Code Generation untuk Object Relational Mapping Dengan Menggunakan LINQ dan Text Templating

Ridwan Rismanto, S.ST., M.Kom.
Jurusan Teknologi Informasi, Politeknik Negeri Malang
Email: rismanto@polinema.ac.id

Abstract

The increase towards Object Oriented Design creates a big gap between OO and relational database. In the other side, relational database still a big favor in designing business-powered application. Developers use Object Relational Mapping (ORM) to fill the gap between OO and RDBMS but still, the problem in the association of RDBMS and OO exists which caused underestimation towards ORM methodology. This research provides a solution to detect relationship inside tables of database, generate LINQ query to map the field of tables to the attribute of the entity classes. Using T4 text templating we are able to generate the whole Data Access Object class which provides transparent encapsulation and relational mapping for the system.

Keywords *: Object Relational Mapping, Relational Database, Object Technology, LINQ, Multi Tier Architecture, Text Templating, T4*

1. PENDAHULUAN

Relational Database (RDBs) adalah tulang punggung dari sistem informasi dan tidak seorangpun mengetahui apakah akan ada penggantnya [Atzeni et al., 2013]. Sementara itu dilain sisi, desain berorientasi objek semakin banyak diadopsi. Banyak pengembang aplikasi yang menjembatani gap antara desain berbasis *relational database* dengan *object oriented design* dengan Object-Relational Mapping. Akan tetapi, permasalahan *Impedance Mismatch Problem* (IMP) [Atkinson & Buneman, 1987] masih terus mengikuti setiap perkembangan desain aplikasi berorientasi object yang selalu menghasilkan beberapa halangan atau kesulitan pada Object-Relational Mapping (ORM).

Dalam kurun waktu beberapa tahun belakangan ini, perkembangan dan implementasi ORM framework yang didukung oleh pengembang *platform* sistem informasi seperti Java, C#, Phyton dan Ruby on Rails semakin pesat. Framework-framework tersebut mengembangkan ORM dengan mengikuti *design pattern* yang sudah teruji dan terbukti kehandalannya.

Salah satu permasalahan yang ada para Object Relational Mapping adalah pada *associations*. Pada domain model, *associations* merepresentasikan hubungan atau *relationship* antara entitas-entitas yang ada dan juga digunakan untuk navigasi data. Pada paradigma OOP, asosiasi direpresentasikan sebagai *object references*, akan tetapi, asosiasi pada relational database direpresentasikan melalui *foreign key*.

Asosiasi antar object sifatnya adalah *unidirectional.* Sedangkan asosiasi pada basis data adalah *bi-directional* karena menggunakan *foreign key* untuk melakukan *join* pada tabel. [Bauer, C. & King, G., 2007]

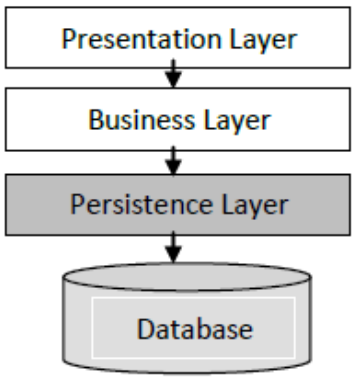
Penelitian ini bertujuan untuk mendeteksi tabel-tabel yang berelasi pada relational database, dan men-setting asosiasi pada object pada desain sistem berorientasi objek dengan cara men-generate kode program pada Data Access Object yang menjembatani antara entity class dengan relational database pada layer *persistence*. Dengan demikian maka proses mapping pada ORM dapat dilakukan pada tabel-tabel yang berelasi.

2. KAJIAN LITERATUR

2.1 Layered Software Architecture

Pembangunan perangkat lunak dengan paradigma berorientasi object menggunakan model multi-layer pada arsitekturnya, yang membagi perangkat lunak menjadi beberapa layer. Pada umumnya, layer-layer tersebut terbagi menjadi tiga, yaitu:

1. Layer Presentasi
2. Layer Bisnis/logic
3. Layer Persistence/penyimpanan data



Gambar 1. Arsitektur Multi-layer [Bauer, C. & King, G., 2007]

2.2 Object Persistence

Paradigma berorientasi objek merupakan model domain sebagai diagram *class* dan *object model*. Dalam sebuah aplikasi berorientasi objek, *persistence* memungkinkan keadaan/state suatu objek yang akan disimpan ke sebuah media penyimpanan dan keadaan yang sama dari objek dapat dibaca kembali. Hal ini tidak terbatas hanya untuk objek tunggal, banyak objek yang saling berhubungan satu sama lain dapat disimpan *state*-nya dan kemudian dibaca kembali. [Bauer, C. & King, G., 2007]

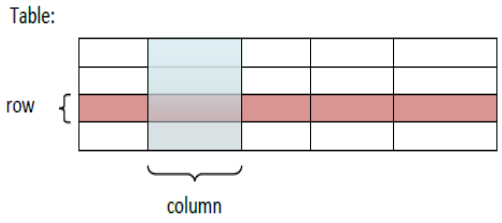
2.2 Relational Database

Hampir semua jenis aplikasi perangkat lunak membutuhkan *data persistence* atau dengan kata lain penyimpanan data. Hingga saat ini, Relational Data Base Management System (RDBMS) masih menjadi teknologi yang dominan pada organisasi-organisasi yang membutuhkan manajemen dan pengolahan data. RDBMS berbasis *relational model* ini pertama kali diperkenalkan oleh E. F. Codd pada tahun 1970.

