

# MODUL I

## *Persistence* dan Hibernate

### 1.1 Tujuan

1. Mahasiswa memahami pengertian *persistence* dan *persistent data*
2. Mahasiswa memahami berbagai solusi untuk *persistence*
3. Mahasiswa memahami keterkaitan antar komponen dalam solusi *persistence*.
4. Mahasiswa memahami dan mampu menggunakan JDBC untuk solusi *persistence*
5. Mahasiswa memahami dan mampu memulai suatu proyek yang menggunakan Hibernate tanpa JPA sebagai solusi *persistence*.
6. Mahasiswa memahami dan mampu menggunakan operasi dasar Hibernate tanpa JPA sebagai solusi suatu masalah *persistence*.

### 1.2 Teori Singkat

*Persistence* merupakan konsep yang sangat mendasar dalam membangun aplikasi. Pada dasarnya, *persistence* berkaitan dengan pengelolaan data dalam suatu aplikasi. Aplikasi yang melakukan pengolahan data harus menyimpan dan kemudian juga melakukan proses untuk me-*retrieve* data. Operasi tersebut dikenal dengan istilah *persistence* dan data yang terlibat dalam aplikasi tersebut dikenal dengan istilah *persistent data*.

Pada dasarnya, data dalam suatu aplikasi disimpan dalam suatu basis data relasional. Software basis data yang digunakan untuk keperluan tersebut pada dasarnya merupakan software basis data yang dikelola dengan perintah SQL (*Structured Query Language*). Beberapa contoh software tersebut diantaranya adalah MySQL, PostgreSQL, Oracle, HSQLDB, dan lain-lain.

Untuk mengakses *persistent data* yang tersimpan dalam basis data, diperlukan suatu API yang dikenal dengan istilah JDBC. Setiap software basis data mempunyai *driver* JDBC yang memungkinkan pemrogram untuk mengakses *persistent data* menggunakan Java. Dengan demikian, program

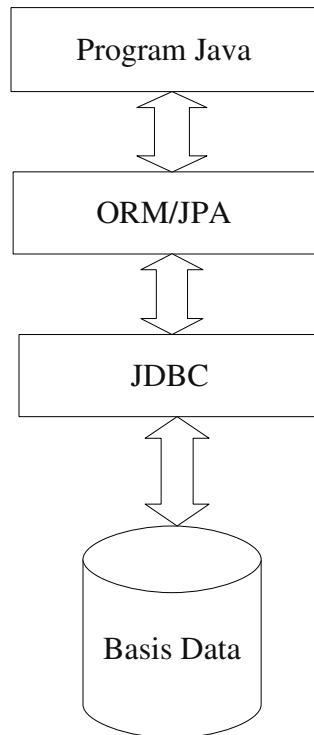
Java yang dibuat menggunakan query SQL yang kemudian diteruskan ke JDBC dan pada akhirnya akan diteruskan oleh *driver* JDBC ke basis data yang bersangkutan.

Penggunaan SQL dalam Java atau bahasa-bahasa pemrograman berorientasi obyek lainnya mendatangkan berbagai kemungkinan masalah. Berbagai masalah yang muncul tersebut dikenal dengan *impedance mismatch*. Ketidaksesuaian antara basisdata relasional yang berbasis pada paradigma matematis dengan PBO yang merupakan paradigma di tataran rekayasa peranti lunak serta ketidaksesuaian berbagai tipe data membuat pengelolaan data persistent ini menjadi hal yang kompleks.

ORM (*Object/Relational Mapping*) merupakan teknik pemrograman untuk memetakan tabel suatu basis data relasional ke suatu *class* serta memungkinkan untuk mengelola perbedaan sistem tipe pada basis data relasional dengan bahasa pemrograman berorientasi obyek. Untuk mengimplementasikan ORM ini, terdapat berbagai software dari berbagai vendor. Beberapa diantaranya adalah Hibernate, TopLink Essentials, EclipseLink, iBatis, dan lain-lain.

Banyaknya vendor yang memproduksi software ORM menyebabkan implementasi yang berbeda-beda sehingga menyulitkan para pemrogram. Kondisi tersebut menyebabkan JCP (*Java Community Process*) membuat standardisasi yang dimasukkan ke dalam EJB 3. Standar tersebut adalah JPA (*Java Persistence API*). Hibernate saat ini juga mengimplementasikan JSR 220 JPA. Dengan menggunakan JPA, terdapat keseragaman dalam mengatasi masalah *persistence* sehingga tidak terlalu menyulitkan pemrogram dalam mengembangkan aplikasi.

Secara konseptual, hubungan antara komponen-komponen untuk *persistence* ini adalah sebagai berikut:



### 1.3 Praktik

#### **Persiapan Basis Data MySQL**

Untuk praktik, siapkanlah basis data serta table dan isi sebagai berikut:

##### **Membuat basis data**

```
CREATE DATABASE mydb;
```

##### **Membuat table**

```
CREATE TABLE customers ( id int NOT NULL, name varchar(35) NOT NULL, address varchar(50) NOT NULL, PRIMARY KEY (`id`) )
```

##### **Mengisikan data**

```
insert into customers values (1, 'Zaky A. Aditya', 'Griya Purwa Asri');
insert into customers values (2, 'Bambang PDP', 'Gedongkuning');
```

#### **Persistence Menggunakan JDBC**

## Source Code (TestMyDB.java)

```
import java.sql.*;

public class TestMyDb {

    public static void main(String[] args) {

        System.out.println("Contoh akses ke MySQL menggunakan JDBC.");

        Connection conn = null;
        String url = "jdbc:mysql://localhost:3306/";
        String dbName = "mydb";
        String driver = "com.mysql.jdbc.Driver";
        String userName = "root";
        String password = "passwordmu";
        String query = "SELECT * FROM customers";

        try {
            Class.forName(driver).newInstance();

conn=DriverManager.getConnection(url+dbName,userName,password);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("Connected to the database");
            while (rs.next()) {
                Integer i = rs.getInt("ID");
                String n = rs.getString("name");
                String a = rs.getString("address");
                System.out.println("ID = " + i + ", Nama: " + n + ", Alamat: "
+ a);
            }
            conn.close();
            System.out.println("Disconnected from database");
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("gagal koneksi ke database");
        }
    }
}
```

## Kompilasi dan menjalankan program

Untuk mengkompilasi, letakkan / copy file mysql-connector-java-5.1.8-bin.jar ke *directory* yang sama dengan letak dari TestMyDb.java. Setelah itu, kerjakan langkah-langkah berikut:

```

$ export CLASSPATH=$CLASSPATH:./mysql-connector-java-5.1.8-
bin.jar
$ javac TestMyDb.java
$ java TestMyDb
Contoh akses ke MySQL menggunakan JDBC.
Connected to the database
ID = 1, Nama: Zaky A. Aditya, Alamat: Griya Purwa Asri
ID = 2, Nama: Bambang PDP, Alamat: Gedongkuning
Disconnected from database
$

```

### **Persistence menggunakan Hibernate**

Untuk praktik ini, semua file disimpan dibawah direktori **hellohibernate**. Susunan direktori dan file yang terdapat pada direktori **hellohibernate** adalah sebagai berikut:

```

hellohibernate/
|-- build.xml
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- freemarker.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   |-- slf4j-simple-1.5.8.jar
`-- src
    |-- hello
    |   |-- Customer.hbm.xml
    |   |-- Customer.java
    |   |-- HelloHibernate.java
    |-- hibernate.cfg.xml
    |-- log4j.properties

```

```
`-- persistence
  `-- HibernateUtil.java
```

Berikut adalah isi dari file-file tersebut:

### build.xml

```
<project name="HelloHibernate" default="compile" basedir=". ">
  <!-- Name of project and version -->
  <property name="proj.name"      value="HelloHibernate"/>
  <property name="proj.version"    value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir"    value="src"/>
  <property name="lib.dir"         value="lib"/>
  <property name="build.dir"       value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>
  <!-- Useful shortcuts -->
  <patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
  </patternset>
  <!-- Clean up -->
  <target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
  </target>
  <!-- Compile Java source -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac
      srcdir="${src.java.dir}"
      destdir="${build.dir}"
      nowarn="on"
      <classpath refid="project.classpath"/>
    </javac>
  </target>
  <!-- Copy metadata to build classpath -->
  <target name="copymetafiles">
    <copy todir="${build.dir}">
      <fileset dir="${src.java.dir}">
        <patternset refid="meta.files"/>
      </fileset>
    </copy>
  </target>
  <target name="run" depends="compile, copymetafiles"
    description="Build and run HelloHibernate">
    <java fork="true"
```

```

        classname="hello.HelloHibernate"
        classpathref="project.classpath">
        <classpath path="\${build.dir}"/>
    </java>
</target>
<taskdef name="hibernatetool"
    classname="org.hibernate.tool.ant.HibernateToolTask"
    classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
    description="Exports a generated schema to DB and file">
    <hibernatetool destdir="\${basedir}"/>
    <classpath path="\${build.dir}"/>
    <configuration
        configurationfile="\${build.dir}/hibernate.cfg.xml"/>
    <hbm2ddl
        drop="true"
        create="true"
        export="true"
        outputfilename="hellohibernate-ddl.sql"
        delimiter=";"
        format="true"/>
    </hibernatetool>
</target>
</project>

```

### hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/mydb
        </property>
        <property name="hibernate.connection.username">
            root
        </property>
        <property name="hibernate.connection.password">
            passwordmu
        </property>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQL5Dialect
        </property>
        <!-- Use the C3P0 connection pool provider -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>
        <!-- Show and print nice SQL on stdout -->
    </session-factory>
</hibernate-configuration>

```

```

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <!-- List of XML mapping files -->
    <mapping resource="hello/Customer.hbm.xml"/>
  </session-factory>
</hibernate-configuration>

```

## log4j.properties

```

# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE}
    %5p %c{1}:%L - %m%n
# Root logger option
log4j.rootLogger=INFO, stdout
# Hibernate logging options (INFO only shows startup messages)
log4j.logger.org.hibernate=INFO
# Log JDBC bind parameter runtime arguments
log4j.logger.org.hibernate.type=INFO

```

## Customer.hbm.xml

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class
    name="hello.Customer"
    table="CUSTOMERS">
    <id
      name="id"
      column="ID">
      <generator class="increment"/>
    </id>
    <property
      name="name"
      column="NAME"/>
    <property
      name="address"
      column="ADDRESS"/>
    <many-to-one
      name="nextCustomer"
      cascade="all"
      column="NEXT_CUSTOMER_ID"
      foreign-key="FK_NEXT_CUSTOMER"/>
  </class>
</hibernate-mapping>

```

## Customer.java

```
package hello;

public class Customer {
    private Long id;
    private String name;
    private String address;
    private Customer nextCustomer;

    Customer() {}
    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
    }
    public Long getId() {
        return id;
    }
    private void setId(Long 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 getNextCustomer() {
        return nextCustomer;
    }
    public void setNextCustomer(Customer nextCustomer) {
        this.nextCustomer = nextCustomer;
    }
}
```

## HelloHibernate.java

```
package hello;

import java.util.*;
import org.hibernate.*;
import persistence.*;

public class HelloHibernate {

    public static void main(String[] args) {
```

```

// First unit of work
Session session = HibernateUtil.getSessionFactory().openSession();
Transaction tx = session.beginTransaction();
Customer customer = new Customer("Linus Torvalds","Finlandia");
Long msgId = (Long) session.save(customer);
tx.commit();
session.close();

// Second unit of work
Session newSession =
HibernateUtil.getSessionFactory().openSession();
Transaction newTransaction = newSession.beginTransaction();
List customers = newSession.createQuery("from Customer c order by
c.name asc").list();
System.out.println( customers.size() + " customer(s) found:" );
for ( Iterator iter = customers.iterator();
iter.hasNext(); ) {
Customer loadedCust = (Customer) iter.next();
System.out.println( loadedCust.getName() );
System.out.println( loadedCust.getAddress() );
}
newTransaction.commit();
newSession.close();

// Shutting down the application
HibernateUtil.shutdown();

}
}

```

## HibernateUtil.java

```

package persistence;

import org.hibernate.*;
import org.hibernate.cfg.*;

public class HibernateUtil {
    private static SessionFactory sessionFactory;
    static {
        try {
            sessionFactory =
Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory() {
        // Alternatively, you could look up in JNDI here
        return sessionFactory;
    }
    public static void shutdown() {
        // Close caches and connection pools

```

```
    sessionFactory().close();
}
}
```

## Mengkompilasi

Untuk mengkompilasi, masuk ke direktori hellohibernate dan eksekusi ant seperti berikut ini:

```
[bpdp@bpdp-arch hellohibernate]$ ant
Buildfile: build.xml

clean:
    [delete] Deleting directory /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin
    [mkdir] Created dir: /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin

compile:
    [javac] Compiling 3 source files to /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin

BUILD SUCCESSFUL
Total time: 1 second
```

Untuk menjalankan hasil kompilasi, eksekusi ant dengan parameter run berikut ini:

```
[bpdp@bpdp-arch hellohibernate]$ ant run
Buildfile: build.xml

clean:
    [delete] Deleting directory /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin
    [mkdir] Created dir: /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin

compile:
    [javac] Compiling 3 source files to /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin

copymetafiles:
    [copy] Copying 3 files to /home/bpdp/kerjaan/modul-
praktikum/orm/bab1/hellohibernate/bin

run:
    [java] 30 [main] INFO org.hibernate.cfg.Environment - Hibernate
```

