JPA - Entity Relationships

This chapter takes you through the relationships between Entities. Generally the relations are more effective between tables in the database. Here the entity classes are treated as relational tables (concept of JPA), therefore the relationships between Entity classes are as follows:

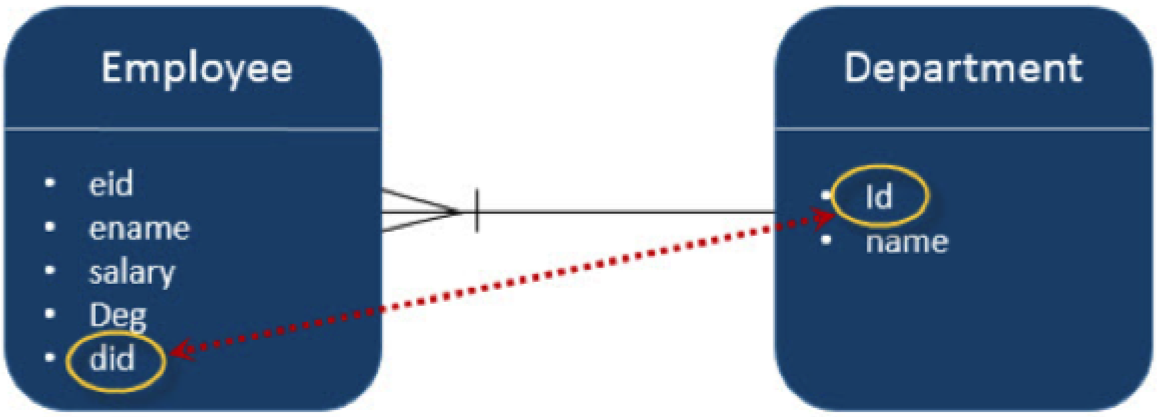
- @ManyToOne Relation
- @OneToMany Relation
- @OneToOne Relation
- @ManyToMany Relation

@ManyToOne Relation

Many-To-One relation between entities: Where one entity (column or set of columns) is/are referenced with another entity (column or set of columns) which contain unique values. In relational databases these relations are applicable by using foreign key/primary key between tables.

Let us consider an example of relation between Employee and Department entities. In unidirectional manner, i.e.from Employee to Department, Many-To-One relation is applicable. That means each record of employee contains one department id, which should be a primary key in Department table. Here in the Employee table, Department id is foreign Key.

The diagram explains Many-To-One relation as follows:



Create a JPA project in eclipse IDE named **JPA_Eclipselink_MTO**. All the modules of this project are shown as follows:

Creating Entities

Follow the above given diagram for creating entities. Create a package named **'com.tutorialspoin.eclipselink.entity'** under **'src'** package. Create a class named **Department.java** under given package. The class Department entity is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

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

@Entity
public class Department {

   @Id
   @GeneratedValue( strategy=GenerationType.AUTO )

   private int id;
   private String name;

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName( ){
      return name;
   }

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

Create the second entity in this relation - Employee entity class named **Employee.java** under **'com.tutorialspoint.eclipselink.entity'** package. The Employee entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Employee{

   @Id
   @GeneratedValue( strategy= GenerationType.AUTO )

   private int eid;
   private String ename;
   private double salary;
   private String deg;

   @ManyToOne
   private Department department;

   public Employee(int eid, String ename, double salary, String deg) {
      super( );
      this.eid = eid;
      this.ename = ename;
      this.salary = salary;
      this.deg = deg;
   }

   public Employee( ) {
      super();
   }

   public int getEid( ) {
      return eid;
   }

   public void setEid(int eid)  {
      this.eid = eid;
   }

   public String getEname( ) {
      return ename;
   }

   public void setEname(String ename) {
      this.ename = ename;
   }

   public double getSalary( ) {
      return salary;
   }

   public void setSalary(double salary) {
      this.salary = salary;
   }

   public String getDeg( ) {
      return deg;
   }

   public void setDeg(String deg) {
      this.deg = deg;
   }

   public Department getDepartment() {
      return department;
   }

   public void setDepartment(Department department) {
      this.department = department;
   }
}
```

Persistence.xml

Persistence.xml file is required to configure the database and the registration of entity classes.

Persitence.xml will be created by the eclipse IDE while creating a JPA Project. The configuration details are user specifications. The persistence.xml file is shown as follows:

```
<?xml version="1.0" encoding = "UTF-8"?>

<persistence version = "2.0"
   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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_
      <class>com.tutorialspoint.eclipselink.entity.Employee</class>
      <class>com.tutorialspoint.eclipselink.entity.Department</class>

      <properties>
         <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql
         <property name = "javax.persistence.jdbc.user" value = "root"/>
         <property name = "javax.persistence.jdbc.password" value="root"/>
         <property name = "javax.persistence.jdbc.driver" value="com.mysql.
         <property name = "eclipselink.logging.level" value = "FINE"/>
         <property name = "eclipselink.ddl-generation" value = "create-tabl
      </properties>

   </persistence-unit>
</persistence>
```

Service Classes

This module contains the service classes, which implements the relational part using the attribute initialization. Create a package under **‘src’** package named **‘com.tutorialspoint.eclipselink.service’**. The DAO class named **ManyToOne.java** is created under given package. The DAO class is shown as follows:

```
package com.tutorialspointeclipselink.service;

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

import com.tutorialspoint.eclipselink.entity.Department;
import com.tutorialspoint.eclipselink.entity.Employee;

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

   EntityManagerFactory emfactory = Persistence.createEntityManagerFactory(
   EntityManager entitymanager = emfactory.createEntityManager( );
   entitymanager.getTransaction( ).begin( );

   //Create Department Entity
   Department department = new Department();
   department.setName("Development");

   //Store Department
   entitymanager.persist(department);

   //Create Employee1 Entity
   Employee employee1 = new Employee();
   employee1.setEname("Satish");
   employee1.setSalary(45000.0);
   employee1.setDeg("Technical Writer");
```

```
      employee1.setDepartment(department);

      //Create Employee2 Entity
      Employee employee2 = new Employee();
      employee2.setEname("Krishna");
      employee2.setSalary(45000.0);
      employee2.setDeg("Technical Writer");
      employee2.setDepartment(department);

      //Create Employee3 Entity
      Employee employee3 = new Employee();
      employee3.setEname("Masthanvali");
      employee3.setSalary(50000.0);
      employee3.setDeg("Technical Writer");
      employee3.setDepartment(department);

      //Store Employees
      entitymanager.persist(employee1);
      entitymanager.persist(employee2);
      entitymanager.persist(employee3);

      entitymanager.getTransaction().commit();
      entitymanager.close();
      emfactory.close();
      }
   }
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output, check MySQL workbench. In this example two tables are created.

Pass the following query in MySQL interface and the result of **Department** table in a tabular format is shown as follows in the query:

```
Select * from department;

Id      Name
101     Development
```

Pass the following query in MySQL interface and the result of **Employee** table in a tabular format is shown as follows in the query:

```
Select * from employee;

Eid Deg                     Ename           Salary   Department_Id
```

```
102 Technical Writer    Satish          45000   101
103 Technical Writer    Krishna         45000   101
104 Technical Writer    Masthan Wali    50000   101
```

In the above table Deparment_Id is the foreign key (reference field) from Department table.

@OneToMany Relation

In this relationship each row of one entity is referenced to many child records in other entity. The important thing is that child records cannot have multiple parents. In a one-to-many relationship between Table A and Table B, each row in Table A is linked to 0, 1 or many rows in Table B.

Let us consider the above example. If **Employee** and **Department** is in a reverse unidirectional manner, relation is Many-To-One relation. Create a JPA project in eclipse IDE named **JPA_Eclipselink_OTM**. All the modules of this project are shown as follows:

Creating Entities

Follow the above given diagram for creating entities. Create a package named **'com.tutorialspoin.eclipselink.entity'** under **'src'** package. Create a class named **Department.java** under given package. The class Department entity is shown as follows:

```java
package com.tutorialspoint.eclipselink.entity;

import java.util.List;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Department {

    @Id
    @GeneratedValue( strategy=GenerationType.AUTO )

    private int id;
    private String name;

    @OneToMany( targetEntity=Employee.class )
    private List employeelist;

    public int getId() {
        return id;
    }
```

```
      public void setId(int id) {
          this.id = id;
      }

      public String getName( ) {
          return name;
      }

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

      public List getEmployeelist() {
        return employeelist;
      }

     public void setEmployeelist(List employeelist) {
        this.employeelist = employeelist;
     }
  }
```

