

O-R Mapping Hibernate

Pemetaan Object – Relational
dengan
Hibernate

Endy Muhardin <endy@artivisi.com>

Materi

- Pengenalan Hibernate
- Instalasi dan Konfigurasi
- Persistence Class
- Integrasi XDoclet
- One to One Relationship
- One to Many Relationship
- Parent – Child Relationship
- Inheritance
- CRUD
- Advanced Query

Hibernate

- Hibernate adalah perangkat untuk memudahkan akses database
- Hibernate menangani koneksi database dari aplikasi Java ke Database Server
- Ada dua pendekatan penggunaan Hibernate:
 - Struktur Database -> Object Model
 - Object Model -> Struktur Database

Fitur Hibernate

- Pemilihan tipe data otomatis
- Mendukung banyak database populer
- Mapping Java Object menjadi Struktur Tabel Database
- Connection Pooling
- Transaction Management
- Memudahkan mapping one-to-many dan many-to-many

Asumsi yang salah tentang Hibernate

- Programmer tidak perlu menguasai konsep SQL query
- Programmer tidak perlu mengetahui konsep JDBC

Instalasi Hibernate

- Download hibernate di <http://www.hibernate.org>
- Extract di folder instalasi
- Copy hibernate.jar kedalam CLASSPATH
- Buat file konfigurasi hibernate.properties
- Buat file mapping Java - Tabel

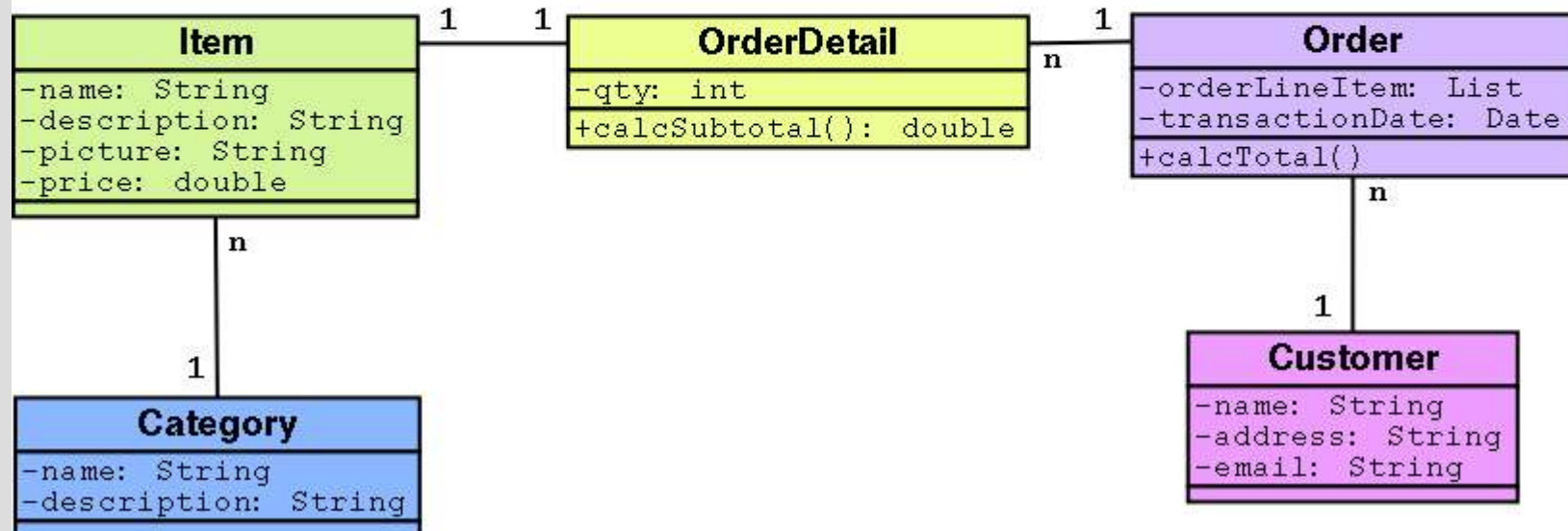
hibernate.properties

```
hibernate.connection.driver_class = com.mysql.jdbc.Driver
hibernate.connection.url = jdbc:mysql://localhost:3306/projects
hibernate.connection.username = project
hibernate.connection.password = project
hibernate.dialect = net.sf.hibernate.dialect.MySQLDialect
```