### 3.3.2.GA

```
[java] 33 [main] INFO org.hibernate.cfg.Environment -
hibernate.properties not found
[java] 37 [main] INFO org.hibernate.cfg.Environment - Bytecode
provider name : javassist
[java] 47 [main] INFO org.hibernate.cfg.Environment - using JDK
1.4 java.sql.Timestamp handling
[java] 159 [main] INFO org.hibernate.cfg.Configuration -
configuring from resource: /hibernate.cfg.xml
[java] 159 [main] INFO org.hibernate.cfg.Configuration -
Configuration resource: /hibernate.cfg.xml
[java] 275 [main] INFO org.hibernate.cfg.Configuration - Reading
mappings from resource : hello/Customer.hbm.xml
[java] 380 [main] INFO org.hibernate.cfg.HbmBinder - Mapping
class: hello.Customer -> CUSTOMERS
[java] 483 [main] INFO org.hibernate.cfg.Configuration -
Configured SessionFactory: null
[java] 493 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - Using
Hibernate built-in connection pool (not for production use!)
[java] 494 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - Hibernate
connection pool size: 20
[java] 494 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - autocommit
mode: false
[java] 500 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - using
driver: com.mysql.jdbc.Driver at URL: jdbc:mysql://localhost:3306/mydb
[java] 501 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - connection
properties: {user=root, password=****}
[java] 796 [main] INFO org.hibernate.cfg.SettingsFactory - RDBMS:
MySQL, version: 5.1.38-log
[java] 797 [main] INFO org.hibernate.cfg.SettingsFactory - JDBC
driver: MySQL-AB JDBC Driver, version: mysql-connector-java-5.1.8
( Revision: ${svn.Revision} )
[java] 837 [main] INFO org.hibernate.dialect.Dialect - Using
dialect: org.hibernate.dialect.MySQL5Dialect
[java] 845 [main] INFO
org.hibernate.transaction.TransactionFactoryFactory - Using default
transaction strategy (direct JDBC transactions)
[java] 847 [main] INFO
org.hibernate.transaction.TransactionManagerLookupFactory - No
TransactionManagerLookup configured (in JTA environment, use of read-
write or transactional second-level cache is not recommended)
[java] 847 [main] INFO org.hibernate.cfg.SettingsFactory -
Automatic flush during beforeCompletion(): disabled
[java] 847 [main] INFO org.hibernate.cfg.SettingsFactory -
Automatic session close at end of transaction: disabled
[java] 848 [main] INFO org.hibernate.cfg.SettingsFactory - JDBC
batch size: 15
[java] 848 [main] INFO org.hibernate.cfg.SettingsFactory - JDBC
batch updates for versioned data: disabled
```

```

    [java] 848 [main] INFO org.hibernate.cfg.SettingsFactory -
Scrollable result sets: enabled
    [java] 849 [main] INFO org.hibernate.cfg.SettingsFactory - JDBC3
getGeneratedKeys(): enabled
    [java] 849 [main] INFO org.hibernate.cfg.SettingsFactory -
Connection release mode: auto
    [java] 850 [main] INFO org.hibernate.cfg.SettingsFactory - Maximum
outer join fetch depth: 2
    [java] 851 [main] INFO org.hibernate.cfg.SettingsFactory - Default
batch fetch size: 1
    [java] 851 [main] INFO org.hibernate.cfg.SettingsFactory -
Generate SQL with comments: disabled
    [java] 851 [main] INFO org.hibernate.cfg.SettingsFactory - Order
SQL updates by primary key: disabled
    [java] 851 [main] INFO org.hibernate.cfg.SettingsFactory - Order
SQL inserts for batching: disabled
    [java] 851 [main] INFO org.hibernate.cfg.SettingsFactory - Query
translator: org.hibernate.hql.ast.ASTQueryTranslatorFactory
    [java] 854 [main] INFO
org.hibernate.hql.ast.ASTQueryTranslatorFactory - Using
ASTQueryTranslatorFactory
    [java] 854 [main] INFO org.hibernate.cfg.SettingsFactory - Query
language substitutions: {}
    [java] 854 [main] INFO org.hibernate.cfg.SettingsFactory - JPA-QL
strict compliance: disabled
    [java] 855 [main] INFO org.hibernate.cfg.SettingsFactory - Second-
level cache: enabled
    [java] 855 [main] INFO org.hibernate.cfg.SettingsFactory - Query
cache: disabled
    [java] 855 [main] INFO org.hibernate.cfg.SettingsFactory - Cache
region factory : org.hibernate.cache.impl.NoCachingRegionFactory
    [java] 855 [main] INFO org.hibernate.cfg.SettingsFactory -
Optimize cache for minimal puts: disabled
    [java] 857 [main] INFO org.hibernate.cfg.SettingsFactory -
Structured second-level cache entries: disabled
    [java] 864 [main] INFO org.hibernate.cfg.SettingsFactory - Echoing
all SQL to stdout
    [java] 866 [main] INFO org.hibernate.cfg.SettingsFactory -
Statistics: disabled
    [java] 866 [main] INFO org.hibernate.cfg.SettingsFactory - Deleted
entity synthetic identifier rollback: disabled
    [java] 867 [main] INFO org.hibernate.cfg.SettingsFactory - Default
entity-mode: pojo
    [java] 867 [main] INFO org.hibernate.cfg.SettingsFactory - Named
query checking : enabled
    [java] 932 [main] INFO org.hibernate.impl.SessionFactoryImpl -
building session factory
    [java] 1163 [main] INFO
org.hibernate.impl.SessionFactoryObjectFactory - Not binding factory to
JNDI, no JNDI name configured
    [java] Hibernate:
    [java] select
    [java] max(ID)
    [java] from

```

```

[java]          CUSTOMERS
[java] Hibernate:
[java]          insert
[java]          into
[java]          CUSTOMERS
[java]          (NAME, ADDRESS, NEXT_CUSTOMER_ID, ID)
[java]          values
[java]          (?, ?, ?, ?)
[java] Hibernate:
[java]          select
[java]          customer0_.ID as ID0_,
[java]          customer0_.NAME as NAME0_,
[java]          customer0_.ADDRESS as ADDRESS0_,
[java]          customer0_.NEXT_CUSTOMER_ID as NEXT4_0_
[java]          from
[java]          CUSTOMERS customer0_
[java]          order by
[java]          customer0_.NAME asc
[java] 1 customer(s) found:
[java] Linus Torvalds
[java] Finlandia
[java] 1530 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
[java]          1531 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

BUILD SUCCESSFUL
Total time: 3 seconds
[bpdp@bpdp-arch hellohibernate]$

```

### **Catatan:**

Hasil **ant** dan **ant run** di atas adalah hasil lengkap, untuk pembahasan berikutnya, biasanya akan menghasilkan output yang panjang juga tetapi hanya akan diambil bagian penting saja.

## **1.4 Latihan**

1. Pada direktori hellohibernate, eksekusilah **ant schemaexport**. Apa yang terjadi? Mengapa?
2. Direktori baru apa saja yang muncul setelah **ant** dan **ant run**? Jelaskan.
3. Tambahkan satu field lagi pada tabel CUSTOMERS, nama dan tipe serta isi bebas, setelah itu tampilkan juga pada hasilnya dengan menggunakan

**ant run.**

### **1.5 Tugas**

Buatlah proyek baru menggunakan Hibernate untuk mengambil semua data dari suatu tabel pada suatu database. Nama tabel dan definisi field serta isi data bebas.

## MODUL II

### JPA Menggunakan Hibernate

#### 2.1 Tujuan

1. Mahasiswa memahami fungsi dan kegunaan JPA serta keterkaitannya dengan standar JSR 220 dalam akses *persistent data*.
2. Mahasiswa memahami dasar-dasar penggunaan Hibernate Core, Hibernate Annotation, dan Hibernate Entity Manager untuk mengimplementasikan JPA.
3. Mahasiswa memahami struktur proyek JPA menggunakan Hibernate.
4. Mahasiswa mampu menggunakan Annotation dalam mendefinisikan Model.

#### 2.2 Teori Singkat

JPA (*Java Persistence API*) adalah framework yang merupakan standar dari Java untuk mengelola data pada basis data relasional dengan menggunakan teknologi obyek. JPA didefinisikan oleh JSR, saat ini JPA distandarkan dengan menggunakan JSR 220. Saat ini ada beberapa implementasi dari JPA oleh berbagai vendor, diantaranya adalah Hibernate, Apache OpenJPA, EclipseLink, Oracle TopLink, dan lain-lain. Standarisasi ini menyebabkan developer tidak mengalami kesulitan yang terlalu banyak pada saat mengimplementasikan persistent data dalam aplikasi karena paket yang serta metadata yang digunakan sama. JPA ini terdiri atas 3 bagian:

1. API (*Application Programming Interface*), didefinisikan pada paket *javax.persistence*.
2. JPQL (*Java Persistence Query Language*), bahasa *query* untuk mengakses persisten data, mirip dengan SQL.
3. Metadata.

JPA menggunakan Annotation dan bukan XML (seperti Customer.hbm.xml seperti di modul 1), meskipun Hibernate tetap mendukung penggunaan XML. Meskipun demikian, mulai saat ini akan digunakan Annotation dan Entity

Manager karena Annotation dan Entity Manager adalah standar yang terdapat pada JPA.

## 2.3 Praktik

Buatlah struktur direktori serta file-file sesuai dengan uraian berikut.

### Struktur Direktori dan File-file

```
bab2/
|-- build.xml
|-- etc
|   |-- META-INF
|   |   |-- persistence.xml
|   |-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
|   |-- hibernate-entitymanager.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   |-- slf4j-simple-1.5.8.jar
`-- src
    |-- hello
    |   |-- Customer.java
    |   |-- HelloHibernate.java
    |-- hibernate.cfg.xml
    |-- persistence
    |-- HibernateUtil.java
```

### Isi File-File

#### build.xml

```
<project name="HelloHibernateJPA" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="HelloHibernateJPA"/>
```

```

<property name="proj.version" value="1.0"/>
<!-- Global properties for this build -->
<property name="src.java.dir" value="src"/>
<property name="lib.dir" value="lib"/>
<property name="build.dir" value="bin"/>
<!-- Classpath declaration -->
<path id="project.classpath">
  <fileset dir="${lib.dir}">
    <include name="**/*.jar"/>
    <include name="**/*.zip"/>
  </fileset>
</path>
<!-- Useful shortcuts -->
<patternset id="meta.files">
  <include name="**/*.xml"/>
  <include name="**/*.properties"/>
</patternset>
<!-- Clean up -->
<target name="clean">
  <delete dir="${build.dir}"/>
  <mkdir dir="${build.dir}"/>
</target>
<!-- Compile Java source -->
<target name="compile" depends="clean">
  <mkdir dir="${build.dir}"/>
  <javac
    srcdir="${src.java.dir}"
    destdir="${build.dir}"
    nowarn="on">
    <classpath refid="project.classpath"/>
  </javac>
</target>
<!-- Copy metadata to build classpath -->
<property name="src.etc.dir" value="etc"/>
<target name="copymetafiles">
  <copy todir="${build.dir}">
    <fileset dir="${src.java.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
  <!-- Copy configuration files from etc/ -->
  <copy todir="${build.dir}">
    <fileset dir="${src.etc.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
</target>
<target name="run" depends="compile, copymetafiles"
  description="Build and run HelloHibernate JPA">
  <java fork="true"
    classname="hello.HelloHibernate"
    classpathref="project.classpath">
    <classpath path="${build.dir}"/>
  </java>

```

```

</target>
<taskdef name="hibernatetool"
  classname="org.hibernate.tool.ant.HibernateToolTask"
  classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
  description="Exports a generated schema to DB and file">
  <hibernatetool destdir="${basedir}">
  <classpath path="${build.dir}"/>
  <annotationconfiguration
    configurationfile="${build.dir}/hibernate.cfg.xml"/>
  <hbm2ddl
    drop="true"
    create="true"
    export="true"
    outputfilename="hellohibernatejpa-ddl.sql"
    delimiter=";"
    format="true"/>
  </hibernatetool>
</target>
</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="hellohibernate">
    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>

```

### log4j.properties

Sama dengan modul 1

### Customer.java

```

package hello;

import javax.persistence.*;

@Entity
@Table(name = "CUSTOMERS")
public class Customer {

    @Id @GeneratedValue

```

```

@Column(name = "ID")
private Long id;

@Column(name = "NAME")
private String name;

@Column(name = "ADDRESS")
private String address;

@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "NEXT_CUSTOMER_ID")
private Customer nextCustomer;

private Customer() {}

public Customer(String name, String address) {
    this.name = name;
    this.address = address;
}

public Long getId() {
    return id;
}

private void setId(Long 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 getNextCustomer() {
    return nextCustomer;
}

public void setNextCustomer(Customer nextCustomer) {
    this.nextCustomer = nextCustomer;
}
}