Create the second entity in this relation -Employee entity class, named **Employee.java** under **'com.tutorialspoint.eclipselink.entity'** package. The Employee entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

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

@Entity
public class Employee {

   @Id
   @GeneratedValue( strategy= GenerationType.AUTO )

   private int eid;
   private String ename;
   private double salary;
   private String deg;
   public Employee(int eid, String ename, double salary, String deg) {
      super( );
      this.eid = eid;
      this.ename = ename;
      this.salary = salary;
      this.deg = deg;
   }

   public Employee( ) {
      super();
   }

   public int getEid( ) {
      return eid;
   }

   public void setEid(int eid) {
      this.eid = eid;
   }

   public String getEname( ) {
      return ename;
   }

   public void setEname(String ename) {
      this.ename = ename;
   }

   public double getSalary( ) {
      return salary;
   }

   public void setSalary(double salary) {
      this.salary = salary;
   }

   public String getDeg( ) {
      return deg;
   }

   public void setDeg(String deg) {
      this.deg = deg;
   }
}
```

Persistence.xml

Persistence.xml will be created by the eclipse IDE while creating a JPA Project. The configuration details are user specifications. The persistence.xml file is shown as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>

<persistence version = "2.0" xmlns = "http://java.sun.com/xml/ns/persistenc
   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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_
      <class>com.tutorialspoint.eclipselink.entity.Employee</class>
      <class>com.tutorialspoint.eclipselink.entity.Department</class>

      <properties>
         <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql:
         <property name = "javax.persistence.jdbc.user" value = "root"/>
         <property name = "javax.persistence.jdbc.password" value = "root",
         <property name = "javax.persistence.jdbc.driver" value = "com.mysc
         <property name = "eclipselink.logging.level" value = "FINE"/>
         <property name = "eclipselink.ddl-generation" value = "create-tabl
      </properties>

   </persistence-unit>
</persistence>
```

Service Classes

This module contains the service classes, which implements the relational part using the attribute initialization. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**. The DAO class named **OneToMany.java** is created under given package. The DAO class is shown as follows:

```
package com.tutorialspointeclipselink.service;

import java.util.List;
import java.util.ArrayList;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import com.tutorialspoint.eclipselink.entity.Department;
import com.tutorialspoint.eclipselink.entity.Employee;

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

   EntityManagerFactory emfactory = Persistence.createEntityManagerFactory
   EntityManager entitymanager = emfactory.createEntityManager( );
   entitymanager.getTransaction( ).begin( );

   //Create Employee1 Entity
   Employee employee1 = new Employee();
   employee1.setEname("Satish");
   employee1.setSalary(45000.0);
   employee1.setDeg("Technical Writer");

   //Create Employee2 Entity
   Employee employee2 = new Employee();
   employee2.setEname("Krishna");
   employee2.setSalary(45000.0);
   employee2.setDeg("Technical Writer");

   //Create Employee3 Entity
   Employee employee3 = new Employee();
   employee3.setEname("Masthanvali");
   employee3.setSalary(50000.0);
   employee3.setDeg("Technical Writer");

   //Store Employee
   entitymanager.persist(employee1);
   entitymanager.persist(employee2);
   entitymanager.persist(employee3);

   //Create Employeelist
   List<Employee> emplist = new ArrayList();
   emplist.add(employee1);
   emplist.add(employee2);
   emplist.add(employee3);

   //Create Department Entity
   Department department = new Department();
   department.setName("Development");
   department.setEmployeelist(emplist);

   //Store Department
   entitymanager.persist(department);
```

```
      entitymanager.getTransaction().commit();
      entitymanager.close();
      emfactory.close();
      }
   }
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output check MySQL workbench as follows. In this project three tables are created.

Pass the following query in MySQL interface and the result of **department_employee** table in a tabular format is shown as follows in the query:

```
Select * from department_Id;

Department_Id   Employee_Eid
254             251
254             252
254             253
```

In the above table, deparment_id and employee_id fields are the foreign keys (reference fields) from department and employee tables.

Pass the following query in MySQL interface and the result of department table in a tabular format is shown as follows in the query:

```
Select * from department;

Id      Name
254     Development
```

Pass the following query in MySQL interface and the result of employee table in a tabular format is shown as follows in the query:

```
Select * from employee;

Eid     Deg                     Ename          Salary
251     Technical Writer        Satish         45000
252     Technical Writer        Krishna        45000
253     Technical Writer        Masthanvali    50000
```

@OneToOne Relation

In One-To-One relationship, one item can belong to only one other item. It means each row of one entity is referred to one and only one row of another entity.

Let us consider the above example. **Employee** and **Department** in a reverse unidirectional manner, the relation is One-To-One relation. It means each employee belongs to only one department. Create a JPA project in eclipse IDE named **JPA_Eclipselink_OTO**. All the modules of this project are shown as follows:

Creating Entities

Follow the above given diagram for creating entities. Create a package named **'com.tutorialspoin.eclipselink.entity'** under **'src'** package. Create a class named **Department.java** under given package. The class Department entity is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

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

@Entity
public class Department {

   @Id
   @GeneratedValue( strategy=GenerationType.AUTO )
   private int id;
   private String name;

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName( ) {
      return name;
   }

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

Create the second entity in this relation -Employee entity class, named **Employee.java** under **‘com.tutorialspoint.eclipselink.entity’** package. The Employee entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

@Entity
public class Employee {

   @Id
   @GeneratedValue( strategy= GenerationType.AUTO )
   private int eid;
   private String ename;
   private double salary;
   private String deg;

   @OneToOne
   private Department department;

   public Employee(int eid, String ename, double salary, String deg) {
      super( );
      this.eid = eid;
      this.ename = ename;
      this.salary = salary;
      this.deg = deg;
   }

   public Employee( ) {
      super();
   }

   public int getEid( ) {
      return eid;
   }

   public void setEid(int eid) {
      this.eid = eid;
   }

   public String getEname( ) {
      return ename;
   }

   public void setEname(String ename) {
      this.ename = ename;
   }

   public double getSalary( ) {
      return salary;
   }

   public void setSalary(double salary) {
      this.salary = salary;
   }

   public String getDeg( ) {
      return deg;
   }

   public void setDeg(String deg) {
      this.deg = deg;
   }

   public Department getDepartment() {
      return department;
   }

   public void setDepartment(Department department) {
      this.department = department;
   }
}
```

Persistence.xml

Persistence.xml will be created by the eclipse IDE while creating a JPA Project. The configuration details are user specifications. The persistence.xml file is shown as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<persistence version = "2.0" xmlns = "http://java.sun.com/xml/ns/persistenc
   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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_
      <class>com.tutorialspoint.eclipselink.entity.Employee</class>
      <class>com.tutorialspoint.eclipselink.entity.Department</class>

      <properties>
         <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql
         <property name = "javax.persistence.jdbc.user" value = "root"/>
         <property name = "javax.persistence.jdbc.password" value = "root"
         <property name = "javax.persistence.jdbc.driver" value = "com.mysq
         <property name = "eclipselink.logging.level" value = "FINE"/>
         <property name = "eclipselink.ddl-generation" value = "create-tabl
      </properties>

   </persistence-unit>
</persistence>
```

Service Classes

This module contains the service classes, which implements the relational part using the attribute initialization. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**. The DAO class named **OneToOne.java** is created under the given package. The DAO class is shown as follows:

```
package com.tutorialspointeclipselink.service;

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

import com.tutorialspoint.eclipselink.entity.Department;
import com.tutorialspoint.eclipselink.entity.Employee;

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

   EntityManagerFactory emfactory = Persistence.createEntityManagerFactory
   EntityManager entitymanager = emfactory.createEntityManager( );
   entitymanager.getTransaction( ).begin( );

   //Create Department Entity
   Department department = new Department();
   department.setName("Development");
```

```
      //Store Department
      entitymanager.persist(department);

      //Create Employee Entity
      Employee employee = new Employee();
      employee.setEname("Satish");
      employee.setSalary(45000.0);
      employee.setDeg("Technical Writer");
      employee.setDepartment(department);

      //Store Employee
      entitymanager.persist(employee);

      entitymanager.getTransaction().commit();
      entitymanager.close();
      emfactory.close();
      }
   }
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output, check MySQL workbench as follows. In the above example two tables are created.