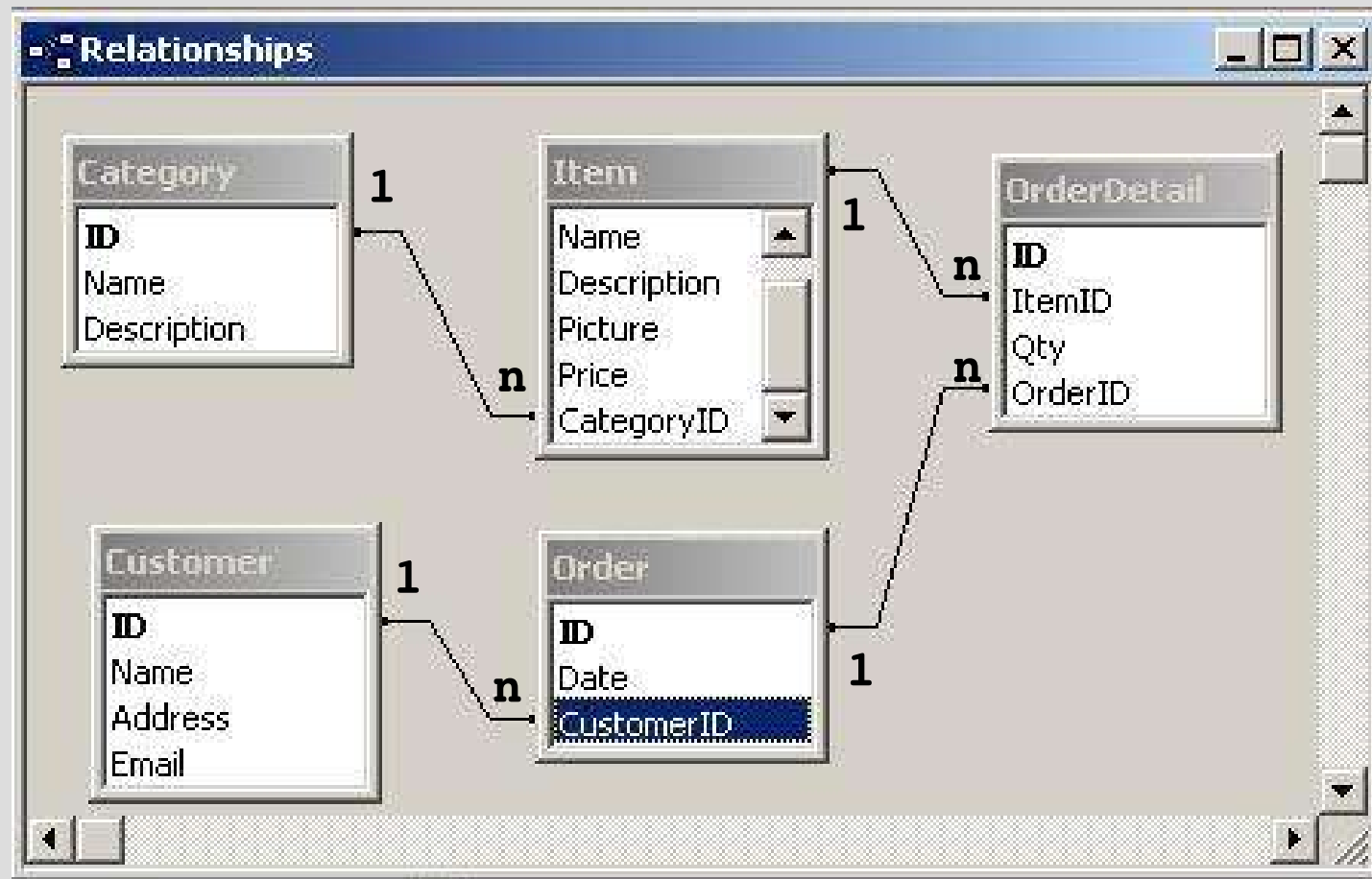
- JDBC Driver yang digunakan untuk koneksi
- Lokasi koneksi database: komputer **localhost**, port **3306**, ke database bernama **projects**
- Username yang digunakan adalah : project
- Password yang digunakan adalah : project
- SQL Query didesain untuk berjalan di MySQL