Menurut konsep relational model, data direpresentasikan dengan simbol matematis *n-ary relations*. Sebuah relasi didefinisikan sebagai n-set tabel. Sebuah tabel adalah beberapa set nilai atribut dan sebuah atribut adalah pasangan dari nama atribut dan tipe datanya.

Relational model memungkinkan data dioperasikan dengan menggunakan operasi *relational* antara lain: *Restriction, projection, cartesian* dan *relational join.* Dengan menggunakan *relational model* memungkinkan kita untuk mendefinisikan struktur data dan aturan-aturan (*constraint*) dan memastikan integritas data. [Bauer, C. & King, G., 2007]

Application Programming Interface (API) untuk berinteraksi dengan *relational database model* adalah SQL (*Structured Query Language*). Bahasa SQL mereferensikan konsep-konsep *relational model* dengan beberapa pernyataan: sebuah relasi adalah sebuah *tuple*, sebuah *tuple* adalah baris didalam tabel dan sebuah atribut adalah sebuah kolom didalam tabel, seperti terlihat pada Gambar 2.



Gambar 2. Tabel dalam RDBMS

2.3 Object Relational Mapping

Object Relational Mapping (ORM) berarti memetakan (*mapping*) object kedalam tabel pada *relational database*. Mapping tersebut dilakukan dengan menggunakan *metadata* yang mendeskripsikan relasi antara beberapa atribut objek dan kolom pada tabel. Ada banyak sekali tool-tool ORM yang tersedia antara lain Hibernate dan Toplink. Tool

tersebut memungkinkan kita untuk melakukan setidaknya operasi dasar pada *database* seperti *create, read, update* dan *delete* (CRUD).

Metadata untuk object-relational mapping adalah bagian dari domain model pada aplikasi perangkat lunak. Metatadata menjelaskan kebutuhan dan ketentuan dari bagian untuk melakukan pemetaan (*mapping*) dari entitas data yang ada dan relasinya pada domain basis data pada aplikasi perangkat lunak. Query (atau SQL) yang ditulis terhadap skema basis data yang berhubungan dengan domain model pada sebuah aplikasi perangkat lunak adalah bergantung pada pemetaan/*mapping* yang diatur oleh object-relational mapping metadata. [Lipitsainen, 2011]

ORM merupakan sebuah teknik pemrograman yang memetakan sebuah objek dengan database. Didalam ORM, terdapat dua model yang relevan, yaitu *object model* dan *relational model.* [Mark, 1997]

ORM ini akan menjadi 'jembatan' antara objek yang didefinisikan dalam kode program dengan database, bagaimana objek itu disimpan, diambil, dihapus dan sebagainya. Dengan ORM ini, programmer dibantu untuk melakukan aksi-aksi yang diperlukan terkait komunikasi objek ketika program dijalankan dengan database seperti menyimpan objek, mengambil data objek dari database kemudian ditampilkan, menghapus objek, mengubah objek dan sebagainya.

Sebagai contoh penggunaan adalah ketika ada definisi data berupa person yang memiliki atribut id dan nama. Di database didefinisikan sebuah tabel dengan kolom id dan nama. Dengan tanpa menggunakan ORM, untuk menyimpan objek itu ke database adalah dengan menjalan perintah SQL :

```
"INSERT INTO persons (id, nama) values ('1','ridwan rismanto')".
```

Dengan menggunakan ORM, perintah SQL itu akan digantikan dengan fungsi misalkan save (tergantung fungsi yang disediakan oleh pustaka ORM yang digunakan), sehingga pemanggilannya adalah :

```
Person person = new Person (1, "Ridwan Rismanto");

person.insert();
```

ORM kemudian yang akan memetakan fungsi insert() sama dengan perintah SQL 'INSERT INTO….". [Rismanto, 2015]

2.4 *Entity Class*

Sebuah *entity* atau entitas, adalah sebuah objek *persistence* yang paling sederhana. Pada umumnya sebuah *entity class* merepresntasikan sebuah tabel pada database relasional dan tiap *instance* dari sebuah entitas adalah baris data pada sebuah tabel.

Dalam mewujudkan object-relational mapping, class entity haruslah merupakan *top-level* class, dengan kata lain bukan class turunan (subclass) dari class lain. Sebuah entity class tidak boleh dideklarasikan sebagai *final*.