```

## HelloHibernate.java

```
package hello;

import java.util.*;
import org.hibernate.*;
import javax.persistence.*;

public class HelloHibernate {

    public static void main(String[] args) {

        // Start EntityManagerFactory
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("hellohibernate");
        // First unit of work
        EntityManager em = emf.createEntityManager();
        EntityTransaction tx = em.getTransaction();
        tx.begin();
        Customer customer = new Customer("Galvin King", "USA");
        em.persist(customer);
        tx.commit();
        em.close();
        // Second unit of work
        EntityManager newEm = emf.createEntityManager();
        EntityTransaction newTx = newEm.getTransaction();
        newTx.begin();
        List customers = newEm.createQuery("from Customer c order by c.name
asc").getResultList();
        System.out.println( customers.size() + " customer(s) found" );
        for (Object c : customers) {
            Customer loadedCust = (Customer) c;
            System.out.println(loadedCust.getName());
        }
        newTx.commit();
        newEm.close();
        // Shutting down the application
        emf.close();

    }
}
```

## hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/mydb
        </property>
    </session-factory>
</hibernate-configuration>
```

```

</property>
<property name="hibernate.connection.username">
    root
</property>
<property name="hibernate.connection.password">
    passwordmu
</property>
<property name="hibernate.dialect">
    org.hibernate.dialect.MySQL5Dialect
</property>
<!-- Use the C3P0 connection pool provider -->
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_period">3000</property>
<!-- Show and print nice SQL on stdout -->
<property name="show_sql">true</property>
<property name="format_sql">true</property>

<mapping class="hello.Customer" />
<mapping package="hello" />

</session-factory>
</hibernate-configuration>

```

### HibernateUtil.java

Sama dengan modul 1.

### Kompilasi dan Eksekusi Hasil Kompilasi

Kompilasilah dengan menggunakan **ant** (lihat modul 1). Berikut adalah sebagian hasil dari **ant run** yang digunakan untuk mengeksekusi hasil kompilasi sesuai dengan task di apache ant:

```

.....
.....
[java] Hibernate:
[java]     insert
[java]     into
[java]         CUSTOMERS
[java]         (ADDRESS, NAME, NEXT_CUSTOMER_ID)
[java]     values
[java]         (?, ?, ?)
[java] Hibernate:
[java]     select
[java]         customer0_.ID as ID_,
[java]         customer0_.ADDRESS as ADDRESS0_,

```

```

[java]         customer0_.NAME as NAME0_,
[java]         customer0_.NEXT_CUSTOMER_ID as NEXT4_0_
[java]         from
[java]         CUSTOMERS customer0_
[java]         order by
[java]         customer0_.NAME asc
[java] 2 customer(s) found
[java] Galvin King
[java] Linus Torvalds
[java] 2868 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
                                [java]         2868         [main]         INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

BUILD SUCCESSFUL
Total time: 4 seconds

```

## 2.4 Latihan

1. Hapus file **persistence.xml**, setelah itu kompilasi ulang dan eksekusilah hasil kompilasi tersebut. Bisakah proses tersebut dilaksanakan? Mengapa?
2. Tambahkan program di atas untuk mengisi 5 data lagi ke dalam tabel CUSTOMERS.

## 2.5 Tugas

1. Dengan menggunakan JPA dari Hibernate, buatlah suatu proyek baru untuk mengisi data ke suatu tabel baru serta menampilkan data yang telah diisi tersebut.
2. Jelaskan maksud dari Annotation `@Entity`, `@Table`, `@Id`, `@GeneratedValue` dan `@Column`

## MODUL III

### Relasi Antar Tabel: Pengenalan dan Relasi One-to-One

#### 3.1 Tujuan

1. Mahasiswa memahami pengertian relasi antar tabel
2. Mahasiswa memahami tipe-tipe relasi yang bisa diimplementasikan di dalam Hibernate JPA
3. Mahasiswa mampu menggunakan Hibernate JPA untuk relasi One-to-One.

#### 3.2 Teori Singkat

Suatu tabel sering juga disebut sebagai suatu Entity, meskipun dalam dunia teknologi obyek / ORM tidak selalu satu tabel / entity berarti satu Model. Dalam suatu aplikasi, biasanya terdapat keterkaitan antar entity sehingga memungkinkan data yang akan dikelola tidak *redundant* dan tidak konsisten. Untuk mendapatkan berbagai entity serta keterkaitan antar entitiy tersebut, biasanya akan dilakukan analisis yang disebut sebagai proses **normalisasi**. Ada beberapa tipe keterkaitan antar entity yang bisa diimplementasikan di Hibernate JPA:

1. One-to-One: entity A hanya mempunyai satu komponen di entity B, demikian juga sebaliknya.
2. One-to-Many: entity A mempunya lebih dari satu komponen di entity B, sedangkan satu entity B pasti memiliki keterkaitan dengan satu entity A.
3. Many-to-One: banyak komponen di entity A mempunyai keterkaitan dengan satu komponen di entity B, sedangkan satu entity B memiliki keterkaitan dengan banyak komponen di entity A.
4. Many-to-One: banyak komponen di entity A mempunyai keterkaitan dengan banyak komponen di entity B dan sebaliknya.

Pada Hibernate JPA, keterkaitan **One-to-One** bisa diimplementasikan menggunakan Annotation `@OneToOne`.

### 3.3 Praktik

#### Tabel-tabel MySQL

Praktik ini menggunakan 2 tabel yaitu STUDENTS (untuk menampung data mahasiswa) dan LOCKERS (untuk menampung data loker per mahasiswa). Berikut skrip SQL untuk membuat 2 tabel tersebut:

```
CREATE TABLE `STUDENTS` (`id` bigint(20) NOT NULL AUTO_INCREMENT,  
`name` varchar(50) COLLATE latin1_general_ci NOT NULL, `address`  
varchar(150) COLLATE latin1_general_ci NOT NULL, PRIMARY KEY (`id`))  
ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1  
COLLATE=latin1_general_ci
```

```
CREATE TABLE `LOCKERS` (`id` bigint(20) NOT NULL AUTO_INCREMENT,  
`name` varchar(10) COLLATE latin1_general_ci NOT NULL, `student_ID`  
bigint(20) DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB  
AUTO_INCREMENT=2 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

Hubungan relasi antara 2 tabel ini adalah one-to-one, setiap mahasiswa di tabel STUDENTS mempunyai satu loker di tabel LOCKERS dan sebaliknya.

#### Struktur Direktori dan Berbagai File

```
bab3  
|-- build.xml  
|-- etc  
|   |-- META-INF  
|   |   `-- persistence.xml  
|   `-- log4j.properties  
|-- lib  
|   |-- antlr-2.7.6.jar  
|   |-- asm-attrs.jar  
|   |-- asm.jar  
|   |-- c3p0-0.9.1.jar  
|   |-- cglib-2.2.jar  
|   |-- commons-collections-3.1.jar  
|   |-- commons-logging-1.1.1.jar  
|   |-- dom4j-1.6.1.jar  
|   |-- ejb3-persistence.jar  
|   |-- freemarker.jar
```

```

| |-- hibernate-annotations.jar
| |-- hibernate-commons-annotations.jar
| |-- hibernate-core.jar
| |-- hibernate-entitymanager.jar
| |-- hibernate-tools.jar
| |-- hibernate3.jar
| |-- javassist-3.9.0.GA.jar
| |-- jta-1.1.jar
| |-- mysql-connector-java-5.1.8-bin.jar
| |-- slf4j-api-1.5.8.jar
| `-- slf4j-simple-1.5.8.jar
`-- src
    |-- hibernate.cfg.xml
    |-- onetoone
    | |-- Locker.java
    | |-- OneToOne.java
    | `-- Student.java
    `-- persistence
        `-- HibernateUtil.java

```

### build.xml

```

<project name="OneToOne" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="OneToOne"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>
  <!-- Useful shortcuts -->
  <patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
  </patternset>
  <!-- Clean up -->
  <target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
  </target>
  <!-- Compile Java source -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>

```

```

        <javac
            srcdir="${src.java.dir}"
            destdir="${build.dir}"
            nowarn="on">
            <classpath refid="project.classpath"/>
        </javac>
    </target>
    <!-- Copy metadata to build classpath -->
    <property name="src.etc.dir" value="etc"/>
    <target name="copymetafiles">
        <copy todir="${build.dir}">
            <fileset dir="${src.java.dir}">
                <patternset refid="meta.files"/>
            </fileset>
        </copy>
        <!-- Copy configuration files from etc/ -->
        <copy todir="${build.dir}">
            <fileset dir="${src.etc.dir}">
                <patternset refid="meta.files"/>
            </fileset>
        </copy>
    </target>
    <target name="run" depends="compile, copymetafiles"
        description="Build and run OneToOne">
        <java fork="true"
            classname="onetoone.OneToOne"
            classpathref="project.classpath">
            <classpath path="${build.dir}"/>
        </java>
    </target>

    <taskdef name="hibernatetool"
        classname="org.hibernate.tool.ant.HibernateToolTask"
        classpathref="project.classpath"/>
    <target name="schemaexport" depends="compile, copymetafiles"
        description="Exports a generated schema to DB and file">
        <hibernatetool destdir="${basedir}">
        <classpath path="${build.dir}"/>
        <annotationconfiguration
            configurationfile="${build.dir}/hibernate.cfg.xml"/>
        <hbm2ddl
            drop="true"
            create="true"
            export="true"
            outputfilename="onetoone-ddl.sql"
            delimiter=";"
            format="true"/>
        </hibernatetool>
    </target>
</project>

```

## persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="onetoone">
    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>
```

## log4j.properties

Sama seperti di modul 1

## hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
      root
    </property>
    <property name="hibernate.connection.password">
      passwordmu
    </property>
    <property name="hibernate.dialect">
      org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="onetoone.Student" />
    <mapping package="onetoone" />
  </session-factory>
</hibernate-configuration>
```

```

    <mapping class="onetoone.Locker" />
    <mapping package="onetoone" />

</session-factory>
</hibernate-configuration>

```

### Locker.java

```

package onetoone;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

@Entity
@Table(name = "LOCKERS")
public class Locker {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    public Long getId() {
        return id;
    }

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

    @Column(name = "NAME", nullable=false, length=10, insertable=true)
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @OneToOne(cascade=CascadeType.ALL)
    private Student student;

```

```

public Student getStudent() {
    return student;
}

public void setStudent(Student student) {
    this.student = student;
}
}

```

## Student.java

```

package onetoone;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENTS")
public class Student {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)
    private String name;

    @Column(name = "ADDRESS", nullable=false, length=150,
insertable=true)
    private String address;

    public Long getId() {
        return id;
    }

    private void setId(Long 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;
}
}

```

## OneToOne.java

```

package onetoone;

import java.util.*;
import org.hibernate.*;
import javax.persistence.*;

/**
 * @author Bambang Purnomosidi D. P.
 *
 */

public class OneToOne {

    /**
     * @param args
     */
    public static void main(String[] args) {

        EntityManager em = null;
        EntityManagerFactory emf = null;
        try {
            emf = Persistence.createEntityManagerFactory("onetoone", new
HashMap());
            em = emf.createEntityManager();
            em.getTransaction().begin();
            Student std = new Student();
            Locker lkr = new Locker();
            std.setName("Zaky Ahmad Aditya");
            std.setAddress("Griya Purwa Asri H-304");
            lkr.setName("LKR - 02");
            lkr.setStudent(std);
            em.persist(lkr);
            em.getTransaction().commit();
            em.getTransaction().begin();
            Student std1 = new Student();
            Locker lkr1 = new Locker();
            std1.setName("Bambang Purnomosidi D. P.");
            std1.setAddress("GPA H - 304");
            lkr1.setName("LKR - 01");
            lkr1.setStudent(std1);
            em.persist(lkr1);
            em.getTransaction().commit();

```

```

System.out.println("Done inserting data");

EntityManager newEm = emf.createEntityManager();
EntityTransaction newTx = newEm.getTransaction();
newTx.begin();

List lockers = newEm.createQuery("from Locker").getResultList();
System.out.println( lockers.size() + " locker(s) found" );
for (Object l : lockers) {
    Locker loadedLkr = (Locker) l;
    System.out.println(loadedLkr.getName() + " is used by " +
loadedLkr.getStudent().getName());
}

    newTx.commit();
    newEm.close();
} catch (Exception e) {
    System.out.println(e.getMessage());
} finally {
    em.close();
    emf.close();
}
}
}

```

### HibernateUtil.java

Sama seperti pada modul 1

### Hasil Kompilasi dan Eksekusi

Kompilasi menggunakan **ant** dan kemudian jalankan dengan **ant run**. Berikut adalah sebagian hasil saat hasil kompilasi dieksekusi:

```

.....
.....
[java] Hibernate:
[java]      insert
[java]      into
[java]          STUDENTS
[java]          (ADDRESS, NAME)
[java]      values
[java]          (?, ?)
[java] Hibernate:
[java]      insert
[java]      into
[java]          LOCKERS