Studi Kasus Online Shopping

Class Diagram
Online Store - Business Model
Author : Endy Muhardin
ver : 1.0.0 - 110204



Struktur Tabel



Category.java

```
public class Category
{
    private Long id;
    private String name;
    private String description;

    public Long getId()
    {
        return this.id ;
    }

    private void setId(Long id)
    {
        this.id = id;
    }
}
```

Hibernate Mapping *.hbm.xml

```
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

<hibernate-mapping>
    <class
        name="Category"
        table="Category"
        dynamic-update="false"
        dynamic-insert="false"
    >
        <!-- definisi class ID / Primary Key di sini -->
        <!-- definisi class property di sini -->
    </class>

</hibernate-mapping>
```

Category.hbm.xml

ID / Primary Key

```
<id name="id"  
    column="CategoryId"  
    type="java.lang.Long">  
  
    <generator class="native">  
    </generator>  
  
</id>
```

Category.hbm.xml Properties

```
<property
  name="name"
  type="java.lang.String"
  update="true"
  insert="true"
  column="Name"
/>
<property
  name="description"
  type="java.lang.String"
  update="true"
  insert="true"
  column="Description"
/>
```

Auto Generate Table

- Hibernate dapat membuat tabel otomatis di database, dengan menggunakan file *.hbm.xml
- Pembuatan tabel dilakukan dengan menggunakan class SchemaExport

DataManager.java

```
import net.sf.hibernate.*;
import net.sf.hibernate.cfg.*;
import net.sf.hibernate.tool.hbm2ddl.*;

public class DataManager
{
    public static void main(String[] args)
    {
        try {

            Configuration c = new Configuration().addClass(Category.class);
            new SchemaExport(c).createTable(true, true);

        } catch (HibernateException err) {
            err.printStackTrace();
        } catch (MappingException err) {
            err.printStackTrace();
        }
    }
}
```

CategoryDAO.java

Constructor

```
import net.sf.hibernate.*;
import net.sf.hibernate.cfg.*;
import net.sf.hibernate.tool.hbm2ddl.*;

public class CategoryDAO
{
    SessionFactory sf = null;
    public CategoryDAO()
    {
        try {

            Configuration c = new Configuration().addClass(Category.class);
            sf = c.buildSessionFactory();

        } catch (HibernateException err) {
            err.printStackTrace();
        } catch (MappingException err) {
            err.printStackTrace();
        }
    }
}
```

CategoryDAO.java

INSERT

```
import net.sf.hibernate.*;
import net.sf.hibernate.cfg.*;
import net.sf.hibernate.tool.hbm2ddl.*;

public class CategoryDAO
{
    public void create(Category c)
    {
        try {

            Session s = sf.openSession();
            s.save(c);
            s.flush();
            s.close();

        } catch (HibernateException err) {
            err.printStackTrace();
        } catch (MappingException err) {
            err.printStackTrace();
        }
    }
}
```

CategoryDAO.java

SEARCH

```
public List search(String attr, String value)
{
    List result = null;
    try {

        String q = "from Category c where c."+attr+" like %?%";
        Session s = sf.openSession();
        result = s.find(q, Hibernate.STRING, value);

    } catch (HibernateException err) {
        err.printStackTrace();
    } catch (MappingException err) {
        err.printStackTrace();
    }
    return result;
}
```

CategoryDAO.java

LOAD

```
public Category search(Long id)
{
    Category result = null;
    try {

        Session s = sf.openSession();
        result = (Category)s.load(Category.class, id);

    } catch (HibernateException err) {
        err.printStackTrace();
    } catch (MappingException err) {
        err.printStackTrace();
    }
    return result;
}
```

