasi 'HAS_MANY' diimplementasikan pada kelas 'Employee' dan ditunjukkan oleh tabel 2.7. Karena tabel 'Contract' menyimpan kunci primer dari tabel 'Project' dan 'Employee', maka dideklarasikan dua atribut jenis relasi 'BELONGS_TO' yang diimplementasikan

pada kelas 'Contract' dan ditunjukkan oleh tabel 2.8.

Tabel 2.6. Fungsi Relasi pada Kelas Project

```
class Project extends CActiveRecord{
  public function relations(){
    return array(
      'contractsRelation'=>
      array(self::HAS_MANY,'Contract','pid
      '),);}}
```

Tabel 2.7. Fungsi Relasi pada Kelas Employee

```
class Employee extends CActiveRecord{
  public function relations(){
    return array(
     'contractsRelation'=>
      array(self::HAS_MANY,'Contract','eid
      '),);}}
```

Tabel 2.8. Fungsi Relasi pada Kelas Contract

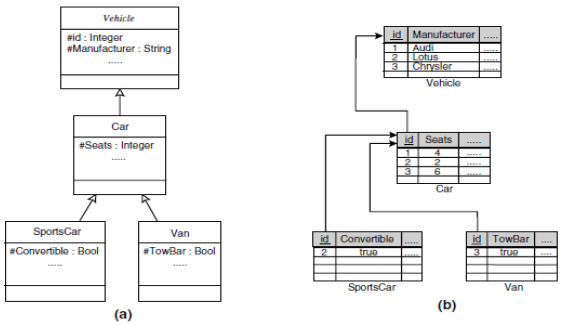
```
class Contract extends CActiveRecord{
  public function relations(){
    return array(
     'eRelation'=>
      array(self::BELONGS_TO,'Employee','
      eid'),
      'pRelation'=>a
      rray(self::BELONGS_TO,'Project','pid
      '),);}}
```

2.4.3. *Mapping of Inheritance Relationships*

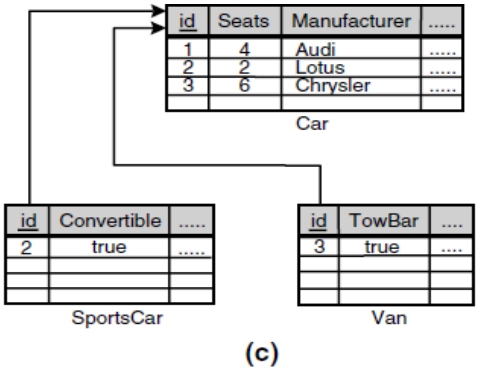
Pembahasan *mapping of inheritance relationships* berfokus pada pemetaan keterhubungan pewarisan antar kelas [3]. Sebuah *mapping of inheritance relationships*, mampu menjadi solusi bagi ketidaksesuaian pada aspek subtipe dengan cara melakukan pendekatan mengenai pemetaan pewarisan antar kelas.

Ada beberapa cara untuk memetakan pewarisan kelas menjadi tabel. Kelas yang dipetakan ditunjukkan oleh gambar 2.5(a). Cara yang pertama adalah menetapkan sebuah tabel untuk setiap kelas, yang berisi *field* tunggal untuk setiap atribut ditunjukkan pada gambar 2.5(b). Keuntungan dari cara ini adalah kemudahan dalam perawatan kelas, karena perubahan setiap kelas hanya berpengaruh terhadap satu tabel. Pengambilan sebuah objek membutuhkan *join* yang mungkin menjadi hambatan [3].

.Cara yang kedua ditunjukkan oleh gambar 2.6(c). Hal tersebut dilakukan dengan cara mewariskan atribut dari *superclass* kepada setiap tabel yang memiliki atribut sesuai dengan kelasnya. Kinerja dengan pendekatan ini tergantung penggunaan kelas abstrak. Adanya modifikasi kelas abstrak menyebabkan terjadinya penyesuaian yang tidak hanya terjadi pada satu tabel, tetapi semua tabel yang sesuai dengan sub kelas dari kelas yang berubah [3].