```

```

[java]         (NAME, student_ID)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         STUDENTS
[java]         (ADDRESS, NAME)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         LOCKERS
[java]         (NAME, student_ID)
[java]     values
[java]         (?, ?)
[java] Done inserting data
[java] Hibernate:
[java]     select
[java]         locker0_.ID as ID1_,
[java]         locker0_.NAME as NAME1_,
[java]         locker0_.student_ID as student3_1_
[java]     from
[java]         LOCKERS locker0_
[java] Hibernate:
[java]     select
[java]         student0_.ID as ID0_0_,
[java]         student0_.ADDRESS as ADDRESS0_0_,
[java]         student0_.NAME as NAME0_0_
[java]     from
[java]         STUDENTS student0_
[java]     where
[java]         student0_.ID=?
[java] Hibernate:
[java]     select
[java]         student0_.ID as ID0_0_,
[java]         student0_.ADDRESS as ADDRESS0_0_,
[java]         student0_.NAME as NAME0_0_
[java]     from
[java]         STUDENTS student0_
[java]     where
[java]         student0_.ID=?
[java] 2 locker(s) found
[java] LKR - 02 is used by Zaky Ahmad Aditya
[java] LKR - 01 is used by Bambang Purnomosidi D. P.
[java] 1607 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
[java]         1608 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

```

BUILD SUCCESSFUL  
Total time: 3 seconds

```
[bpdp@bpdp-arch bab3]$
```

Hasil di tabel adalah sebagai berikut (mungkin lain dengan yang anda kerjakan):

```
mysql> select * from LOCKERS;
+----+-----+-----+
| id | name      | student_ID |
+----+-----+-----+
| 13 | LKR - 02 |          14 |
| 14 | LKR - 01 |          15 |
+----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from STUDENTS;
+----+-----+-----+
| id | name                | address                |
+----+-----+-----+
| 14 | Zaky Ahmad Aditya   | Griya Purwa Asri H-304 |
| 15 | Bambang Purnomosidi D. P. | GPA H - 304          |
+----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

### 3.4 Latihan

1. Modifikasilah program di atas untuk mengisikan 2 data lagi dan menampilkan hasilnya.
2. Cobalah menghapus tabel yang anda buat kemudian anda kompilasi dan jalankan lagi, apa yang terjadi? Jelaskan.

### 3.5 Tugas

Buatlah proyek Hibernate JPA baru untuk mengakses data persistent pada 2 entity yang mempunyai hubungan one-to-many. Nama tabel, definisi field, serta isi data bebas. Program yang anda buat harus bisa mengisikan data dan menampilkan data yang telah anda isikan tersebut.

## MODUL IV

### Relasi One-to-Many dan Many-to-One

#### 4.1 Tujuan

1. Mahasiswa memahami tipe relasi One-to-Many dan Many-to-One
2. Mahasiswa mampu mengimplementasikan relasi One-to-Many dan Many-to-One menggunakan Hibernate JPA.

#### 4.2 Teori Singkat

Relasi One-to-Many dan Many-to-One pada dasarnya sama, tetapi perlu didefinisikan di depan untuk mengantisipasi akan diakses dari unsur induk dulu atau unsur anak. Relasi ini digunakan untuk dua entity yang salah satu merupakan induk dan mempunyai lebih dari satu data terkait pada entity anak.

Pada Hibernate (JPA maupun non JPA), bisa digunakan 3 tabel entity untuk mengimplementasikan tipe relasi ini, yaitu:

1. Tabel induk
2. Tabel anak
3. Tabel penghubung.

Pada tabel MySQL, tipe tabel yang digunakan adalah InnoDB, bukan MyISAM.

Pada masing-masing definisi model, digunakan Annotation berikut ini:

1. @OneToMany
2. @ManyToOne

#### 4.3 Praktik

##### Persiapan Tabel MySQL

Ada 3 tabel untuk keperluan ini yaitu:

1. Tabel induk (STUDENTS, menampung data mahasiswa)
2. Tabel anak (MARKS, menampung data nilai)
3. Tabel penghubung (STUDENTSMARKS, menghubungkan data mahasiswa dengan data nilai).

Berikut adalah definisi tabel-tabel berikut dengan perintah SQL:

```
CREATE TABLE `STUDENTS` (`id` bigint(20) NOT NULL
AUTO_INCREMENT, `name` varchar(50) COLLATE
latin1_general_ci NOT NULL, `address` varchar(150) COLLATE
latin1_general_ci NOT NULL, PRIMARY KEY (`id`))
ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci
```

```
CREATE TABLE `MARKS` (`id` bigint(20) NOT NULL
AUTO_INCREMENT, `subject` varchar(50) COLLATE
latin1_general_ci NOT NULL, `mark` char(1) COLLATE
latin1_general_ci NOT NULL, PRIMARY KEY (`id`))
ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci
```

```
CREATE TABLE `STUDENTSMARKS` (`student_id` bigint(20)
DEFAULT NULL, `mark_id` bigint(20) DEFAULT NULL, KEY
`mark_id` (`mark_id`), KEY `student_id` (`student_id`),
CONSTRAINT `STUDENTSMARKS_ibfk_1` FOREIGN KEY (`mark_id`)
REFERENCES `MARKS` (`id`), CONSTRAINT
`STUDENTSMARKS_ibfk_2` FOREIGN KEY (`student_id`)
REFERENCES `STUDENTS` (`id`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1 COLLATE=latin1_general_ci
```

## **One-To-Many**

### **Struktur Direktori dan File-file**

```
one-to-many
|-- build.xml
|-- etc
|   |-- META-INF
|   |   |-- persistence.xml
|   |-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
```

```

| |-- hibernate-entitymanager.jar
| |-- hibernate-tools.jar
| |-- hibernate3.jar
| |-- javassist-3.9.0.GA.jar
| |-- jta-1.1.jar
| |-- mysql-connector-java-5.1.8-bin.jar
| |-- slf4j-api-1.5.8.jar
| `-- slf4j-simple-1.5.8.jar
|-- src
| |-- hibernate.cfg.xml
| |-- onetomany
| | |-- Mark.java
| | |-- OneToMany.java
| | |-- Student.java
| |-- persistence
| `-- HibernateUtil.java

```

### build.xml

```

<project name="OneToMany" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="OneToMany"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>
  <!-- Useful shortcuts -->
  <patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
  </patternset>
  <!-- Clean up -->
  <target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
  </target>
  <!-- Compile Java source -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac
      srcdir="${src.java.dir}"
      destdir="${build.dir}"
      nowarn="on">
      <classpath refid="project.classpath"/>
    </javac>

```

```

</target>
<!-- Copy metadata to build classpath -->
<property name="src.etc.dir" value="etc"/>
<target name="copymetafiles">
  <copy todir="${build.dir}">
    <fileset dir="${src.java.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
  <!-- Copy configuration files from etc/ -->
  <copy todir="${build.dir}">
    <fileset dir="${src.etc.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
</target>
<target name="run" depends="compile, copymetafiles"
  description="Build and run OneToMany">
  <java fork="true"
    classname="onetomany.OneToMany"
    classpathref="project.classpath">
    <classpath path="${build.dir}"/>
  </java>
</target>

<taskdef name="hibernatetool"
  classname="org.hibernate.tool.ant.HibernateToolTask"
  classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
  description="Exports a generated schema to DB and file">
  <hibernatetool destdir="${basedir}">
  <classpath path="${build.dir}"/>
  <annotationconfiguration
    configurationfile="${build.dir}/hibernate.cfg.xml"/>
  <hbm2ddl
    drop="true"
    create="true"
    export="true"
    outputfilename="onetomany-ddl.sql"
    delimiter=";"
    format="true"/>
  </hibernatetool>
</target>

</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">

```

```

    <persistence-unit name="onetomany">
      <properties>
        <property name="hibernate.ejb.cfgfile"
          value="/hibernate.cfg.xml"/>
      </properties>
    </persistence-unit>
  </persistence>

```

## log4j.properties

Sama seperti modul 1

## hibernate.cfg.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="onetomany">
    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>

```

[bpdp@bpdp-arch one-to-many]\$ cat src/hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
      root
    </property>
    <property name="hibernate.connection.password">
      passwordmu
    </property>
    <property name="hibernate.dialect">
      org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>

```

```

    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="onetomany.Student" />
    <mapping package="onetomany" />
    <mapping class="onetomany.Mark" />
    <mapping package="onetomany" />

</session-factory>
</hibernate-configuration>

```

## Student.java

```

package onetomany;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENTS")
public class Student {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)
    private String name;

    @Column(name = "ADDRESS", nullable=false, length=150,
insertable=true)
    private String address;

    public Long getId() {
        return id;
    }

    private void setId(Long 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;
}

@OneToMany(cascade=CascadeType.ALL)
@JoinTable(name = "STUDENTSMARKS", joinColumns = {
    @JoinColumn(name="student_id", unique = true)
}, inverseJoinColumns = {
    @JoinColumn(name="mark_id")
}
)

private Set<Mark> mark;

/**
 * @return student's mark
 */
public Set<Mark> getMark() {
    return mark;
}

/**
 * @param children the children to set
 */
public void setMark(Set<Mark> mark) {
    this.mark = mark;
}
}

```

### Mark.java

```

package onetomany;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;

```

```

import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "MARKS")
public class Mark {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    public Long getId() {
        return id;
    }

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

    @Column(name = "SUBJECT", nullable=false, length=50, insertable=true)
    private String subject;

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    @Column(name = "MARK", nullable=false, length=1, insertable=true)
    private String mark;

    public String getMark() {
        return mark;
    }

    public void setMark(String mark) {
        this.mark = mark;
    }

    @ManyToOne(optional=true)
    @JoinTable(name = "STUDENTSMARKS", joinColumns = {
        @JoinColumn(name="mark_id")
    }, inverseJoinColumns = {
        @JoinColumn(name="student_id")
    })
    private Student student;

```

```

/**
 * @return the student
 */
public Student getStudent() {
    return student;
}

/**
 * @param student the student to set
 */
public void setStudent(Student student) {
    this.student = student;
}
}

```

### OneToMany.java

```

package onetomany;

import java.util.HashMap;
import java.util.HashSet;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * @author bpdp
 */
public class OneToMany {

    /**
     * @param args
     */
    public static void main(String[] args) {

        EntityManagerFactory emf=null;
        EntityManager em=null;

        try {

            emf=Persistence.createEntityManagerFactory("onetomany", new
HashMap());
            em=emf.createEntityManager();
            em.getTransaction().begin();

            Student student = new Student();
            student.setName("Zaky Ahmad Aditya");
            student.setAddress("Griya Purwa Asri H-304");

```

```

Mark mark = new Mark();
mark.setSubject("ORM Programming");
mark.setMark("A");
Mark mark1 = new Mark();
mark1.setSubject("Object-Oriented Programming");
mark1.setMark("B");
Mark mark2 = new Mark();
mark2.setSubject("Enterprise Information System");
mark2.setMark("C");

HashSet childSet = new HashSet();
childSet.add(mark);
childSet.add(mark1);
childSet.add(mark2);

student.setMark(childSet);
em.persist(student);
em.getTransaction().commit();
System.out.println("Done inserting data");

} catch(Exception e) {
System.out.println(e.getMessage());
} finally {
emf.close();
em.close();
}
}
}

```

### HibernateUtil.java

Sama dengan modul 1.

### **Kompilasi dan Eksekusi Hasil Kompilasi**

Gunakan **ant** untuk kompilasi dan **ant run** untuk eksekusi hasil kompilasi.

Berikut adalah potongan hasil dari **ant run**:

```

.....
.....
[java] Hibernate:
[java]      insert
[java]      into
[java]          STUDENTS
[java]          (ADDRESS, NAME)
[java]      values
[java]          (?, ?)

```

```

[java] Hibernate:
[java]     insert
[java]     into
[java]         MARKS
[java]         (MARK, SUBJECT)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         MARKS
[java]         (MARK, SUBJECT)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         MARKS
[java]         (MARK, SUBJECT)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         STUDENTSMARKS
[java]         (student_id, mark_id)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         STUDENTSMARKS
[java]         (student_id, mark_id)
[java]     values
[java]         (?, ?)
[java] Hibernate:
[java]     insert
[java]     into
[java]         STUDENTSMARKS
[java]         (student_id, mark_id)
[java]     values
[java]         (?, ?)
[java] Done inserting data
[java] 2143 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
[java] 2143 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

BUILD SUCCESSFUL
Total time: 3 seconds

```

Jika dilihat pada tabel di MySQL, hasilnya adalah sebagai berikut:

```
mysql> select * from STUDENTS;
+-----+-----+-----+
| id | name                | address                |
+-----+-----+-----+
| 1  | Zaky Ahmad Aditya  | Griya Purwa Asri H-304 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> select * from MARKS;
+-----+-----+-----+
| id | subject                | mark |
+-----+-----+-----+
| 1  | ORM Programming       | A    |
| 2  | Enterprise Information System | C    |
| 3  | Object-Oriented Programming | B    |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> select * from STUDENTSMARKS;
+-----+-----+
| student_id | mark_id |
+-----+-----+
|          1 |        1 |
|          1 |        2 |
|          1 |        3 |
+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql>
```

## **Many-to-One**

### **Struktur Direktori dan File-file**

```
many-to-one
|-- build.xml
|-- etc
|   |-- META-INF
|   |   |-- persistence.xml
|   |-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
```