CategoryDAO.java

UPDATE

```
public void update(Category c)
{
    try {

        Session s = sf.openSession();
        s.update(c);
        s.flush();
        s.close();

    } catch (HibernateException err) {
        err.printStackTrace();
    } catch (MappingException err) {
        err.printStackTrace();
    }
}
```

CategoryDAO.java

DELETE

```
public void delete(Category c)
{
    try {

        String q = "from Category c where c."+attr+" like %?%";
        Session s = sf.openSession();
        result = s.find(q, Hibernate.STRING, value);

    } catch (HibernateException err) {
        err.printStackTrace();
    } catch (MappingException err) {
        err.printStackTrace();
    }
    return result;
}
```

Mapping One To Many

- Hubungan One to Many dijelaskan dengan contoh Item – Category
- Tiap Item memiliki satu category, sedangkan tiap category memiliki banyak Item
- Hubungan yang dibuat bersifat satu arah, artinya dari Item kita dapat mengetahui Category, tapi tidak sebaliknya

Item.java

```
public class Item
{
    private Long id;
    private String name;
    private String description;
    private String picture;
    private double price;
    private Category cat;

    // .. getter – setter method yang lainnya

    public Category getCategory()
    { return cat; }

    public void setCategory(Category c)
    {
        cat = c;
    }
}
```

Item.hbm.xml

```
<?xml version="1.0"?>
<!-- DTD Definition -->

<hibernate-mapping>
  <class name="com.artivisi.library.Category"
        table="library_Categories"
        dynamic-update="false"
        dynamic-insert="false"
  >
    <!-- definisi class ID / Primary Key di sini -->
    <!-- definisi class property di sini -->

    <many-to-one name="category"
                 column="Category_ID"
                 not-null="true"
    />

  </class>

</hibernate-mapping>
```

Auto Generate Table

```
import net.sf.hibernate.*;
import net.sf.hibernate.cfg.*;
import net.sf.hibernate.tool.hbm2ddl.*;

public class DataManager
{
    public static void main(String[] args)
    {
        try {

            Configuration c = new Configuration().addClass(Category.class)
                            .addClass(Item.class);

            new SchemaExport(c).createTable(true, true);

        } catch (HibernateException err) {
            err.printStackTrace();
        } catch (MappingException err) {
            err.printStackTrace();
        }
    }
}
```

Bidirectional One to Many

- Hubungan one to many, pihak one dapat 'melihat' pihak many, dan sebaliknya.
- Misalnya : Order-Customer
- Order memiliki method:
 - `public Customer getCustomer();`
- Customer juga memiliki method :
 - `public Set getOrders();`

Mapping Customer - Order

- Di sisi Order, sama dengan sisi Item pada Item-Category
- Di sisi Customer, ditambahkan satu collection bertipe Set
- Set diisi dengan referensi one-to-many ke class Order

Customer.hbm.xml

```
<class name="Customer" table="Customer">
  <id name="id">
    <generator class="native"/>
  </id>
  <property name="name"/>
  <! -- property yang lainnya -->

  <map name="orders" inverse="true" lazy="true">
    <key column="CustomerId"/>
    <one-to-many class="Order"/>
  </map>
</class>
```