Gambar 2.5. Alternatif Pemetaan Pewarisan [3]



Gambar 2.6. Satu Tabel Setiap *Concrete Class* [3]

Pendekatan pemetaan pewarisan antar kelas menjadi tabel seperti yang ditunjukkan oleh gambar 2.5(b) dan 2.6(c) diimplementasikan menjadi kelas dengan memanfaatkan *Object-Relational Middleware Framework* Yii. Implementasi tersebut dilakukan seperti yang dijelaskan pada subbab 2.4.2. tahap implementasi kelas pada pemetaan asosiasi.

2.5. Implementasi *Object Relational Mapping*

Konsep *Object Relational Mapping* (ORM) yang utama adalah memetakan dari tabel menjadi objek. *Columns* (Kolom-kolom) yang berada pada tabel diubah menjadi atribut-atribut objek tersebut. Satu objek mewakili satu *row* (baris) dalam tabel [9].

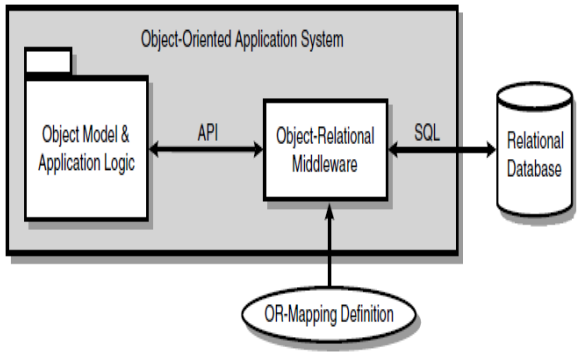
Pada gambar 2.7. dijelaskan bahwa pengaksesan basis data relasional dapat dilakukan dengan cara memasukkan pernyataan SQL ke dalam kode

aplikasi, tetapi untuk model objek yang lebih kompleks antara aplikasi dan skema relasional, penulisan SQL secara langsung dinilai tidak cocok, disebabkan karena seringnya pergantian maupun modifikasi dari skema dalam kode aplikasi [3].

Enkapsulasi pernyataan SQL menjadi kelas data tersendiri adalah pendekatan baik dari segi arsitektur, tetapi masih memungkinkan adanya perubahan skema relasional. Pendekatan yang lebih baik yaitu menggunakan ORM dengan memanfaatkan *middleware* yang memisahkan basis data relasional dengan aplikasi berorientasi objek yang ditunjukkan oleh gambar 2.7.

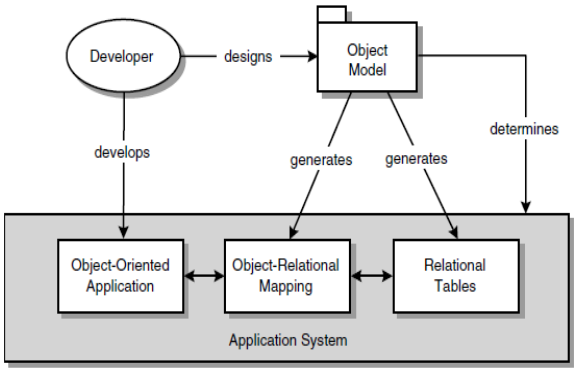
Middleware adalah perangkat lunak komputer yang menyediakan layanan untuk aplikasi perangkat lunak di luar yang tersedia dari sistem operasi. *Middleware* yang digunakan pada ORM memiliki fokus pada pengolahan sintaks SQL pada basis data menjadi API (*Application Programming Interface*) yang digunakan pada aplikasi yang dikembangkan.

API adalah sebuah bahasa dan format pesan yang digunakan oleh sebuah aplikasi untuk berkomunikasi dengan : sistem operasi, atau program lain. API digunakan melalui pemanggilan fungsi API pada program. API pada *OR-Middleware* digunakan untuk mengolah data pada logika aplikasi.



Gambar 2.7. Enkapsulasi Sebuah RDBMS dengan *OR-Middleware* [3]

Pada gambar 2.8. dijelaskan bahwa tahap pertama pada pengembangan sistem adalah pengembangan model objek sesuai kebutuhan. Ditentukan kelas yang harus dijadikan sebagai *persistent* [3]. Fungsi operasi terhadap atribut kelas *persistent* diletakkan pada ORM tepatnya pada *OR-Middleware.* Model objek yang dijadikan sebagai kelas *persistent* dijadikan tabel relasional.