```

| |-- hibernate-core.jar
| |-- hibernate-entitymanager.jar
| |-- hibernate-tools.jar
| |-- hibernate3.jar
| |-- javassist-3.9.0.GA.jar
| |-- jta-1.1.jar
| |-- mysql-connector-java-5.1.8-bin.jar
| |-- slf4j-api-1.5.8.jar
| `-- slf4j-simple-1.5.8.jar
|-- src
| |-- hibernate.cfg.xml
| |-- manytoone
| | |-- ManyToOne.java
| | |-- Mark.java
| | `-- Student.java
|-- persistence
| `-- HibernateUtil.java

```

### build.xml

```

<project name="ManyToOne" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="ManyToOne"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>
  <!-- Useful shortcuts -->
  <patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
  </patternset>
  <!-- Clean up -->
  <target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
  </target>
  <!-- Compile Java source -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac
      srcdir="${src.java.dir}"
      destdir="${build.dir}"
      nowarn="on">
      <classpath refid="project.classpath"/>
    </javac>
  </target>

```

```

    </javac>
</target>
<!-- Copy metadata to build classpath -->
<property name="src.etc.dir" value="etc"/>
<target name="copymetafiles">
    <copy todir="${build.dir}">
        <fileset dir="${src.java.dir}">
            <patternset refid="meta.files"/>
        </fileset>
    </copy>
    <!-- Copy configuration files from etc/ -->
    <copy todir="${build.dir}">
        <fileset dir="${src.etc.dir}">
            <patternset refid="meta.files"/>
        </fileset>
    </copy>
</target>
<target name="run" depends="compile, copymetafiles"
description="Build and run ManyToOne">
    <java fork="true"
        classname="manytoone.ManyToOne"
        classpathref="project.classpath">
        <classpath path="${build.dir}"/>
    </java>
</target>
<taskdef name="hibernatetool"
    classname="org.hibernate.tool.ant.HibernateToolTask"
    classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
description="Exports a generated schema to DB and file">
    <hibernatetool destdir="${basedir}">
    <classpath path="${build.dir}"/>
    <annotationconfiguration
        configurationfile="${build.dir}/hibernate.cfg.xml"/>
    <hbm2ddl
        drop="true"
        create="true"
        export="true"
        outputfilename="manytoone-ddl.sql"
        delimiter=";"
        format="true"/>
    </hibernatetool>
</target>
</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="manytoone">

```

```

    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>

```

## log4j.properties

Sama seperti modul 1

## hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
      root
    </property>
    <property name="hibernate.connection.password">
      passwordmu
    </property>
    <property name="hibernate.dialect">
      org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="manytoone.Student" />
    <mapping package="manytoone" />
    <mapping class="manytoone.Mark" />
    <mapping package="manytoone" />

  </session-factory>
</hibernate-configuration>

```

## Student.java

```
package manytoone;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENTS")
public class Student {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)
    private String name;

    @Column(name = "ADDRESS", nullable=false, length=150,
insertable=true)
    private String address;

    public Long getId() {
        return id;
    }

    private void setId(Long 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;
    }
}
```

```

@OneToMany(cascade=CascadeType.ALL)
@JoinTable(name = "STUDENTSMARKS", joinColumns = {
    @JoinColumn(name="student_id", unique = true)
}, inverseJoinColumns = {
    @JoinColumn(name="mark_id")
}
)

private Set<Mark> mark;

/**
 * @return student's mark
 */
public Set<Mark> getMark() {
    return mark;
}

/**
 * @param children the children to set
 */
public void setMark(Set<Mark> mark) {
    this.mark = mark;
}
}

```

## Mark.java

```

package manytoone;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "MARKS")
public class Mark {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    public Long getId() {

```

```

    return id;
}

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

@Column(name = "SUBJECT", nullable=false, length=50, insertable=true)
private String subject;

public String getSubject() {
    return subject;
}

public void setSubject(String subject) {
    this.subject = subject;
}

@Column(name = "MARK", nullable=false, length=1, insertable=true)
private String mark;

public String getMark() {
    return mark;
}

public void setMark(String mark) {
    this.mark = mark;
}

@ManyToOne(optional=true)
@JoinTable(name = "STUDENTSMARKS", joinColumns = {
    @JoinColumn(name="mark_id")
}, inverseJoinColumns = {
    @JoinColumn(name="student_id")
}
)

private Student student;

/**
 * @return the student
 */
public Student getStudent() {
    return student;
}

/**
 * @param student the student to set
 */
public void setStudent(Student student) {
    this.student = student;
}

```

```
}
```

## ManyToOne.java

```
package manytoone;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * @author bpdp
 *
 */

public class ManyToOne {

    public static void main(String[] args) {

        EntityManagerFactory emf=null;
        EntityManager em=null;

        try {

            emf=Persistence.createEntityManagerFactory("manytoone", new
HashMap());
            em=emf.createEntityManager();
            em.getTransaction().begin();

            Mark mark = new Mark();
            mark.setSubject("Java Web Programming");
            mark.setMark("A");
            Mark mark1 = new Mark();
            mark1.setSubject("Operating System");
            mark1.setMark("B");

            HashSet childSet=new HashSet();
            childSet.add(mark);
            childSet.add(mark1);

            Student student = new Student();
            student.setName("Bambang Purnomosidi D. P.");
            student.setAddress("GPA H-304");

            student.setMark(childSet);
            em.persist(student);

            Student studentRecord = em.find(Student.class, student.getId());
```

```

System.out.println("Student: "+studentRecord.getName());
Set<Mark> studentSet = studentRecord.getMark();
Iterator it = studentSet.iterator();
while (it.hasNext()){
    Mark mrk = (Mark)it.next();
    System.out.println("Subject: "+mrk.getSubject());
    System.out.println("Mark: "+mrk.getMark());
}

em.getTransaction().commit();
System.out.println("Done");

} catch(Exception e) {
    System.out.println(e.getMessage());
} finally {
    emf.close();
    em.close();
}
}
}
}

```

### HibernateUtil.java

Sama dengan modul 1

### **Kompilasi dan Eksekusi Hasil Kompilasi**

Untuk mengkompilasi, seperti biasa gunakan **ant**, dan **ant run** untuk mengeksekusi hasil kompilasi. Berikut adalah potongan hasil dari **ant run**:

```

.....
.....
[java] Hibernate:
[java]      insert
[java]      into
[java]          STUDENTS
[java]          (ADDRESS, NAME)
[java]      values
[java]          (?, ?)
[java] Hibernate:
[java]      insert
[java]      into
[java]          MARKS
[java]          (MARK, SUBJECT)
[java]      values
[java]          (?, ?)
[java] Hibernate:
[java]      insert

```

```

[java]      into
[java]      MARKS
[java]      (MARK, SUBJECT)
[java]      values
[java]      (?, ?)
[java] Student: Bambang Purnomosidi D. P.
[java] Subject: Java Web Programming
[java] Mark: A
[java] Subject: Operating System
[java] Mark: B
[java] Hibernate:
[java]      insert
[java]      into
[java]      STUDENTSMARKS
[java]      (student_id, mark_id)
[java]      values
[java]      (?, ?)
[java] Hibernate:
[java]      insert
[java]      into
[java]      STUDENTSMARKS
[java]      (student_id, mark_id)
[java]      values
[java]      (?, ?)
[java] Done
[java] 1545 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
[java]      1545 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

```

BUILD SUCCESSFUL  
Total time: 3 seconds

Jika dilihat pada tabel di MySQL, hasilnya adalah sebagai berikut:

```

mysql> select * from STUDENTS;
+-----+-----+-----+
| id | name                | address                |
+-----+-----+-----+
| 1  | Zaky Ahmad Aditya   | Griya Purwa Asri H-304 |
| 2  | Bambang Purnomosidi D. P. | GPA H-304             |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from MARKS;
+-----+-----+-----+
| id | subject                | mark |
+-----+-----+-----+
| 1  | ORM Programming       | A    |
| 2  | Enterprise Information System | C    |
+-----+-----+-----+

```

```

| 3 | Object-Oriented Programming | B |
| 4 | Java Web Programming         | A |
| 5 | Operating System              | B |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

```
mysql> select * from STUDENTSMARKS;
```

```

+-----+-----+
| student_id | mark_id |
+-----+-----+
|          1 |        1 |
|          1 |        2 |
|          1 |        3 |
|          2 |        4 |
|          2 |        5 |
+-----+-----+
5 rows in set (0.00 sec)

```

```
mysql>
```

#### 4.4 Latihan

1. Tambahkan satu field baru pada STUDENTS, yaitu field **major** untuk menyimpan data program studi dari mahasiswa yang bersangkutan. Setelah itu modifikasilah program untuk mengantisipasi penambahan field tersebut.

#### 4.5 Tugas

Buat proyek baru dengan Hibernate JPA untuk mengimplementasikan One-to-Many dan Many-to-Many dengan tabel yang anda buat sendiri.

## MODUL V

### Relasi Many-to-Many

#### 5.1 Tujuan

1. Mahasiswa memahami relasi Many-to-Many pada dua entity.
2. Mahasiswa mampu memahami dan menggunakan dasar-dasar penggunaan Hibernate JPA untuk mengimplementasikan relasi Many-to-Many

#### 5.2 Teori Singkat

Relasi Many-to-Many antara 2 entity adalah relasi yang memungkinkan masing-masing entity untuk mempunyai komponen data lebih dari satu pada tabel satunya. Contoh sederhana dari konsep ini adalah relasi antara tabel mahasiswa dengan kuliah, mahasiswa mungkin mengambil lebih dari satu mata kuliah, sementara satu mata kuliah bisa diambil lebih dari satu mahasiswa.

Hibernate JPA mengimplementasikan relasi Many-to-Many dengan mendefinisikan 3 tabel, dengan dua tabel yang terhubung serta satu tabel sebagai penghubung antara kedua tabel tersebut. Hibernate JPA mengimplementasikan Many-to-Many menggunakan Annotation **@ManyToMany**.

#### 5.3 Praktik

##### Persiapan Tabel MySQL

Ada 3 tabel yang harus dipersiapkan disini:

1. Tabel STUDENTS, untuk menampung data mahasiswa.
2. Tabel SUBJECTS, untuk menampung data mata kuliah.
3. Tabel STUDENTSSUBJECTS, merupakan tabel penghubung antara STUDENTS dengan SUBJECTS.

Berikut ini adalah skrip SQL untuk membuat tabel-tabel tersebut:

```
CREATE TABLE `STUDENTS` ( `id` bigint(20) NOT NULL AUTO_INCREMENT,
`name` varchar(50) COLLATE latin1_general_ci NOT NULL, `address`
varchar(150) COLLATE latin1_general_ci NOT NULL, PRIMARY KEY (`id`) )
ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci
```

```
CREATE TABLE `SUBJECTS` ( `id` bigint(20) NOT NULL AUTO_INCREMENT,
`subject` varchar(50) COLLATE latin1_general_ci NOT NULL, PRIMARY KEY
(`id`) ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci
```

```
CREATE TABLE `STUDENTSSUBJECTS` ( `student_id` bigint(20) DEFAULT NULL,
`subject_id` bigint(20) DEFAULT NULL, KEY `subject_id` (`subject_id`),
KEY `student_id` (`student_id`), CONSTRAINT `STUDENTSSUBJECTS_ibfk_1`
FOREIGN KEY (`subject_id`) REFERENCES `SUBJECTS` (`id`), CONSTRAINT
`STUDENTSSUBJECTS_ibfk_2` FOREIGN KEY (`student_id`) REFERENCES
`STUDENTS` (`id`) ENGINE=InnoDB DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci
```

## **Struktur Direktori dan Berbagai File**

```
bab5/
|-- build.xml
|-- etc
|   |-- META-INF
|   |   `-- persistence.xml
|   `-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
|   |-- hibernate-entitymanager.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   `-- slf4j-simple-1.5.8.jar
`-- src
    |-- hibernate.cfg.xml
    |-- manytomany
    |   |-- ManyToMany.java
```

```

| |-- Student.java
| |-- Subject.java
|-- persistence
   |-- HibernateUtil.java

```

### build.xml

```

<project name="ManyToMany" default="compile" basedir=". ">
  <!-- Name of project and version -->
  <property name="proj.name" value="ManyToMany"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>
  <!-- Useful shortcuts -->
  <patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
  </patternset>
  <!-- Clean up -->
  <target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
  </target>
  <!-- Compile Java source -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac
      srcdir="${src.java.dir}"
      destdir="${build.dir}"
      nowarn="on">
      <classpath refid="project.classpath"/>
    </javac>
  </target>
  <!-- Copy metadata to build classpath -->
  <property name="src.etc.dir" value="etc"/>
  <target name="copymetafiles">
    <copy todir="${build.dir}">
      <fileset dir="${src.java.dir}">
        <patternset refid="meta.files"/>
      </fileset>
    </copy>
  <!-- Copy configuration files from etc/ -->
  <copy todir="${build.dir}">
    <fileset dir="${src.etc.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>

```