Pass the following query in MySQL interface and the result of **department** table in a tabular format is shown as follows in the query:

```
Select * from department

Id       Name
301      Development
```

Pass the following query in MySQL interface and the result of **employee** table in a tabular format is shown as follows in the query:

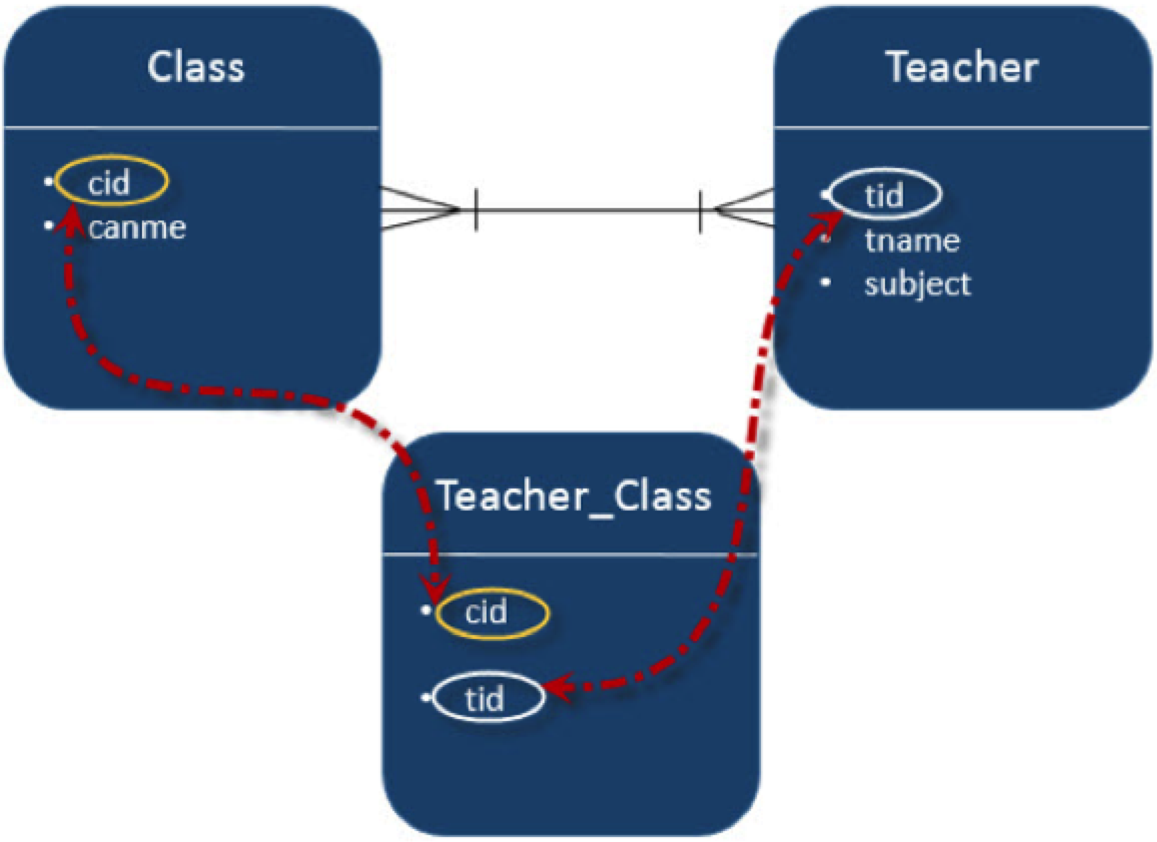
```
Select * from employee

Eid      Deg                     Ename    Salary   Department_id
302      Technical Writer        Satish   45000    301
```

@ManyToMany Relation

Many-To-Many relationship is where one or more rows from one entity are associated with more than one row in other entity.