Gambar 2.8. Pengembangan Sistem Aplikasi Sederhana Mengunakan ORM [3]

2.6. *Framework* Yii

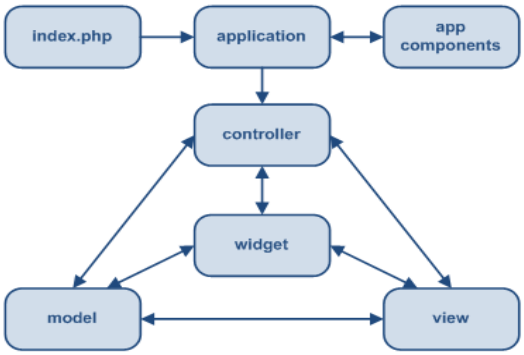
Framework adalah kumpulan kelas dan fungsi (*function, method*) yang disusun secara sistematis berdasarkan kegunaan atau fungsionalitas tertentu untuk mempermudah pengembangan suatu aplikasi.

Framework Yii menggunakan pola arsitektur *model-view-controller* (MVC). Dasar dari arsitektur MVC adalah pemisahan antara logika aplikasi dengan tampilan. *Framework* Yii juga menggunakan pola *Active Record* (AR) pada bagian *model.*

Dasar dari arsitektur AR membungkus struktur basis data ke dalam sebuah kelas dan mengimplementasikan metode pengakses untuk setiap kolom dalam tabel. Bagian utama *Framework* Yii ditunjukkan oleh gambar 2.9. adalah :

1. ***Model*** bertanggung jawab terhadap validasi data. Di dalamnya dituliskan perintah untuk mengambil, mengubah, menghapus dan menambahkan data.
2. ***View*** merupakan tempat untuk meletakkan apa yang ingin ditampilkan di halaman perambah (*browser*). Sebuah berkas *view* umumnya berisi kode hahasa pemrograman sisi klien.
3. ***Controller*** merupakan pengatur utama hubungan antara *model*, *view*, dan juga sumber daya lain yang tersedia.
4. ***Application*** merupakan sebuah berkas yang berisi pengaturan terhadap perangkat lunak. Aturan yang terdapat pada *application* antara lain aturan terhadap nama aplikasi, pemanggilan *model*, komponen yang digunakan, format *url*, basis data, dan kontrol *error*.
5. ***App components*** atau komponen perangkat lunak adalah unit komposisi dengan kontrak atau aturan yang ditentukan oleh *interface*. Sebuah kompenen memiliki konteks penggunaan yang jelas dan bisa berdiri sendiri.
6. ***Widget*** merupakan sebuah komponen yang berguna adalah untuk mengatur tampilan. *Widget*

dapat berdiri sendiri dan biasanya tertanam dalam *script* yang kompleks namun memiliki antarmuka yang berdiri sendiri.



Gambar 2.9. Struktur *Static Framework* Yii [9]

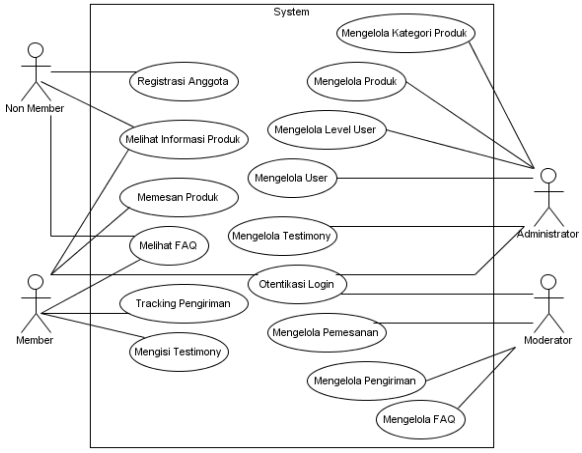
3. Analisis dan Perancangan

3.1. Definisi Kebutuhan Perangkat Lunak

Subbab ini dipaparkan mengenai definisi kebutuhan perangkat lunak. Kebutuhan atau *requirement* bertujuan untuk mendefinisikan lingkungan proyek, menangkap kebutuhan yang esensial dan mengetahui bisnis proses.