Setiap entity harus memiliki primary key, yang merepresentasikan identitas dari entitas tersebut tanpa adanya ambiguitas. Sebuah primary key bisa jadi berupa sebuah kunci sederhana, atau kunci komposit. Sebuah primary key harus diwakili dengan property dari entity class. Contoh dari sebuah entity class dapat dilihat pada Gambar 3.

```java
import java.io.Serializable;
import javax.persistence.*;

@Entity
@Table(name = "CUSTOMER")
public class Customer implements Serializable{

 private int id;
 private String name;
 private String address;

 @Id
 public int getId() {
     return id;
 }
 public void setId(int id) {
     this.id = id;
 }
 public String getName() {
     return name;
 }
 public void setName(String name) {
     this.name = name;
 }
 public String getAddress() {
     return address;
 }
 public void setAddress(String address) {
     this.address = address;
 }
 public Customer(){}
 public Customer(int id, String name, String address){
     this.id = id;
     this.name = name;
     this.address = address;
 }
}
```

Gambar 3. Entity Class pada Java [Lipitsainen, 2011]

Dapat dilihat pada Gambar 3 diatas adalah contoh entity class yang ditulis dengan bahasa Java. Class tersebut memiliki beberapa atribut antara lain id, name dan address yang diset dengan access modifier private. Implementasi pada java menggunakan method getter-setter untuk mengakses masing-masing atribut tersebut. Implementasi yang berbeda dengan menggunakan bahasa Visual Basic .NET dapat dilihat pada Gambar 4 berikut ini.

```
Public Class Users

    Public Property UID As Integer
    Public Property Username As String
    Public Property Password As String
    Public Property Pangkat As Integer
    Public Property NIM1 As String
    Public Property Nama1 As String
    Public Property Telepon1 As String
    Public Property Email1 As String
    Public Property Kelas1 As String
    Public Property NIM2 As String
    Public Property Nama2 As String
    Public Property Telepon2 As String
    Public Property Email2 As String
    Public Property Kelas2 As String
    Public Property Periode As Periode
    Public Property Prodi As Prodi

End Class
```

Gambar 4. Entity Class pada Visual Basic .NET

Dapat dilihat pada Gambar 4, implementasi entity class sedikit berbeda antara Visual Basic .NET dan Java. Perbedaan yang paling mencolok adalah pada pendeklarasian atribut dan method-method yang terkandung didalamnya.

Pada Java, masing-masing atribut private diberikan *interface* berupa *method* getter-setter untuk mengakses atributnya. Dengan cara ini maka konsep enkapsulasi pada paradigma berbasis objek dapat dilaksanakan. Cara yang berbeda pada Visual Basic .NET yang menganut skema pada .NET Framework, menyediakan fitur *Property* yang secara otomatis melindungi atribut dengan akses private, dan sekaligus berfungsi sebagai *interface* getter dan setter yang setara dengan Java. Fitur *Property* pada .NET Framework tersedia sejak .NET Framework 2.0. [Cwalina & Abrams, 2005]

Fasilitas *Property* pada .NET Framework juga memungkinkan kita untuk mendeklarasikan sebuah atribut sebagai *read only, write only* ataupun *read-write*.

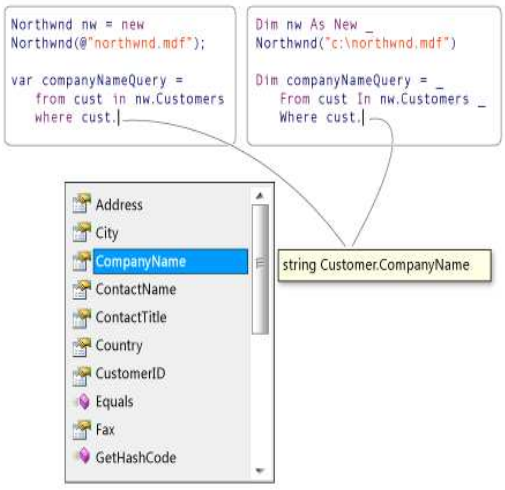
2.5 LINQ

Language-Integrated Query (LINQ) adalah sebuah inovasi yang diperkenalkan pada Visual Studio 2008 dan .NET

Framework versi 3.5 oleh Microsoft™ yang menjembatani gap antara paradigma berbasis objek dan relational database.

Sebelumnya, query terhadap data direpresentasikan sebagai sebuah karakter (string) biasa pada kode program tanpa adanya pengecekan tipe data atau dukungan dari *IntelliSense*. Denbgan kata lain, kita harus mempelajari dan menerapkan bahasa SQL yang berbeda untuk setiap jenis metode penyimpanan data (*persistence*) yang kita gunakan, antara lain: SQL database, dokumen XML, berbagai macam *Web Service*, dan sebagainya. [MSDN, 2016]

LINQ memungkinkan sebuah query menjadi bagian dari bahasa pemrograman pada Visual Basic .NET dan C#. Gambar 5 dibawah ini menunjukkan ilustrasi dari sebuah kode LINQ terhadap sebuah database pada SQL Server dalam bahasa C# dengan fitur *full type checking* dan IntelliSense.



Gambar 5. LINQ dengan *fulltype checking* dan IntelliSense pada Visual Basic .NET [MSDN, 2016]

Dengan menggunakan Visual Studio kita dapat menulis kode LINQ dengan bahasa Visual Basic .NET atau C# dengan database SQL Server, dokumen XML, ADO.NET Datasets dan berbagai macam *collection* objek yang mendukung IEnumerable atau tipe generik yang merupakan *interface* IEnumerable<T>.