```

    </copy>
</target>
<target name="run" depends="compile, copymetafiles"
  description="Build and run ManyToMany">
  <java fork="true"
    classname="manytomany.ManyToMany"
    classpathref="project.classpath">
    <classpath path="${build.dir}"/>
  </java>
</target>
<taskdef name="hibernatetool"
  classname="org.hibernate.tool.ant.HibernateToolTask"
  classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
  description="Exports a generated schema to DB and file">
  <hibernatetool destdir="${basedir}">
  <classpath path="${build.dir}"/>
  <annotationconfiguration
    configurationfile="${build.dir}/hibernate.cfg.xml"/>
  <hbm2ddl
    drop="true"
    create="true"
    export="true"
    outputfilename="manytomany-ddl.sql"
    delimiter=";"
    format="true"/>
  </hibernatetool>
</target>
</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="manytomany">
    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>

```

### log4j.properties

Sama dengan modul 1

### hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
      root
    </property>
    <property name="hibernate.connection.password">
      passwordmu
    </property>
    <property name="hibernate.dialect">
      org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="manytomany.Student" />
    <mapping package="manytomany" />
    <mapping class="manytomany.Subject" />
    <mapping package="manytomany" />

  </session-factory>
</hibernate-configuration>

```

## Student.java

```

package manytomany;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

```

```

@Entity
@Table(name = "STUDENTS")
public class Student {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)
    private String name;

    @Column(name = "ADDRESS", nullable=false, length=150, insertable=true)
    private String address;

    public Long getId() {
        return id;
    }

    private void setId(Long 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;
    }

    @ManyToMany(cascade=CascadeType.ALL)
    @JoinTable(name = "STUDENTSSUBJECTS", joinColumns = {
        @JoinColumn(name="student_id")
    }, inverseJoinColumns = {
        @JoinColumn(name="subject_id")
    }
    )

    private Set<Subject> subjects;

    /**
     * @return student's subjects
     */
    public Set<Subject> getSubjects() {
        return subjects;
    }
}

```

```

    }

    /**
     * @param subject the subject to set
     */
    public void setSubjects(Set<Subject> subjects) {
        this.subjects = subjects;
    }
}

```

## Subject.java

```

package manytomany;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "SUBJECTS")
public class Subject {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    public Long getId() {
        return id;
    }

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

    @Column(name = "SUBJECT", nullable=false, length=50, insertable=true)
    private String subject;

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }
}

```

```

    }

    @ManyToMany(cascade=CascadeType.ALL)
    @JoinTable(name = "STUDENTSSUBJECTS", joinColumns = {
        @JoinColumn(name="subject_id")
    }, inverseJoinColumns = {
        @JoinColumn(name="student_id")
    })

    private Set<Student> students;

    /**
     * @return the students
     */
    public Set<Student> getStudents() {
        return students;
    }

    /**
     * @param subjects the subjects to set
     */
    public void setStudents(Set<Student> students) {
        this.students = students;
    }
}

```

### ManyToMany.java

```

package manytomany;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * @author bpdp
 */
public class ManyToMany {

    /**
     * @param args
     */

    public static void main(String[] args) {

```

```

EntityManagerFactory emf=null;
EntityManager em=null;

try {

    emf=Persistence.createEntityManagerFactory("manytomany", new
HashMap());
    em=emf.createEntityManager();
    em.getTransaction().begin();

    Student student = new Student();
    student.setName("Mahasiswa 1");
    student.setAddress("Alamat mahasiswa 1");
    Student student1 = new Student();
    student1.setName("Mahasiswi 2");
    student1.setAddress("Alamat mahasiswi 2");

    HashSet studentSet = new HashSet();
    studentSet.add(student);
    studentSet.add(student1);

    Subject subject = new Subject();
    subject.setSubject("Distributed System");
    Subject subject1 = new Subject();
    subject1.setSubject("Advanced Database");

    subject.setStudents(studentSet);
    subject1.setStudents(studentSet);

    em.persist(subject);
    em.persist(subject1);

    Subject subjectRecord = em.find(Subject.class, subject.getId());
    System.out.println("Subject: "+ subjectRecord.getSubject());
    Set<Student> stdset = subjectRecord.getStudents();
    Iterator it = stdset.iterator();
    while(it.hasNext()){
        Student s = (Student)it.next();
        System.out.println("Student: "+ s.getName());
    }

    em.getTransaction().commit();
    System.out.println("Done");

} catch(Exception e) {

    System.out.println(e.getMessage());

} finally {

    em.close();
    emf.close();

}

```

```
}  
}
```

### HibernateUtil.java

Sama dengan modul 1

### **Kompilasi dan Eksekusi**

Kompilasi dengan **ant** dan eksekusi dengan menggunakan parameter **run** (**ant run**). Berikut ini adalah sebagian hasil eksekusi:

```
.....  
.....  
[java] Hibernate:  
[java]     insert  
[java]     into  
[java]         SUBJECTS  
[java]         (SUBJECT)  
[java]     values  
[java]         (?)  
[java] Hibernate:  
[java]     insert  
[java]     into  
[java]         STUDENTS  
[java]         (ADDRESS, NAME)  
[java]     values  
[java]         (?, ?)  
[java] Hibernate:  
[java]     insert  
[java]     into  
[java]         STUDENTS  
[java]         (ADDRESS, NAME)  
[java]     values  
[java]         (?, ?)  
[java] Hibernate:  
[java]     insert  
[java]     into  
[java]         SUBJECTS  
[java]         (SUBJECT)  
[java]     values  
[java]         (?)  
[java] Subject: Distributed System  
[java] Student: Mahasiswa 1  
[java] Student: Mahasiswi 2  
[java] Hibernate:  
[java]     insert  
[java]     into
```

```

[java]          STUDENTSSUBJECTS
[java]          (subject_id, student_id)
[java]          values
[java]          (?, ?)
[java] Hibernate:
[java]          insert
[java]          into
[java]          STUDENTSSUBJECTS
[java]          (subject_id, student_id)
[java]          values
[java]          (?, ?)
[java] Hibernate:
[java]          insert
[java]          into
[java]          STUDENTSSUBJECTS
[java]          (subject_id, student_id)
[java]          values
[java]          (?, ?)
[java] Hibernate:
[java]          insert
[java]          into
[java]          STUDENTSSUBJECTS
[java]          (subject_id, student_id)
[java]          values
[java]          (?, ?)
[java] Done
[java] 2719 [main] INFO org.hibernate.impl.SessionFactoryImpl -
closing
[java] 2719 [main] INFO
org.hibernate.connection.DriverManagerConnectionProvider - cleaning up
connection pool: jdbc:mysql://localhost:3306/mydb

BUILD SUCCESSFUL
Total time: 4 seconds

```

Hasil dari eksekusi tersebut terhadap data di tabel MySQL adalah sebagai berikut:

```

mysql> select * from STUDENTSSUBJECTS;
+-----+-----+
| student_id | subject_id |
+-----+-----+
|          3 |          1 |
|          4 |          1 |
|          3 |          2 |
|          4 |          2 |
+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from STUDENTS;

```

```

+----+-----+-----+
| id | name                | address                |
+----+-----+-----+
|  1 | Zaky Ahmad Aditya   | Griya Purwa Asri H-304 |
|  2 | Bambang Purnomosidi D. P. | GPA H-304             |
|  3 | Mahasiswa 1         | Alamat mahasiswa 1     |
|  4 | Mahasiswi 2        | Alamat mahasiswi 2     |
+----+-----+-----+
4 rows in set (0.00 sec)

```

```
mysql> select * from SUBJECTS;
```

```

+----+-----+
| id | subject            |
+----+-----+
|  1 | Distributed System |
|  2 | Advanced Database  |
+----+-----+
2 rows in set (0.00 sec)

```

```
mysql>
```

## 5.4 Latihan

1. Tambahkan satu field baru pada tabel STUDENTS dan SUBJECTS. Definisi, tipe, dan isi data bebas. Setelah itu, modifikasilah program untuk menyertakan hasil perubahan tersebut.

## 5.5 Tugas

Buatlah proyek baru menggunakan Hibernate JPA untuk relasi Many-to-Many. Tabel, field, dan isi data bebas dengan syarat mempunyai pola relasi Many-to-Many.

## MODUL VI

### Query dengan HQL dan JPQL

#### 6.1 Tujuan

1. Mahasiswa memahami fungsi dan kegunaan dari HQL dan JPQL
2. Mahasiswa memahami perbedaan antara HQL, JPQL, serta SQL
3. Mahasiswa memahami dasar-dasar HQL serta JPQL dalam pembuatan aplikasi menggunakan Hibernate JPA dan mampu menggunakannya untuk membuat aplikasi.

#### 6.2 Teori Singkat

Dalam pembuatan aplikasi dengan menggunakan teknologi obyek, data persistent merupakan model yang bersama-sama dengan *controller* dan *view* menjadi satu kesatuan. ORM menyediakan fasilitas berbasis teknologi obyek untuk melakukan pemrosesan terhadap model tersebut. Dalam melakukan pemrosesan tersebut, seperti halnya SQL (*Structured Query Language*), JPA menyediakan bahasa yang diperlukan untuk melakukan query yang bersifat *platform independent* (tidak tergantung pada merk / jenis software basis data relasional tertentu). Bahasa query tersebut adalah JPQL (*Java Persistence API Query Language*).

JPQL merupakan subset dari HQL (*Hibernate Query Language*). HQL merupakan bahasa query yang spesifik digunakan oleh HQL dan mempengaruhi pembuatan standar JPQL karena pengembang dari Hibernate (Gavin King) merupakan salah satu expert di tim yang merumuskan JSR 220. Untuk keperluan pengembangan aplikasi, sebaiknya menggunakan JPQL karena JPA QL ini merupakan standar yang juga diimplementasikan oleh berbagai *tools* ORM dari berbagai vendor. JPQL mendukung SELECT, *bulk* UPDATE serta DELETE. HQL dan JPQL mempunyai dialek yang sangat mirip dengan SQL sehingga memudahkan pengembang dalam mempelajari HQL dan JPQL. Berikut adalah contoh dari JPQL:

```
SELECT a FROM Author a ORDER BY a.firstName, a.lastName
```

## 6.3 Praktik

### Struktur Direktori dan Berbagai File

```
bab6/
|-- build.xml
|-- etc
|   |-- META-INF
|   |   `-- persistence.xml
|   `-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
|   |-- hibernate-entitymanager.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   `-- slf4j-simple-1.5.8.jar
`-- src
    |-- hibernate.cfg.xml
    |-- hqljpqlexample
    |   |-- Hql.java
    |   |-- Jpql.java
    |   |-- Student.java
    |   `-- Subject.java
    `-- persistence
        `-- HibernateUtil.java
```

#### build.xml

```
<project name="HqlJpqlExample" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="HqlJpqlExample"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
```

```

        <fileset dir="\${lib.dir}">
            <include name="**/*.jar"/>
            <include name="**/*.zip"/>
        </fileset>
    </path>
    <!-- Useful shortcuts -->
    <patternset id="meta.files">
        <include name="**/*.xml"/>
        <include name="**/*.properties"/>
    </patternset>
    <!-- Clean up -->
    <target name="clean">
        <delete dir="\${build.dir}"/>
        <mkdir dir="\${build.dir}"/>
    </target>
    <!-- Compile Java source -->
    <target name="compile" depends="clean">
        <mkdir dir="\${build.dir}"/>
        <javac
            srcdir="\${src.java.dir}"
            destdir="\${build.dir}"
            nowarn="on">
            <classpath refid="project.classpath"/>
        </javac>
    </target>
    <!-- Copy metadata to build classpath -->
    <property name="src.etc.dir" value="etc"/>
    <target name="copymetafiles">
        <copy todir="\${build.dir}">
            <fileset dir="\${src.java.dir}">
                <patternset refid="meta.files"/>
            </fileset>
        </copy>
        <!-- Copy configuration files from etc/ -->
        <copy todir="\${build.dir}">
            <fileset dir="\${src.etc.dir}">
                <patternset refid="meta.files"/>
            </fileset>
        </copy>
    </target>

    <target name="runhql" depends="compile, copymetafiles"
        description="Build and run HQL">
        <java fork="true"
            classname="hqljpqlexample.Hql"
            classpathref="project.classpath">
            <classpath path="\${build.dir}"/>
        </java>
    </target>
    <target name="runjpql" depends="compile, copymetafiles"
        description="Build and run JPQL">
        <java fork="true"
            classname="hqljpqlexample.Jpql"
            classpathref="project.classpath">

```

```

        <classpath path="\${build.dir}"/>
    </java>
</target>

<taskdef name="hibernatetool"
    classname="org.hibernate.tool.ant.HibernateToolTask"
    classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
    description="Exports a generated schema to DB and file">
    <hibernatetool destdir="\${basedir}">
    <classpath path="\${build.dir}"/>
    <annotationconfiguration
        configurationfile="\${build.dir}/hibernate.cfg.xml"/>
    <hbm2ddl
        drop="true"
        create="true"
        export="true"
        outputfilename="hqljppqlexample-ddl.sql"
        delimiter=";"
        format="true"/>
    </hibernatetool>
</target>

</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="hqljppqlexample">
        <properties>
            <property name="hibernate.ejb.cfgfile"
                value="/hibernate.cfg.xml"/>
        </properties>
    </persistence-unit>
</persistence>