3.1.1. Use Case Diagram

Gambar 3.1 adalah *use case diagram* pada perangkat lunak *e-commerce.*



Gambar 3.1. *Use Case Diagram E-commerce*

3.1.2. Use Case Detail

Use case detail berisi penjelasan dari suatu *use case* yang meliputi *actor* yang berinteraksi dengan *use case*, kondisi awal, kondisi akhir dan skenario utama, skenario abnormal. Tabel 3.1 merupakan *use case detail* dari *use case* melihat informasi produk.

Tabel 3.1. *Use Case Detail* Melihat Informasi Produk

Nomor *use case* : 2
Nama *use case* : Melihat informasi produk
Aktor : *Non-member*, *Member*
Kondisi awal : *Non-member* atau *member* belum bisa melakukan transaksi pembelian
Skenario utama : 1. *Non-member* atau *member* memilih menu produk. 2. *Non-member* atau *member* menekan tombol detail untuk melihat informasi produk. 3. *Non-member* atau *member* menekan tombol *add to cart* untuk memasukkan produk ke daftar produk yang ingin dibeli dan melihat total harga.
Skenario abnormal : -
Kondisi akhir : *Shopping cart* terisi
Sketsa antarmuka : -

3.2. Analisis

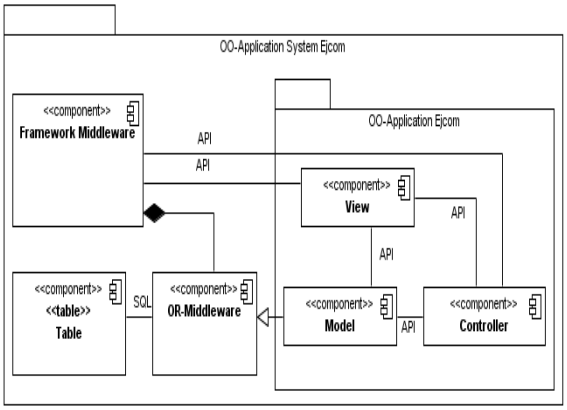
Subbab ini dipaparkan mengenai analisis perangkat lunak. Analisis atau *analysis* bertujuan menciptakan arsitektur sistem yang akan dieksekusi.

3.2.1. Analisis Pengembangan Perangkat Lunak Menggunakan ORM

Subbab ini dipaparkan mengenai pengembangan perangkat lunak ditunjukkan oleh *component diagram* pada gambar 3.2. Pengembangan perangkat lunak tahap analisis dibagi menjadi 3 bagian secara urut yaitu :

1. Analisis logika aplikasi *Model-View-Controller* (MVC) pada *framework* Yii menghasilkan realisasi *use case* tahap analisis. Analisis ini bertujuan untuk mengetahui kelas *boundary*, *control*, dan *entity*.
2. Analisis *Object Relational Middleware* (*OR-Middleware*) yang digunakan pada *middleware framework* Yii. Analisis ini bertujuan untuk menunjukkan kelas-kelas dan fungsi-fungsi pada *middleware framework* yang digunakan.
3. Analisis tabel relasional yang digunakan. Analisis ini berfungsi untuk mengetahui tabel-tabel yang digunakan. Analisis tabel dibuat berdasarkan analisis kelas dengan jenis kelas *entity*.