Secara umum, tahapan dalam penggunaan LINQ antara lain:

1. Mendapatkan sumber data (*data source*).
2. Membuat query.
3. Mengeksekusi query.

Pada LINQ, eksekusi query berbeda dengan penciptaan query itu sendiri. Kita tidak mendapatkan data hanya dengan mengeksekusi query tanpa adanya sumber data. Gambar 6 dibawah ini menunjukkan contoh operasi query dalam tiga tahap secara umum. Pada contoh ini sumber data berasal dari array dengan tipe data integer. Namun pada dasarnya, sumber data ini bisa berasal dari selain array (database misalnya).

```
' Data source.
Dim numbers() As Integer = {0, 1, 2, 3, 4, 5, 6}

' Query creation.
Dim evensQuery = From num In numbers
                 Where num Mod 2 = 0
                 Select num

' Query execution.
For Each number In evensQuery
    Console.Write(number & " ")
Next
```

Gambar 6. Tahapan penggunaan LINQ pada Visual Basic .NET [MSDN, 2016]

Query tersebut diatas akan menghasilkan output pada console sebagai mana dapat dilihat pada Gambar 7 berikut ini:

```
Output:

0 2 4 6
```

Gambar 7. Hasil eksekusi LINQ pada sumber data array [Microsoft, 2008]

Secara umum query LINQ dieksekusi dengan metode *deferred execution*. Metode ini menciptakan query namun tidak secara langsung mengeksekusinya, akan tetapi query tersebut disimpan di sebuah variabel query, dalam contoh diatas variabel querynya adalah *evensQuery*. Query tersebut

dieksekusi ketika kita memanggil looping *For Each* yang mengembalikan nilai berupa urutan (*sequences*) nilai, dengan menerapkan operator standar query, seperti *Count* atau *Max*.

Metode eksekusi lain yang dapat digunakan adalah *immediate execution*. Pada metode ini, query dieksekusi bersamaan dengan pendefinisian. Eksekusi di-*trigger* saat kita memanggil method yang membutuhkan akses terhadap elemen-elemen yang ada pada hasil query. Beberapa contoh penerapan *immediate execution* dapat dilihat pada Gambar 8 berikut ini.

```
Dim numEvens = (From num In numbers
                Where num Mod 2 = 0
                Select num).Count()

Dim numEvensAgg = Aggregate num In numbers
                  Where num Mod 2 = 0
                  Select num
                  Into Count()
```

Gambar 8. Contoh *immediate execution* pada LINQ [MSDN, 2016]

Perbandingan antara metode eksekusi *deferred execution* dan *immediate execution* dapat dilihat pada Gambar 9 berikut ini.

```
' Immediate execution.
Dim evensList = (From num In numbers
                 Where num Mod 2 = 0
                 Select num).ToList()

' Deferred execution.
Dim evensQuery3 = From num In numbers
                  Where num Mod 2 = 0
                  Select num
' . . .
Dim evensArray = evensQuery3.ToArray()
```

Gambar 9. Perbandingan antara *deferred execution* dan *immediate execution* pada LINQ [MSDN, 2016]

2.6 T4 Text Templates

T4 Text Template adalah gabungan antara *text blocks* dan kontrol logika yang dapat menghasilkan file teks. Kontrol logika ditulis sebagai bagian dari kode program dalam bahasa C# atau Visual Basic .NET. File yang dihasilkan dapat berupa teks dengan berbagai macam bentuk, seperti jalaman web, file *resource*, atau kode program dalam berbagai macam bahasa pemrograman. [MSDN, 2016]

Terdapat dua macam T4 teks template, antara lain:

1. *Run Time* T4 teks template.
2. *Design-time* T4 teks template.

Pada *Run-time* T4 teks template, teks template diekskusi didalam aplikasi untuk menghasilkan file teks, biasanya berupa bagian dari outputnya. Sebagai contoh, pada Gambar 10 berikut ini terdapat sebuah template untuk mendefinisikan sebuah dokumen HTML.

```
<html><body>
 The date and time now is: <#= DateTime.Now #>
</body></html>
```

Gambar 10. Template HTML untuk T4 teks template

Dapat diperhatikan pada Gambar 10 diatas, bahwasanya template yang dibuat sudah mendekati output yang akan dihasilkan. Kesamaan template dengan output ini membantu kita untuk terhindar dari kesalahan ketika dibutuhkan perubahan pada template tersebut.

Untuk menghasilkan output, didalam aplikasi dapat kita berikan kode program yang berfungsi untuk men-*generate* kode yang sudah didefiniskan didalam template tersebut. Sebagai contoh dapat dilihat pada Gambar 11.

```
string webResponseText = new MyTemplate().TransformText();
```

Gambar 11. Generate template pada kode program

Pada metode run-time ini, tidak diperlukan Visual Studio untuk menghasilkan dokumen teks yang di-*generate*.

Pada *Design-time* T4 teks template, teks template dieksekusi didalam Visual Studio dan merupakan bagian dari kode program dan *resource* dari aplikasi. Umumnya kita

dapat menggunakan beberapa template yang membaca data didalam sebuah file atau database dan kemudian men-*generate* beberapa kode program berdasarkan database yang dibaca. Setiap template menghasilkan satu file dan dapat dieksekusi dengan menggunakan Visual Studio atau MSBuild.