Order.hbm.xml

```
<class name="Order" table="Order">
  <id name="id">
    <generator class="native"/>
  </id>
  <property name="date"/>

  <many-to-one name="customer" column="customer_id"/>

  <list name="orderDetails" table="OrderDetail" lazy="true">
    <key column="OrderId"/>
    <index column="ID"/>

    <composite-element class="OrderDetail">
      <property name="qty"/>
      <many-to-one name="item" column="ItemId"/>
    </composite-element>

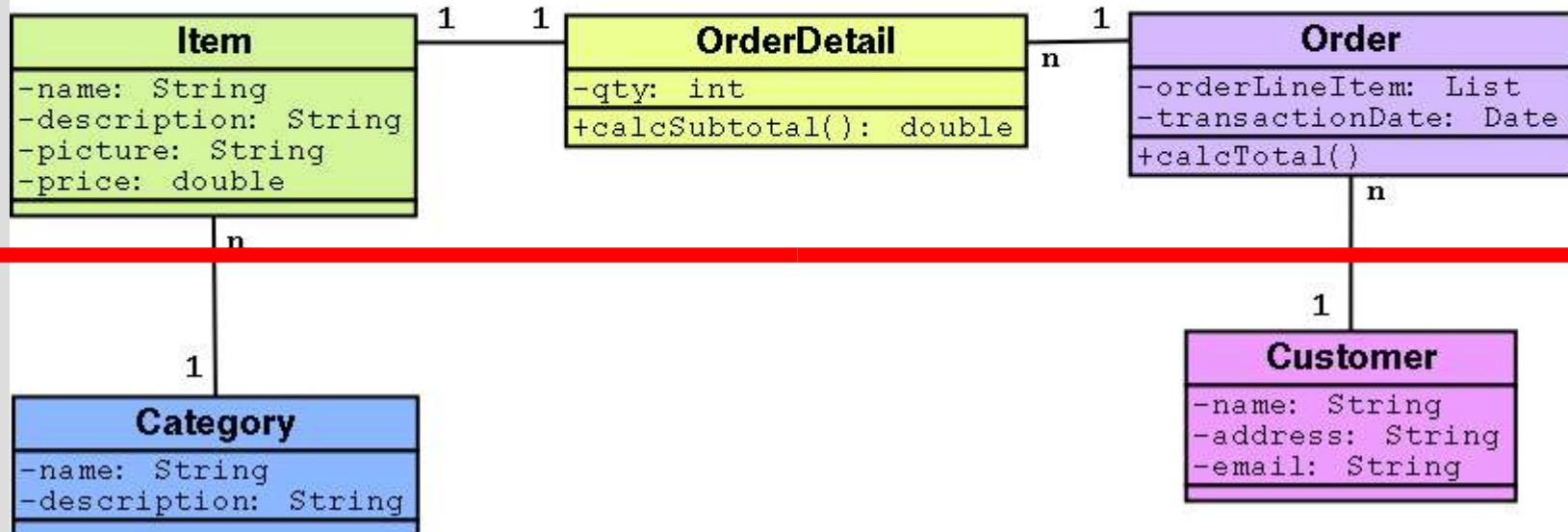
  </list>
</class>
```