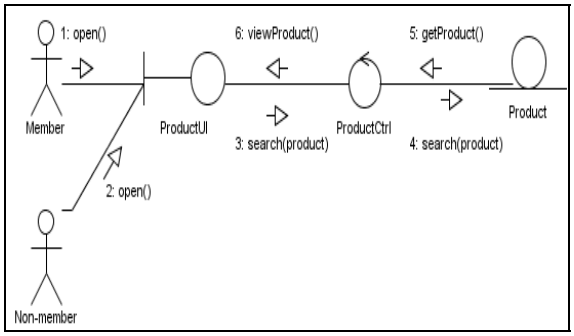
Fungsi	Tanggung jawab
findAll()	Menampilkan semua data dari tabel tertentu
findAllById()	Menampilkan satu baris data yang memiliki *primary key* tertentu
save()	Melakukan *insert* maupun *update* data
delete()	Menghapus data yang memiliki *primary key* tertentu



Gambar 3.2. Analisis Pengembangan Perangkat Lunak Menggunakan ORM

3.2.1.1. Analisis Logika Aplikasi *MVC*

Analisis logika aplikasi menghasilkan *communication diagram*. *Communication diagram* yang dihasilkan yaitu *communication diagram* melihat informasi produk yang ditunjukkan oleh gambar 3.3.



Gambar 3.3. *Communication Diagram* Melihat Informasi Produk

3.2.1.2. Analisis *Object Relational Middleware*

Subbab ini dipaparkan mengenai kelas dan fungsi utama *OR-Middleware*. Hasil kelas pada tahap analisis *OR-Middleware* ditunjukkan oleh tabel 3.2.

Tabel 3.2. *Analysis Class* pada *Framework Middleware* Yii

Class	Tanggung jawab
ClassModel	Menyediakan fungsi-fungsi SQL untuk *insert*, *update*, *delete* dan *retrieve* data.

Diperlukan analisis terhadap tanggung jawab masing-masing fungsi pada kelas ClassModel. Analisis fungsi tersebut dapat dilihat pada tabel 3.3.

Tabel 3.3. *Analysis* Fungsi pada *Framework Middleware* Yii

3.2.1.3. Analisis Tabel Relasional

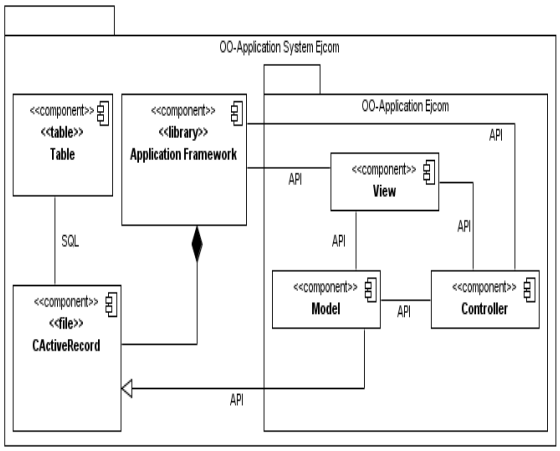
Tabel relasional tahap analisis dihasilkan dari *communication diagram* dengan jenis kelas *entity*. Pada*communication diagram* melihat informasi produk dihasilkan tabel *product*.

3.3. Perancangan

Subbab ini dipaparkan mengenai perancangan perangkat lunak. Perancangan atau *design* bertujuan menciptakan arsitektur sistem yang akan dieksekusi

3.3.1. Perancangan Perangkat Lunak Menggunakan *OR-Middleware* Yii

Subbab ini dipaparkan mengenai pengembangan perangkat lunak tahap perancangan. Pengembangan ini ditunjukkan oleh *component diagram* pada gambar 3.4.



Gambar 3.4. Perancangan Pengembangan Perangkat Lunak Menggunakan Yii *Framework*

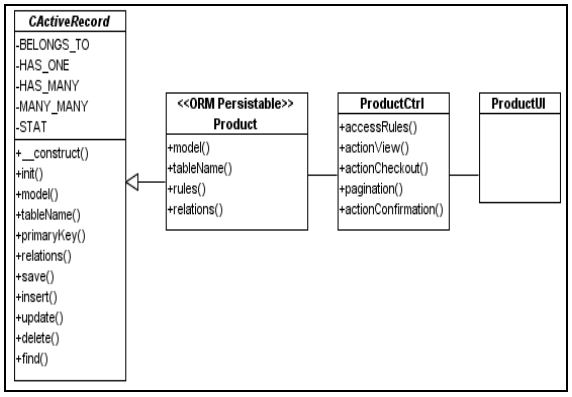
Pengembangan perangkat lunak tahap perancangan dibagi menjadi 3 bagian secara urut yaitu :

1. Perancangan mengenai logika aplikasi *Model-View-Controller* (MVC) pada *framework* Yii. Menghasilkan *use case realization* tahap perancangan.
2. Perancangan *OR-Middleware* yang digunakan pada *middleware framework* Yii.
3. Perancangan tabel relasional.