Contoh pada Gambar 12 dibawah ini mengambil input data dari sebuah XML. Kapanpun kita mengedit file XML tersebut, maka engine T4 akan meng-*generate* ulang bagian dari kode program.

```
<#@ output extension=".txt" #>
<#@ assembly name="System.Xml" #>
<#
 System.Xml.XmlDocument configurationData = ...; // Read a data file here.
#>
namespace Fabrikam.<#= configurationData.SelectSingleNode("jobName").Value #>
{
  ... // More code here.
}
```

Gambar 12. Generate template dengan *design-time* pada kode program

Hasil eksekusinya akan menghasilkan file teks kurang lebih seperti yang dapat dilihat di Gambar 13.

```
namespace Fabrikam.FirstJob
{
  ... // More code here.
}
```

Gambar 13. Hasil teks yang di-*generate* menggunakan *design-time*

Design-time template memungkinkan kita untuk menghasilkan kode-generation dengan lebih cepat dan reliabel terutama saat dibutuhkan pengubahan pada konfigurasi saat terdapat kebutuhan yang berbeda. *Design-time* template juga merupakan tool yang sangat berguna pada proses *Agile Development.* [MSDN, 2016]

3. METODE PENELITIAN

Persiapan yang dibutuhkan untuk penelitian ini adalah IDE Visual Studio 2013. Didalam VS 2013 ini sudah termasuk diantaranya tool LINQ dan T4 teks template untuk mendukung terlaksananya penelitian ini. Ruang lingkup yang dicakup penelitian ini antara lain:

1. Database engine yang digunakan adalah Microsoft SQL Server.
2. Bahasa pemrograman Visual Basic .NET
3. Tool yang digunakan adalah IDE (Integrated Development Environment) Visual Studio 2013.
4. Entity class sudah dibuat sebelumnya, sesuai dengan struktur tabel yang ada.

Tahapan yang perlu dilaksanakan antara lain:

1. Mendapatkan sumber data, yaitu berupa database dengan sample dua atau lebih tabel yang saling berelasi.
2. Membuat template menggunakan T4 teks templating untuk meng-*generate* kode program dengan menggabungkan LINQ sebagai bagian dari template
3. Menguji file kode program yang dihasilkan.

3.1 Pembuatan T4 Teks Template

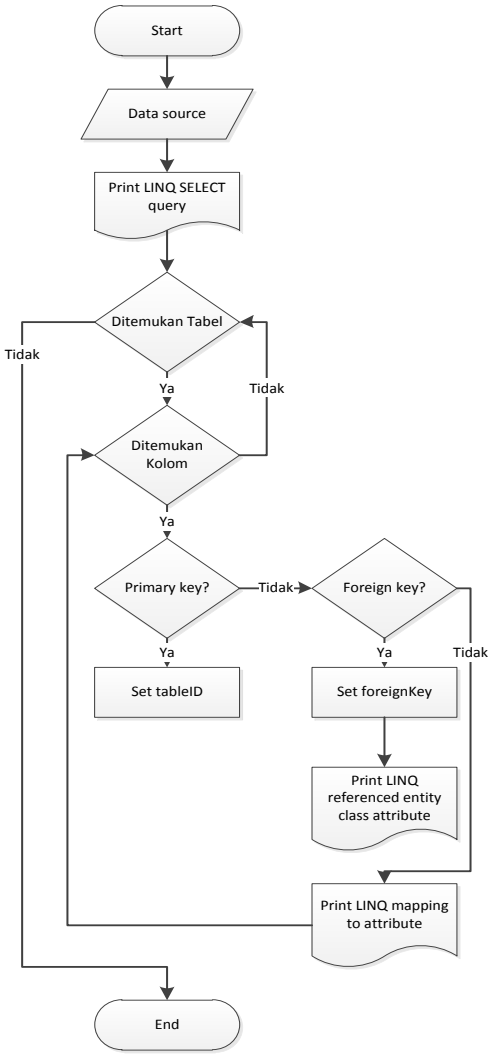
Teks template dibuat dengan metode *design-time T4 template* dan disesuaikan dengan database yang sudah disiapkan. Diasumsikan Entity class sudah dibuat dengan struktur sesuai dengan tabel database yang akan di-mapping. Secara garis besar, tahapan utama yang dikerjakan dalam teks template ini adalah:

1. Menggenerate nama class Data Access Object (DAO) sesuai dengan nama table yang dalam hal ini sama dengan nama *class entity*.
2. Membaca tabel-tabel yang ada pada database yang kemudian disimpan kedalam *DataTable*
3. Setiap tabel yang ditemukan digenerate kode LINQ untuk menyeleksi row pada *DataTable*
4. Mencari *primary key* dan *foreign key* nya.