```

### log4j.properties

Sama dengan modul 1

### hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">

```

```

        com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
        jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
        root
    </property>
    <property name="hibernate.connection.password">
        passwordmu
    </property>
    <property name="hibernate.dialect">
        org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="hqljppqlexample.Student" />
    <mapping package="hqljppqlexample" />
    <mapping class="hqljppqlexample.Subject" />
    <mapping package="hqljppqlexample" />

</session-factory>
</hibernate-configuration>

```

## Student.java

```

package hqljppqlexample;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENTS")
public class Student {

```

```

@Id @GeneratedValue(strategy=GenerationType.AUTO)
@Column(name = "ID")
private Long id;

@Column(name = "NAME", nullable=false, length=50, insertable=true)
private String name;

@Column(name = "ADDRESS", nullable=false, length=150, insertable=true)
private String address;

public Long getId() {
    return id;
}

private void setId(Long 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;
}

@ManyToMany(cascade=CascadeType.ALL)
@JoinTable(name = "STUDENTSSUBJECTS", joinColumns = {
    @JoinColumn(name="student_id")
}, inverseJoinColumns = {
    @JoinColumn(name="subject_id")
}
)

private Set<Subject> subjects;

/**
 * @return student's subjects
 */
public Set<Subject> getSubjects() {
    return subjects;
}

/**
 * @param subject the subject to set
 */

```

```

    public void setSubjects(Set<Subject> subjects) {
        this.subjects = subjects;
    }
}

```

### Subject.java

```

package hqljppqlexample;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "SUBJECTS")
public class Subject {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    public Long getId() {
        return id;
    }

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

    @Column(name = "SUBJECT", nullable=false, length=50, insertable=true)
    private String subject;

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    @ManyToMany(cascade=CascadeType.ALL)
    @JoinTable(name = "STUDENTSSUBJECTS", joinColumns = {
        @JoinColumn(name="subject_id")
    }

```

```

        }, inverseJoinColumn = {
            @JoinColumn(name="student_id")
        }
    )

    private Set<Student> students;

    /**
     * @return the students
     */
    public Set<Student> getStudents() {
        return students;
    }

    /**
     * @param subjects the subjects to set
     */
    public void setStudents(Set<Student> students) {
        this.students = students;
    }
}

```

### Hql.java

```

package hqljpqlexample;

// untuk HQL
import org.hibernate.Session;
import org.hibernate.*;
import org.hibernate.cfg.*;

// untuk mengakses hasil
import java.util.*;

/**
 * @author bpdp
 *
 */

public class Hql {

    public static void main(String[] args) {

        Session session = null;

        try {

            SessionFactory sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
            session = sessionFactory.openSession();

            System.out.println("Hibernate Query Language (HQL) example");

```

```

String HQL_QUERY ="from Student as s where s.id=1";
Query query = session.createQuery(HQL_QUERY);

for(Iterator it=query.iterate();it.hasNext();) {
    Student student = (Student)it.next();
    System.out.println("ID: " + student.getId());
    System.out.println("Name: " + student.getName());
    System.out.println("Address: " + student.getAddress());
}

session.close();

} catch (Exception e) {
    System.out.println(e.getMessage());
} finally {
}
}
}

```

### Jpql.java

```

package hqljqpqlexample;

// untuk JPQL
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

// untuk mengakses hasil
import java.util.Iterator;
import java.util.List;

/**
 * @author bpdp
 *
 */

public class Jpql {

    public static void main(String[] args) {

        System.out.println("Java Persistence Query Language (JPQL) example");
        EntityManagerFactory emf =
        Persistence.createEntityManagerFactory("hqljqpqlexample");
        EntityManager em=emf.createEntityManager();

        try {
            EntityTransaction entr = em.getTransaction();
            entr.begin();
            Query JPQL_QUERY = em.createQuery("SELECT st FROM Student st

```

```

WHERE st.name LIKE :stuName");
JPQL_QUERY.setParameter("stuName", "Zaky Ahmad Aditya");

List result = JPQL_QUERY.getResultList();
if (result.size() != 0) {
    Iterator stIterator = result.iterator();
    while (stIterator.hasNext()) {
        Student st = (Student) stIterator.next();
        System.out.print("Name: " + st.getName());
        System.out.println();
        System.out.print("Address: " + st.getAddress());
        System.out.println();
    }
} else {
    System.out.println("No record found");
}

entr.commit();

} finally {

    em.close();

}

}
}

```

### HibernateUtil.java

Sama dengan modul 1

### Kompilasi dan Eksekusi Hasil Kompilasi

Untuk mengkompilasi, gunakan **ant**, sedangkan untuk menjalankan hasil kompilasi gunakan **ant runhql** (untuk menjalankan praktik HQL) serta **ant runjpql** (untuk menjalankan praktik JP QL). Berikut ini adalah sebagian hasil dari masing-masing praktik:

#### HQL

```

.....
.....
[java] Hibernate Query Language (HQL) example
[java] Hibernate:
[java]      select
[java]          student0_.ID as col_0_0_
[java]      from
[java]          STUDENTS student0_
[java]      where

```

```

[java]         student0_.ID=1
[java] Hibernate:
[java]         select
[java]         student0_.ID as ID0_0_,
[java]         student0_.ADDRESS as ADDRESS0_0_,
[java]         student0_.NAME as NAME0_0_
[java]         from
[java]         STUDENTS student0_
[java]         where
[java]         student0_.ID=?
[java] ID: 1
[java] Name: Zaky Ahmad Aditya
[java] Address: Griya Purwa Asri H-304

```

BUILD SUCCESSFUL  
Total time: 3 seconds

## JPQL

```

.....
.....
[java] Hibernate:
[java]         select
[java]         student0_.ID as ID0_,
[java]         student0_.ADDRESS as ADDRESS0_,
[java]         student0_.NAME as NAME0_
[java]         from
[java]         STUDENTS student0_
[java]         where
[java]         student0_.NAME like ?
[java] Name: Zaky Ahmad Aditya
[java] Address: Griya Purwa Asri H-304

```

BUILD SUCCESSFUL  
Total time: 3 seconds

## 6.4 Latihan

1. Modifikasilah program-program di atas untuk melakukan query ke *STUDENTS* berdasarkan pada alamat dari mahasiswa.

## 6.5 Tugas

Buatlah proyek baru menggunakan Hibernate JPA untuk melakukan query terhadap suatu tabel dengan 2 parameter query *where*. Definisi tabel, tipe data, serta isi tabel bebas. Gunakan JPQL saja.

## MODUL VII

### JPA Named Query

#### 7.1 Tujuan

1. Mahasiswa memahami pengertian JPA Named Query dan memahami fungsi dan kegunaannya.
2. Mahasiswa memahami cara menggunakan JPA Named Query dalam suatu aplikasi JPA.

#### 7.2 Teori Singkat

Query yang dilakukan terhadap data persistent bisa dituliskan langsung pada setiap bagian program yang memerlukan query. Meskipun demikian, JPA memungkinkan pemrogram untuk membuat *Named Query* yaitu query yang telah didefinisikan lebih dulu. Query tersebut juga memungkinkan untuk menerima parameter, jadi pemrogram bisa mendefinisikan terlebih dahulu dan kemudian baru meng-assign parameter / argumen terhadap query tersebut pada badan program.

*Named Query* pada JPA diimplementasikan menggunakan Annotation `@javax.persistence.NamedQuery` dan didefinisikan pada class di Java yang mendefinisikan model tersebut. Parameter didefinisikan dengan menggunakan **?** **angka** dengan angka menunjukkan parameter ke berapa yang dikehendaki. Berikut adalah contoh dari *Named Query*:

```
@javax.persistence.NamedQuery(name="studentByName", query="SELECT st  
FROM Student st WHERE st.name= ?1")
```

Untuk menggunakan *Named Query* tersebut, buat query pada badan program sebagai berikut:

```
Query query=em.createNamedQuery("studentByName");  
query.setParameter(1, "Bambang Purnomosidi D. P.");
```

## 7.3 Praktik

### Struktur Direktori dan Berbagai File

```
bab7/
|-- build.xml
|-- etc
|   |-- META-INF
|   |   |-- persistence.xml
|   |-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
|   |-- hibernate-entitymanager.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   |-- slf4j-simple-1.5.8.jar
`-- src
    |-- hibernate.cfg.xml
    |-- jpanamedquery
    |   |-- JpaNamedQuery.java
    |   |-- Student.java
    |-- persistence
    |-- HibernateUtil.java
```

#### build.xml

```
<project name="JPA Named Query" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="JPA Named Query"/>
  <property name="proj.version" value="1.0"/>
  <!-- Global properties for this build -->
  <property name="src.java.dir" value="src"/>
  <property name="lib.dir" value="lib"/>
  <property name="build.dir" value="bin"/>
  <!-- Classpath declaration -->
  <path id="project.classpath">
    <fileset dir="${lib.dir}">
      <include name="**/*.jar"/>
    </fileset>
  </path>
</project>
```

```

        <include name="**/*.zip"/>
    </fileset>
</path>
<!-- Useful shortcuts -->
<patternset id="meta.files">
    <include name="**/*.xml"/>
    <include name="**/*.properties"/>
</patternset>
<!-- Clean up -->
<target name="clean">
    <delete dir="${build.dir}"/>
    <mkdir dir="${build.dir}"/>
</target>
<!-- Compile Java source -->
<target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac
        srcdir="${src.java.dir}"
        destdir="${build.dir}"
        nowarn="on">
        <classpath refid="project.classpath"/>
    </javac>
</target>
<!-- Copy metadata to build classpath -->
<property name="src.etc.dir" value="etc"/>
<target name="copymetafiles">
    <copy todir="${build.dir}">
        <fileset dir="${src.java.dir}">
            <patternset refid="meta.files"/>
        </fileset>
    </copy>
    <!-- Copy configuration files from etc/ -->
    <copy todir="${build.dir}">
        <fileset dir="${src.etc.dir}">
            <patternset refid="meta.files"/>
        </fileset>
    </copy>
</target>
<target name="run" depends="compile, copymetafiles"
    description="Build and run JPA Named Query">
    <java fork="true"
        classname="jpanamedquery.JpaNamedQuery"
        classpathref="project.classpath">
        <classpath path="${build.dir}"/>
    </java>
</target>
<taskdef name="hibernatetool"
    classname="org.hibernate.tool.ant.HibernateToolTask"
    classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
    description="Exports a generated schema to DB and file">
    <hibernatetool destdir="${basedir}">
    <classpath path="${build.dir}"/>
    <annotationconfiguration

```

```

        configurationfile="${build.dir}/hibernate.cfg.xml"/>
    <hbm2ddl
        drop="true"
        create="true"
        export="true"
        outputfilename="jpanamedquery-ddl.sql"
        delimiter=";"
        format="true"/>
    </hibernatetool>
</target>
</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="jpanamedquery">
        <properties>
            <property name="hibernate.ejb.cfgfile"
                value="/hibernate.cfg.xml"/>
        </properties>
    </persistence-unit>
</persistence>

```

### log4j.properties

Sama dengan modul 1

### hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/mydb
        </property>
        <property name="hibernate.connection.username">
            root
        </property>
        <property name="hibernate.connection.password">
            passwordmu
        </property>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQL5Dialect

```

```

    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="jpanamedquery.Student" />
    <mapping package="jpanamedquery" />

</session-factory>
</hibernate-configuration>

```

## Student.java

```

package jpanamedquery;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENTS")
@NamedQuery(name="studentByName", query="SELECT st
FROM Student st WHERE st.name= ?1")

public class Student {

    @Id @GeneratedValue
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)
    private String name;

    @Column(name = "ADDRESS", nullable=false, length=150, insertable=true)
    private String address;

    public Long getId() {
        return id;
    }

    private void setId(Long 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;
}
}

```

### JpaNamedQuery.java

```

package jpanamedquery;

import java.util.Iterator;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

/**
 * @author bpdp
 *
 */
public class JpaNamedQuery {

    /**
     * @param args
     */

    public static void main(String[] args) {

        EntityManagerFactory emf
Persistence.createEntityManagerFactory("jpanamedquery");
        EntityManager em=emf.createEntityManager();

        try {

            EntityTransaction entr=em.getTransaction();

```

```

entr.begin();

Query query=em.createNamedQuery("studentByName");
query.setParameter(1, "Bambang Purnomosidi D. P.");

List stList=query.getResultList();
Iterator stIterator=stList.iterator();

while(stIterator.hasNext()){
    Student st=(Student)stIterator.next();
    System.out.println("Student ID: " + st.getId());
    System.out.println(" Name: " + st.getName());
    System.out.println(" Address: " + st.getAddress());
    System.out.println();
}
entr.commit();
} finally {
    em.close();
}
}
}

```

### HibernateUtil.java

Sama dengan modul 1.