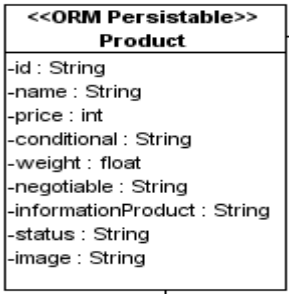
Penyatuan logika aplikasi dengan *OR-Middleware Framework* Yii ditunjukkan dengan class diagram pada realisasi *use case* tahap perancangan.

3.3.1.1. Perancangan Logika Aplikasi MVC dan *OR-Middleware* Yii *Framework*

Salah satu realisasi *use case* yaitu melihat informasi produk ditunjukkan oleh *class diagram* gambar 3.5. Kelas *controller* ProductCtrl menjembatani pengaksesan *persistable class* Product dengan ProductUI. Kelas ProductCtrl bertugas melakukan operasi data oleh *member* atau *non-member* yaitu menampilkan daftar produk. Perancangan *OR-Middleware Framework* Yii ditunjukkan oleh kelas CActiveRecord gambar 3.5.



Gambar 3.5. Diagram Kelas Logika Aplikasi dan CActiveRecord



Gambar 3.6. Diagram Kelas Product

3.3.1.2. Perancangan Tabel Relasional

Perancangan tabel relasional didapat dari diagram kelas. Perancangan tabel relasional product dihasilkan dari perancangan kelas Product. Perancangan tabel relasional product ditunjukkan oleh *ERD* pada gambar 3.7.

product		
+id	**varchar(15)**	**Nullable = false**
#idCategory	*varchar(15)*	*Nullable = false*
name	varchar(30)	Nullable = false
price	int(10)	Nullable = false
conditional	varchar(10)	Nullable = false
weight	float	Nullable = false
negotiable	varchar(3)	Nullable = false
informationProduct	text	Nullable = false
status	varchar(5)	Nullable = false
image	varchar(30)	Nullable = false

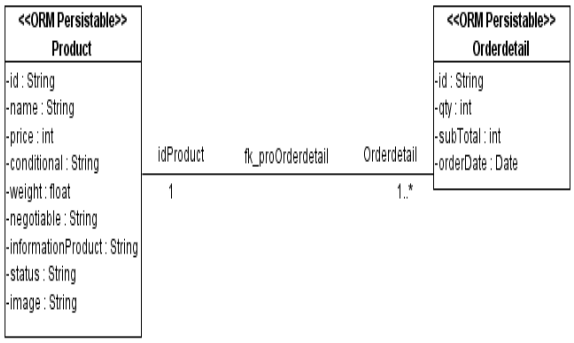
Gambar 3.7. *ERD* product

4. Implementasi dan Pengujian

Bab ini berisi tahap implementasi dan pengujian dalam pengembangan perangkat lunak *e-commerce*.

4.1. Implementasi Kelas *ORM Persistable* untuk Menangasi Navigasi Data

Relasi antara kelas orderdetail dengan kelas product adalah *many-to-one* yang ditunjukkan oleh gambar 4.1.



Gambar 4.1. *Relationship* Kelas Orderdetail dengan Kelas Product

Salah satu ketidaksesuian antara tabel relasional dengan objek adalah masalah navigasi data. Solusi dari navigasi data pada gambar 4.1. adalah mengimplementasikan *relational active record* (RAR). RAR yang diimplementasikan pada kelas Orderdetail adalah `BELONGS_TO` sedangkan pada kelas Product adalah `HAS_MANY`. Masing-masing ditunjukkan oleh tabel 4.1. dan 4.2.