5. Membaca *field* atau kolom yang ada pada tabel.
6. Setiap kolom dilihat tipe datanya. Untuk penelitian ini, tipe data yang dapat ditangani adalah tipe data: *varchar, integer* dan *datetime*.
7. Setiap kolom yang ditemukan digenerate kode LINQ untuk melakukan mapping terhadap atribut pada Entity class yang telah dibuat sebelumnya.
8. Menggenerate method-method yang berfungsi sebagai CRUD yang berisi kode-kode SQL atau disesuaikan dengan database

Lebih lengkap tentang pembuatan teks template ini dapat dilihat pada flowchart pada Gambar 14.

Pada penelitian ini, dibuat template yang nantinya menghasilkan output berupa method yang menerima parameter dengan tipe *DataTable* sebagai *data source*. Didalam method ini terdapat generator teks untuk meng-generate LINQ query yang nantinya akan melakukan mapping dari kolom-kolom tabel kedalam atribut pada *entity class*. Adapun anatomi dari method ini dapat dilihat pada Gambar 15.



Gambar 14. Algoritma *mapping generator*

Nama Method	**MapToList**
Parameter	DataTable dt
Return type	List of <Entity Type>
Header:	
`Private Shared Function MapToList(ByVal dt As DataTable) As List Of <Entity Type>`	

Gambar 15. Anatomi method yang dihasilkan oleh teks template

Dapat dilihat pada Gambar 14 bahwa return type daripada method MapToList adalah *type* dari entity class yang akan dilakukan mapping. Oleh karena itu perlu disepakati bahwa nama table pada database harus sama dengan nama type pada entity class.

Anatomi method-method lain yang berfungsi sebagai CRUD dapat dilihat pada Gambar 16.

Pseudo code pada Gambar 16 menunjukkan tahapan algoritma teks template untuk meng-generate method MapToList yang berisi query LINQ untuk melakukan mapping dari tabel database ke entity class.

Pada saat template generator mendeteksi Foreign Key, maka akan dilakukan pemanggilan method *selectById()* dengan parameter ID foreign key, yang kemudian akan mendapatkan return value berupa object class entity sesuai dengan relasinya pada tabel lain yang kemudian dapat dilakukan mapping terhadap atribut class entity.

```
Set datasource <- DataTable
Print(LINQ SELECT Query)
Look for tables, if table found
    Look for column, if column found
        Is Primary Key?
            Set tableID <- Primary Key
        End If
        Is Foreign Key?
            Set foreignKey <- Foreign Key
            Print LINQ referenced entity
            class attribute
        Else
            Print LINQ mapping to attribute
        End If
    End Loop
End Loop
```

Gambar 16. Pseudo code teks template generator

Nama Method	**MapToObject**
Parameter	DataTable dt
Return type	Entity Type
Header:	
`Private Shared Function MapToObject(ByVal dt As DataTable) As <Entity Type>`	

Nama Method	**Insert**
Parameter	Entity Type
Return type	Integer (generated database ID)
Header:	
`Public Shared Function insert(ByVal objModel As <Entity Type>) As Integer`	

Nama Method	**selectAll**
Parameter	Entity Type (Optional)
Return type	List of <Entity Type>
Header:	
`Public Shared Function selectAll(Optional ByVal objModel <Entity Type> = Nothing) As List Of <Entity Type>`	

Nama Method	**selectById**
Parameter	String id
Return type	Entity Type
Header:	
`Public Shared Function selectById(ByVal id As String) As <Entity Type>`	

Nama Method	**update**
Parameter	Entity Type
Return type	Boolean
Header:	
`Public Shared Function update(ByVal objModel As <Entity Type>) As Boolean`	

Nama Method	**delete**
Parameter	Entity Type
Return type	Boolean
Header:	
`Public Shared Function delete(ByVal objModel As <Entity Type>) As Boolean`	

Tabel 1. Anatomi method

3.2 Association Problem

Permasalahan *Association Problem* yang ada para Object Relation Mapping yang berkaitan dengan relasi tabel pada database ditangani oleh LINQ dengan cara memanggil method *selectById()*. Method ini mengembalikan nilai berupa Entity Type yang kemudian oleh LINQ dilakukan mapping terhadap Entity Class nya. Mapping dilakukan ke atribut dengan tipe Entity yang sama. Semua proses mapping ini digenerate secara otomatis oleh *T4 text templating*.

Contoh hasil generate LINQ untuk memetakan asosiasi type dapat dilihat pada Gambar 17.

```
Private Function MapToList(ByVal dt As DataTable)
    As List(Of Riset)

    Dim list As New List(Of Riset)

    list = (From row As DataRow In dt.Rows
          Select New Riset With {
        .RSID = row("RSID").ToString _
        , .Kategori = row("Kategori").ToString _
        , .Judul = row("Judul").ToString _
        , .Abstrak = row("Abstrak").ToString _
        , .Filename = row("Filename").ToString _
        , .TglUpload = row("TglUpload").ToString _
        , .Keterangan = row("Keterangan").ToString _
        , .Terkunci = row("Terkunci").ToString _
        , .Periode = _
          New PeriodeDAO().selectById(row("PRID").ToString) _
        , .Prodi = _
          New ProdiDAO().selectById(row("PROID").ToString) _
        , .Users = _
          New UsersDAO().selectById(row("UID").ToString) _
        }).ToList

    Return list
End Function
```