### **Kompilasi dan Eksekusi Hasil Kompilasi**

Untuk mengkompilasi, gunakan **ant**. Untuk menjalankan hasil kompilasi, gunakan **ant run**. Berikut adalah sebagian hasil dari proses saat menjalankan hasil kompilasi:

```

.....
.....
[java] Hibernate:
[java]      select
[java]          student0_.ID as ID0_,
[java]          student0_.ADDRESS as ADDRESS0_,
[java]          student0_.NAME as NAME0_
[java]      from
[java]          STUDENTS student0_
[java]      where
[java]          student0_.NAME=?
[java] Student ID: 2
[java] Name: Bambang Purnomosidi D. P.
[java] Address: GPA H-304
[java]

```

BUILD SUCCESSFUL  
Total time: 3 seconds

#### **7.4 Latihan**

1. Buat satu definisi JPA *Named Query* lagi pada model Student dengan menggunakan 2 parameter.

#### **7.5 Tugas**

Buat suatu proyek Hibernate JPA baru. Buat satu model berdasarkan pada satu tabel basis data dengan tipe data dan isi bebas. Buat *JPA Named Query* dengan dua parameter untuk model tersebut dan buat juga program yang menggunakan *Named Query* tersebut.

## MODUL VIII

### Native SQL

#### 8.1 Tujuan

1. Mahasiswa memahami berbagai kondisi yang menyebabkan perlunya mengakses native SQL.
2. Mahasiswa memahami cara menggunakan native SQL dalam aplikasi JPA dan mampu mengaplikasikannya pada pembuatan aplikasi menggunakan JPA.

#### 8.2 Teori Singkat

SQL (*Structured Query Language*) merupakan bahasa standar yang digunakan dalam pengaksesan basis data relasional. Komite yang menangani standar ini disebut dengan ANSI (American National Standard Institute). Saat ini kebanyakan implementasi SQL mengarah pada SQL 92 meskipun sudah ada SQL 99 (92 menunjukkan tahun pembuatan – 1992, 99 menunjukkan tahun 1999).

Meskipun merupakan standar, biasanya berbagai vendor mempunyai pengecualian-kecualian tertentu dari standar ini, sehingga kemungkinan skrip SQL yang dibuat menjadi tidak *portable*. Hal ini juga yang mendasari dibangunnya *tools* ORM.

Meskipun terdapat standar dan ada beberapa penyimpangan dari standar, seorang pemrogram kadang kala masih memerlukan akses ke SQL secara langsung, misalnya untuk mengakses fitur spesifik dari SQL basis data dari vendor yang bersangkutan (*trigger, function, stored procedure*, dan sejenisnya). Dengan menggunakan native SQL ini, ada kemungkinan aplikasi yang dibuat oleh pemrogram menjadi tidak *portable*. Sebaiknya berhati-hati menggunakan fitur ini. Gunakan fitur ini jika hanya terpaksa dan gunakan sesedikit mungkin dari fitur ini.

Untuk menggunakan native SQL dalam aplikasi JPA, gunakan method `CreateSQLQuery` dari `sessionFactory`. Berikut ini adalah contoh kecil dari native

SQL:

```
.....  
.....  
String sql ="select stddev(vst.visit) as stdErr, "+  
            " avg(vst.visit) as mean from VISITS vst";  
  
Query query = session.createSQLQuery(sql)  
                .ddScalar("mean",Hibernate.DOUBLE)  
                .addScalar("stdErr",Hibernate.DOUBLE);  
  
.....  
.....
```

### 8.3 Praktik

#### Persiapan Tabel MySQL dan Data

Untuk keperluan praktik ini, akan digunakan tabel VISITS yang berisi data kunjungan dari konsumen. Berikut adalah skrip SQL untuk definisi tabel serta memasukkan data ke dalam tabel tersebut:

```
mysql> create table VISITS (id bigint(20), name varchar(50), visit int,  
primary key(id)) Type=InnoDB;  
Query OK, 0 rows affected, 1 warning (0.06 sec)  
  
mysql> insert into VISITS values (1, "Visitor number 1", 20);  
Query OK, 1 row affected (0.01 sec)  
  
mysql> insert into VISITS values (2, "Visitor number 2", 19);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into VISITS values (3, "Visitor number 3", 12);  
Query OK, 1 row affected (0.01 sec)  
  
mysql> insert into VISITS values (4, "Visitor number 4", 17);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into VISITS values (5, "Visitor number 5", 21);  
Query OK, 1 row affected (0.01 sec)  
  
mysql> insert into VISITS values (6, "Visitor number 6", 10);  
Query OK, 1 row affected (0.01 sec)  
  
mysql> select * from VISITS;  
+-----+-----+-----+  
| id | name          | visit |  
+-----+-----+-----+  
| 1 | Visitor number 1 | 20 |  
| 2 | Visitor number 2 | 19 |
```

```

| 3 | Visitor number 3 | 12 |
| 4 | Visitor number 4 | 17 |
| 5 | Visitor number 5 | 21 |
| 6 | Visitor number 6 | 10 |
+----+-----+-----+
6 rows in set (0.00 sec)

```

mysql>

## **Struktur Direktori dan Berbagai File**

```

bab8/
|-- build.xml
|-- etc
|   |-- META-INF
|   |   |-- persistence.xml
|   |-- log4j.properties
|-- lib
|   |-- antlr-2.7.6.jar
|   |-- asm-attrs.jar
|   |-- asm.jar
|   |-- c3p0-0.9.1.jar
|   |-- cglib-2.2.jar
|   |-- commons-collections-3.1.jar
|   |-- commons-logging-1.1.1.jar
|   |-- dom4j-1.6.1.jar
|   |-- ejb3-persistence.jar
|   |-- freemarker.jar
|   |-- hibernate-annotations.jar
|   |-- hibernate-commons-annotations.jar
|   |-- hibernate-core.jar
|   |-- hibernate-entitymanager.jar
|   |-- hibernate-tools.jar
|   |-- hibernate3.jar
|   |-- javassist-3.9.0.GA.jar
|   |-- jta-1.1.jar
|   |-- mysql-connector-java-5.1.8-bin.jar
|   |-- slf4j-api-1.5.8.jar
|   |-- slf4j-simple-1.5.8.jar
`-- src
    |-- hibernate.cfg.xml
    |-- nativesql
    |   |-- NativeSql.java
    |   |-- Visit.java
    |-- persistence
    |   |-- HibernateUtil.java

```

### **build.xml**

```

<project name="NativeSQLExample" default="compile" basedir=".">
  <!-- Name of project and version -->
  <property name="proj.name" value="NativeSQLExample"/>

```

```

<property name="proj.version" value="1.0"/>
<!-- Global properties for this build -->
<property name="src.java.dir" value="src"/>
<property name="lib.dir" value="lib"/>
<property name="build.dir" value="bin"/>
<!-- Classpath declaration -->
<path id="project.classpath">
  <fileset dir="${lib.dir}">
    <include name="**/*.jar"/>
    <include name="**/*.zip"/>
  </fileset>
</path>
<!-- Useful shortcuts -->
<patternset id="meta.files">
  <include name="**/*.xml"/>
  <include name="**/*.properties"/>
</patternset>
<!-- Clean up -->
<target name="clean">
  <delete dir="${build.dir}"/>
  <mkdir dir="${build.dir}"/>
</target>
<!-- Compile Java source -->
<target name="compile" depends="clean">
  <mkdir dir="${build.dir}"/>
  <javac
    srcdir="${src.java.dir}"
    destdir="${build.dir}"
    nowarn="on">
    <classpath refid="project.classpath"/>
  </javac>
</target>
<!-- Copy metadata to build classpath -->
<property name="src.etc.dir" value="etc"/>
<target name="copymetafiles">
  <copy todir="${build.dir}">
    <fileset dir="${src.java.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
  <!-- Copy configuration files from etc/ -->
  <copy todir="${build.dir}">
    <fileset dir="${src.etc.dir}">
      <patternset refid="meta.files"/>
    </fileset>
  </copy>
</target>
<target name="run" depends="compile, copymetafiles"
  description="Build and run Native SQL">
  <java fork="true"
    classname="nativesql.NativeSql"
    classpathref="project.classpath">
    <classpath path="${build.dir}"/>
  </java>

```

```

</target>
<taskdef name="hibernatetool"
  classname="org.hibernate.tool.ant.HibernateToolTask"
  classpathref="project.classpath"/>
<target name="schemaexport" depends="compile, copymetafiles"
  description="Exports a generated schema to DB and file">
  <hibernatetool destdir="${basedir}">
  <classpath path="${build.dir}"/>
  <annotationconfiguration
    configurationfile="${build.dir}/hibernate.cfg.xml"/>
  <hbm2ddl
    drop="true"
    create="true"
    export="true"
    outputfilename="nativesql-ddl.sql"
    delimiter=";"
    format="true"/>
  </hibernatetool>
</target>

</project>

```

### persistence.xml

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="nativesql">
    <properties>
      <property name="hibernate.ejb.cfgfile"
        value="/hibernate.cfg.xml"/>
    </properties>
  </persistence-unit>
</persistence>

```

### log4j.properties

Sama seperti modul 1

### hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration SYSTEM
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">

```

```

        jdbc:mysql://localhost:3306/mydb
    </property>
    <property name="hibernate.connection.username">
        root
    </property>
    <property name="hibernate.connection.password">
        passwordmu
    </property>
    <property name="hibernate.dialect">
        org.hibernate.dialect.MySQL5Dialect
    </property>
    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>
    <!-- Show and print nice SQL on stdout -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>

    <mapping class="nativesql.Visit" />
    <mapping package="nativesql" />

</session-factory>
</hibernate-configuration>

```

## Visit.java

```

package nativesql;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "VISITS")
public class Visit {

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "ID")
    private Long id;

    @Column(name = "NAME", nullable=false, length=50, insertable=true)

```

```

private String name;

@Column(name = "VISIT", nullable=false)
private Integer visit;

public Long getId() {
    return id;
}

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

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Integer getVisit() {
    return visit;
}

public void setVisit(Integer visit) {
    this.visit = visit;
}
}

```

### NativeSql.java

```

package nativesql;

import org.hibernate.Session;
import org.hibernate.*;
import org.hibernate.criterion.*;
import org.hibernate.cfg.*;
import java.util.*;

/**
 * @author bpdp
 *
 */

public class NativeSql {

    public static void main(String[] args) {

        Session session = null;

        try {

```

```

SessionFactory sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
session = sessionFactory.openSession();

System.out.println("Native SQL");

String sql ="select stddev(vst.visit) as stdErr, "+
" avg(vst.visit) as mean from VISITS vst";

Query query =
session.createQuery(sql).addScalar("mean",Hibernate.DOUBLE).addScala
r("stdErr",Hibernate.DOUBLE);

Object [] amount = (Object [])
query.uniqueResult();

System.out.println("Rata-rata kunjungan: " + amount[0]);
System.out.println("Standar deviasi: " + amount[1]);

List visit = session.createQuery("select {vst.*} from VISITS
vst").addEntity("vst", Visit.class).list();

for (Iterator it = visit.iterator(); it.hasNext();) {
    Visit visitObject = (Visit) it.next();
    System.out.println("Visitor ID: " + visitObject.getId());
    System.out.println("Name: " + visitObject.getName());
    System.out.println("Visit: " + visitObject.getVisit());
}

session.close();

} catch (Exception e) {
    System.out.println(e.getMessage());
} finally {
}
}
}

```

### HibernateUtil.java

Sama seperti modul 1

### **Kompilasi dan Eksekusi Hasil Kompilasi**

Kompilasi dilakukan dengan perintah **ant**, setelah itu eksekusi hasil kompilasi menggunakan **ant run**. Berikut adalah sebagian hasil dari eksekusi hasil kompilasi:

```

.....
.....
[java] Native SQL
[java] Hibernate:
[java]      select
[java]          stddev(vst.visit) as stdErr,
[java]          avg(vst.visit) as mean
[java]      from
[java]          VISITS vst
[java] Rata-rata kunjungan: 16.5
[java] Standar deviasi: 4.113
[java] Hibernate:
[java]      select
[java]          vst.ID as ID0_0_,
[java]          vst.NAME as NAME0_0_,
[java]          vst.VISIT as VISIT0_0_
[java]      from
[java]          VISITS vst
[java] Visitor ID: 1
[java] Name: Visitor number 1
[java] Visit: 20
[java] Visitor ID: 2
[java] Name: Visitor number 2
[java] Visit: 19
[java] Visitor ID: 3
[java] Name: Visitor number 3
[java] Visit: 12
[java] Visitor ID: 4
[java] Name: Visitor number 4
[java] Visit: 17
[java] Visitor ID: 5
[java] Name: Visitor number 5
[java] Visit: 21
[java] Visitor ID: 6
[java] Name: Visitor number 6
[java] Visit: 10

```

```

BUILD SUCCESSFUL
Total time: 2 seconds

```

Hasil dari eksekusi native SQL pada program di atas sesuai dengan hasil yang dikerjakan langsung pada prompt SQL berikut ini:

```

mysql> select avg(visit) from VISITS;
+-----+
| avg(visit) |
+-----+
| 16.5000 |
+-----+
1 row in set (0.00 sec)

```

```
mysql> select stddev(visit) from VISITS;
+-----+
| stddev(visit) |
+-----+
|          4.1130 |
+-----+
1 row in set (0.00 sec)

mysql>
```

#### 8.4 Latihan

1. Modifikasilah program-program di atas untuk menghasilkan konsumen dengan kunjungan yang terbanyak (MAX) serta konsumen dengan kunjungan paling sedikit (MIN).

#### 8.5 Tugas

Buat proyek Hibernate JPA baru untuk mengakses suatu *function* yang telah anda definisikan pada basis data MySQL.