Kode 4.1. Implementasi Fungsi Relations *Class* ORM Orderdetail

```
class Orderdetail extends CActiveRecord{
    public function relations(){
```

Lanjutan Kode 4.1. Implementasi Fungsi Relations *Class* ORM Orderdetail

```
        return array(
          'idProduct0' => array(self::
          BELONGS_TO,
          'Product',
          'idProduct'),);}}
```

Kode 4.2. Implementasi Fungsi Relations *Class* ORM Product

```
class Product extends CActiveRecord{
    public function relations(){
       'orderdetails' => array(self::
       HAS_MANY,
       'Orderdetail',
       'idProduct'),);} }
```

Apabila ingin dilakukan pengaksesan atribut pada objek produk tetapi melalui objek orderdetail, hal tersebut dapat dilakukan dengan memanggil atribut jenis relasi `idProduct0` seperti yang ditunjukkan oleh kode 4.3.

Kode 4.3. Implementasi Relationship pada RAR

```
<?php foreach($orderdetail as $column):?>
   <tr>
      <td><?php echo $column->idProduct0
      ->name; ?>
      </td>
   </tr>
<?php endforeach;?>
```

4.2. Pengujian

4.2.1. Identifikasi Pengujian

Dalam pengujian ini digunakan teknik pengujian *black box*. Teknik yang digunakan dalam pengujian *black box* antara lain :

1. Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang .
2. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan untuk fungsi yang ada tanpa melihat bagaimana proses mendapatkan keluaran tersebut.
3. Dari keluaran yang dihasilkan, kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus dapat diketahui kesalahan-kesalahannya.

4.2.2. Analisis Hasil Uji

Dari hasil pengujian terhadap implementasi *Object Relational Mapping* pada perangkat lunak *e-commerce* menggunakan *framework* Yii dapat diketahui bahwa perangkat lunak ini telah memenuhi untuk :

1. Melakukan otentikasi pengguna
2. Menjaga integritas data
3. Melakukan validasi data

5. Kesimpulan dan Saran

5.1. Kesimpulan

Implementasi *Object Relational Mapping* mampu menghilangkan ketidaksesuaian penggunaan tabel relasional dengan paradigma *object oriented*. Permasalahan ketidaksesuaian pada aspek kedetailan data, identitas, asosiasi dan navigasi data berhasil diselesaikan dengan melakukan implementasi fungsi-fungsi dari kelas CActiveRecord yang terdapat pada *framework* Yii.

5.2. Saran

1. Perlu dikembangkan lagi sebuah ORM yang secara *drag-and-drop* mampu menghasilkan sebuah kelas ORM yang memiliki tingkat rincian tertentu, misalkan atribut yang unik dan atribut email.
2. Penambahan fitur otomatisasi untuk melakukan konfirmasi pembelian produk maupun informasi penting lainnya melalui *e-mail member*.

REFERENSI

[1] Arwananingtyas, Zahra. 2010. *Implementasi E-Commerce Untuk Ozone Distro*. Semarang: Fakultas MIPA Universitas Diponegoro.
[2] Nugroho, Adi. 2010. *Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP (Unified Software Development Process)*. Yogyakarta: Penerbit Andi.
[3] Philippi, Stephan. 2004. *Model Driven Generation and Testing of Object-Relational Mapping*. Koblenz: Departement of Computer Science, University of Koblenz.
[4] Pressman, R.S. 2001. *Software Engineering : A Practitioner's Approach* 5th ed. New York: McGraw-Hill.
[5] Romadani. 2011. *Implementasi Teknologi ORM Menggunakan Framework Hibernate yang Diterapkan pada Aplikasi Peminjaman.* Jakarta : Universitas Gunadarma.
[6] Savitri, Dian Indah. 2008. *Penerapan Object Relational Mapping Pada Pengembangan Enterprise Resource Planning*. Bogor : Institut Pertanian Bogor.
[7] Suyanto, M. 2003. *Strategi Periklanan pada E-commerce Perusahaan Top Dunia*. Yogyakarta : Penerbit ANDI.
[8] Widodo, Aris Puji, et all. 2004. *Buku Ajar Basis Data*. Semarang: Jurusan Matematika Fakultas MIPA Universitas Diponegoro.
[9] Xue, Qiang, dan Xiang Wei Zhuo. 2008. *The Definitive Guide to Yii 1.1*. [t.t] : Yii Software LLC.