Gambar 17. Hasil generate teks untuk mapping asosiasi pada LINQ

4. HASIL DAN PEMBAHASAN

4.1 Data Source

Sebelum dilaksanakan pengujian terhadap text template generator yang sudah dibuat, perlu disiapkan data source dalam hal ini adalah database pada SQL Server. Pada pengujian ini kami menggunakan tiga buah tabel yang saling berelasi. Tujuan akhir output dari pengujian ini adalah terciptanya sebuah class Data Access Object (DAO) yang digenerate oleh text template yang didalamnya sudah termasuk mapping objek menggunakan LINQ dan juga method-method pendukungnya termasuk juga method-method untuk melakukan operasi dasar database atau CRUD.

Skema tabel berikut ini diambil dari rancangan database Rancang Bangun Sistem Tracking Control Laporan Akhir pada studi kasus Politeknik Negeri Malang. [Rismanto, 2015]

Tabel yang digunakan adalah tiga tabel yang berelasi. Untuk pengujian ini tidak perlu semua tabel di-generate karena fokus pengujian adalah pada penanganan relasi tabel.

1. Tabel Riset

Field	Tipe Data
RSID <pk>	int
PROID <fk>	int
PRID <fk>	int
UID <fk>	int
Kategori	int
Judul	text
Abstrak	text
TglAdd	datetime
Keterangan	varchar(50)
Terkunci	int
NOLA	varchar(10)

2. Tabel Sidang

Field	Tipe Data
SDID <pk>	int
UID <fk>	int
Nama	varchar(50)
Keterangan	text
TglStart	datetime
TglEnd	datetime
TglAdd	datetime
Terkunci	int
PROID <fk>	int
PRID <fk>	int

3. Tabel SidangDetail

Field	Tipe Data
SDTID <pk>	int
SDID <fk>	int
RSID <fk>	int
TglAdd	datetime
Nilai	char(2)
Lulus	int

Dapat dilihat dari struktur tabel SidangDetail terdapat relasi dengan dua tabel lainnya yaitu tabel Riset dan Sidang. Hal ini dibuktikan dengan adanya *foreign key* SDID dari tabel Sidang dan RSID dari tabel Riset.

4.2 Entity Class

ORM melakukan mapping dari relational database ke class entity yang ada pada desain aplikasi. Oleh karena itu perlu disiapkan class entity-nya yang mewakili entity pada sistem dan juga mencerminkan tabel pada database. Terdapat tiga class entity yang perlu disiapkan antara lain:

1. Class Riset

```
Public Class Riset

    Public Property RSID As Integer
    Public Property Kategori As Integer
    Public Property Judul As String
    Public Property Abstrak As String
    Public Property Filename As String
    Public Property TglUpload As Date
    Public Property Keterangan As String
    Public Property Terkunci As Integer
    Public Property Periode As Periode
    Public Property Prodi As Prodi
    Public Property Users As Users

End Class
```

2. Class Sidang

```
Public Class Sidang

    Public Property SDID As Integer
    Public Property Nama As String
    Public Property Keterangan As String
    Public Property TglStart As Date
    Public Property TglEnd As Date
    Public Property TglAdd As Date
    Public Property Terkunci As Integer
    Public Property Users As Users

End Class
```

3. Class SidangDetail

```
Public Class SidangDetail

    Public Property SDTID As Integer
    Public Property TglAdd As Date
    Public Property Nilai As String
    Public Property Lulus As Integer
    Public Property Riset As Riset
    Public Property Sidang As Sidang

End Class
```

4.3 Generate Code

Setelah di-generate kode dengan menggunakan teks templating, didapat method *MapToList* yang didalamnya terdapat kode LINQ untuk proses mapping kedalam entity class. Dapat dilihat pada Gambar 18.

```
Private Function MapToList(ByVal dt As DataTable) _
    As List(Of SidangDetail)

    Dim list As New List(Of SidangDetail)

    list = (From row As DataRow In dt.Rows
            Select New SidangDetail With {
           .SDTID = row("SDTID").ToString _
           , .TglAdd = row("TglAdd").ToString _
           , .Nilai = row("Nilai").ToString _
           , .Lulus = row("Lulus").ToString _
           , .Riset = _
              New RisetDAO().selectById(row("RSID").ToString) _
           , .Sidang = _
              New SidangDAO().selectById(row("SDID").ToString) _
            }).ToList

    Return list
End Function
```

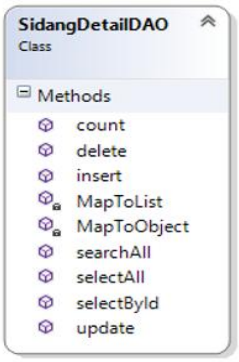
Gambar 18. Hasil generate teks templating untuk entity SidangDetail