Many to Many

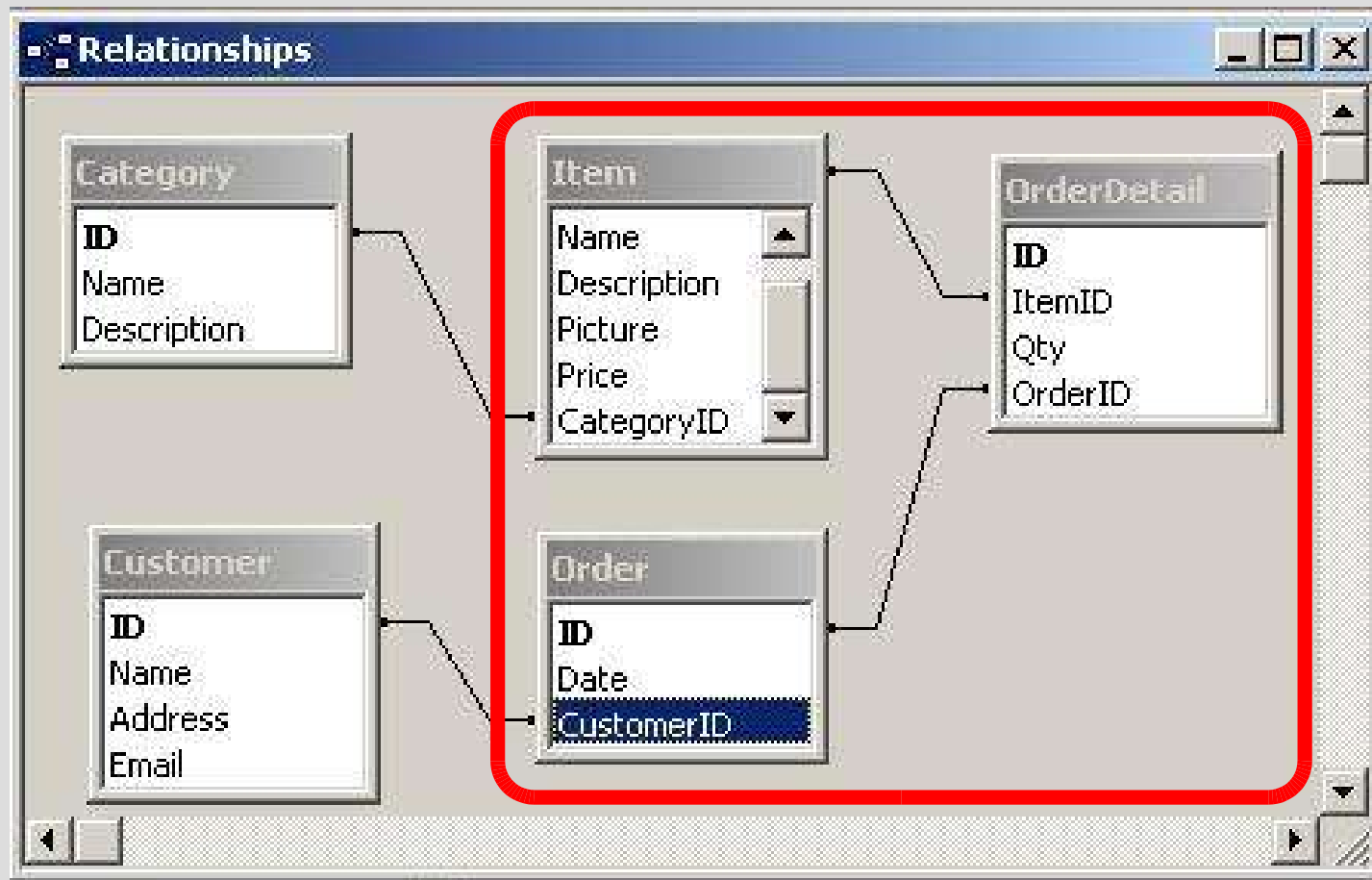
- Hubungan many to many dicontohkan dengan hubungan Order – Item
- Dalam satu kali **Order**, pelanggan memesan banyak **Item** sekaligus
- Satu jenis **Item**, dipesan berkali-kali dalam banyak **Order**
- Hubungan ini direpresentasikan menggunakan class OrderDetail

Object Model Many to Many

Class Diagram
Online Store - Business Model
Author : Endy Muhardin
ver : 1.0.0 - 110204



Struktur Tabel Many to Many



Order.java

```
public class Order
{
    private Long id;
    private java.util.Date trxDate;
    private java.util.List orderDetails;

    // .. getter – setter method yang lainnya

    public java.util.List getOrderDetails()
    { return orderDetails; }

    public void addOrderDetail(OrderDetail ord)
    {
        orderDetails = ord;
    }

    public void setOrderDetails(java.util.List ords)
    {
        orderDetails = ords;
    }
}
```

OrderDetail.java

```
public class OrderDetail
{
    private Long id;
    private Item _item;
    private int qty;

    // .. getter – setter method yang lainnya

    public double calcSubtotal()
    {
        return (_item.getPrice() * qty);
    }
}
```

Order.hbm.xml

```
<class name="Order" table="Order">
  <id name="id">
    <generator class="native"/>
  </id>
  <property name="date"/>

  <many-to-one name="customer" column="customer_id"/>

  <list name="orderDetails" table="OrderDetail" lazy="true">
    <key column="OrderId"/>
    <index column="ID"/>

    <composite-element class="OrderDetail">
      <property name="qty"/>
      <many-to-one name="item" column="ItemId"/>
    </composite-element>

  </list>
</class>
```

Table Mapping Many to Many