Method pendukung untuk *MapToList()* ini adalah method *MapToObject()*. Method ini sebenarnya memanggil *MapToList()* hanya saja diambil indeks ke-0 agar return type-nya berupa *single instance type*. Berikut isi dari method *MapToObject()* pada Gambar 19.

```
Private Function MapToObject(ByVal dt As DataTable) _
    As SidangDetail

    Dim list As New List(Of SidangDetail)

    list = MapToList(dt)

    If list.Count > 0 Then
        Return list(0)
    Else
        Return Nothing
    End If
End Function
```

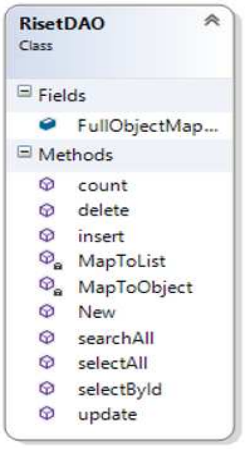
Gambar 19. Method *MapToObject()*

Lebih lengkapnya, class diagram untuk entity SidangDetail ini dapat dilihat pada Gambar 20. Class diagram ini digenerate dengan tool Diagram pada Visual Studio 2013 berdasarkan hasil generate *T4 text templating* yang sudah dipersiapkan sebelumnya.

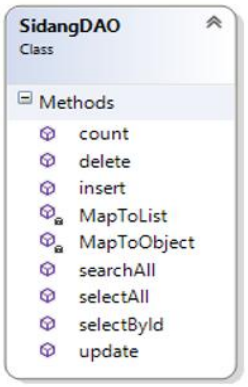


Gambar 20. Class diagram entity SidangDetail

Class diagram untuk entity Riset dan Sidang dapat dilihat pada Gambar 21 dan 22 berikut ini.



Gambar 21. Class diagram entity Riset



Gambar 22. Class diagram entity Sidang

5. KESIMPULAN

Penelitian ini telah berhasil mendeteksi asosiasi yang terdapat pada relational database dan melakukan mapping terhadap entity class yang ada.

Input dari penelitian ini adalah data source yaitu relational database dan entity class.

Output dari penelitian ini sesuai dengan yang diharapkan yaitu class Data Access Object (DAO) yang didalamnya terdapat query LINQ untuk melakukan relational mapping.

Dengan demikian layer *Persistence* [Bauer, C. & King, G., 2007] telah dapat di-*generate* secara otomatis. Layer ini kemudian dapat diadopsi kedalam layer *Business* dan *Presentation*. Dalam penelitian ini *object persistence* masih diwujudkan dalam penyimpanan data kedalam *relational database*. Pada prakteknya, *persistence* dapat berupa apapun, misalnya dokumen XML atau stand-alone RDBMS seperti SQLite.

Beberapa aspek yang dapat dikembangkan untuk penelitian selanjutnya antara lain:

1. Menghandle tipe data lain selain *varchar, integer* dan *datetime*. Fokus utama penelitian ini adalah pada deteksi asosiasi dan relational mapping. Oleh karena itu tipe data atribut yang bisa diproses sementara belum semua. Penelitian selanjutnya dapat mengembangkan topik ini agar dapat menghandle tipe data lain seperti BLOB.
2. Pembuatan layer Web Service. Desain SOA (*Service Oriented Apllication*) semakin banyak dikembangkan. Desain pattern pada penelitian ini sudah sangat cocok untuk diadopsi kedalam ekosistem SOA dengan menyediakan *interface* Web Service atau JSON yang dapat dikonsumsi oleh aplikasi lain.
3. Mapping yang dilakukan baru satu arah, yaitu dari relational database ke entity class. Dapat dikembangkan mapping dua arah, yaitu dari entity class ke relational database.
4. Generate *entity class* menggunakan T4 text templating berdasarkan tabel-tabel yang ada pada relational database, atau sebaliknya.

6. REFERENSI

Atzeni, P., Jensen, C.S., Orsi, G. *The Relational Model Is Dead, SLQ Is Dead, And I Don't Feel So Good Myself*. SIGMOD Rec. ACM, 2013.

Atkninson, M. P., Buneman, O. P. *Types And Persistence In Database Programming Languages*. ACM Computing Surveys, ACM. 1987.

Bauer, C., King, G. *Java Persistence with Hibernate*. Manning Publication Co. 2007.

Lipitsainen, Avro. *ORM – Object Relational Mapping*. DBTechNet. 2011.

Rismanto, Ridwan. *XML Web Service Sebagai Middleware Pada Existing Service*.

Jurnal Teknologi Informasi. STMIK PPKIA Malang. 2015

Cwalina, K., Abrams, B. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries, 2nd Edition.* Addison-Wesley Professional; 2nd Edition. 2008.

Mark L. Fussell. *Foundations of Object-Relational Mapping*. White Paper, ChiMu Corp, 1997.

MSDN, Microsoft. *Code Generation and T4 Text Templates.* Dikutip pada 28 April 2016. https://msdn.microsoft.com/en-us/library/bb126445.aspx

MSDN, Microsoft. *Introduction to LINQ*. Dikutip pada 28 April 2016. https://msdn.microsoft.com/en-us/library/bb397897.aspx

MSDN, Microsoft. *Writing Your First LINQ Query (Visual Basic).* Dikutip pada 28 April 2016. https://msdn.microsoft.com/en-us/library/bb384667.aspx