Let us consider an example of relation between Class and Teacher entities. In bidirectional manner, both Class and Teacher have Many-To-One relation. That means each record of Class is referred by Teacher set (teacher ids), which should be primary keys in Teacher table and stored in Teacher_Class table and vice versa. Here, Teachers_Class table contains both foreign Key fields. Create a JPA project in eclipse IDE named **JPA_Eclipselink_MTM**. All the modules of this project are shown as follows:



Creating Entities

Follow the above given diagram for creating entities. Create a package named **‘com.tutorialspoin.eclipselink.entity’** under **‘src’** package. Create a class named **Clas.java** under given package. The class Department entity is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class Clas {

   @Id
   @GeneratedValue( strategy = GenerationType.AUTO )

   private int cid;
   private String cname;

   @ManyToMany(targetEntity=Teacher.class)
   private Set teacherSet;

   public Clas(){
      super();
   }

   public Clas(int cid, String cname, Set teacherSet) {
      super();
      this.cid = cid;
      this.cname = cname;
      this.teacherSet = teacherSet;
   }

   public int getCid(){
      return cid;
   }

   public void setCid(int cid) {
      this.cid = cid;
   }

   public String getCname() {
      return cname;
   }

   public void setCname(String cname) {
      this.cname = cname;
   }

   public Set getTeacherSet() {
      return teacherSet;
   }

   public void setTeacherSet(Set teacherSet) {
      this.teacherSet = teacherSet;
   }
}
```

Create the second entity in this relation -Employee entity class, named **Teacher.java** under **‘com.tutorialspoint.eclipselink.entity’** package. The Employee entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class Teacher {

   @Id
   @GeneratedValue( strategy = GenerationType.AUTO )
   private int tid;
   private String tname;
   private String subject;

   @ManyToMany(targetEntity = Clas.class)
   private Set clasSet;

   public Teacher(){
      super();
   }

   public Teacher(int tid, String tname, String subject, Set clasSet) {
      super();
      this.tid = tid;
      this.tname = tname;
      this.subject = subject;
      this.clasSet = clasSet;
   }

   public int getTid() {
      return tid;
   }

   public void setTid(int tid) {
      this.tid = tid;
   }
   public String getTname() {
      return tname;
   }

   public void setTname(String tname) {
      this.tname = tname;
   }

   public String getSubject() {
      return subject;
   }

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

   public Set getClasSet() {
      return clasSet;
   }

   public void setClasSet(Set clasSet) {
      this.clasSet = clasSet;
   }
}
```

Persistence.xml

Persistence.xml will be created by the eclipse IDE while cresting a JPA Project. The configuration details are user specifications. The persistence.xml file is shown as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<persistence version = "2.0" xmlns = "http://java.sun.com/xml/ns/persistenc
   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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_
      <class>com.tutorialspoint.eclipselink.entity.Employee</class>
      <class>com.tutorialspoint.eclipselink.entity.Department</class>

      <properties>
      <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql://l
      <property name = "javax.persistence.jdbc.user" value = "root"/>
      <property name = "javax.persistence.jdbc.password" value = "root"/
      <property name = "javax.persistence.jdbc.driver" value = "com.mysql.j
      <property name = "eclipselink.logging.level" value = "FINE"/>
      <property name = "eclipselink.ddl-generation" value = "create-tables'
      </properties>

   </persistence-unit>
</persistence>
```

Service Classes

This module contains the service classes, which implements the relational part using the attribute initialization. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**. The DAO class named **ManyToMany.java** is created under given package. The DAO class is shown as follows:

```
package com.tutorialspoint.eclipselink.service;

import java.util.HashSet;
import java.util.Set;

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

import com.tutorialspoint.eclipselink.entity.Clas;
import com.tutorialspoint.eclipselink.entity.Teacher;

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

   EntityManagerFactory emfactory = Persistence.createEntityManagerFactory(
   EntityManager entitymanager = emfactory.createEntityManager( );
   entitymanager.getTransaction( ).begin( );

   //Create Clas Entity
   Clas clas1 = new Clas(0, "1st", null);
   Clas clas2 = new Clas(0, "2nd", null);
   Clas clas3 = new Clas(0, "3rd", null);

   //Store Clas
   entitymanager.persist(clas1);
   entitymanager.persist(clas2);
   entitymanager.persist(clas3);
```

```
      //Create Clas Set1
      Set<Clas> classSet1 = new HashSet();
      classSet1.add(clas1);
      classSet1.add(clas2);
      classSet1.add(clas3);

      //Create Clas Set2
      Set<Clas> classSet2 = new HashSet();
      classSet2.add(clas3);
      classSet2.add(clas1);
      classSet2.add(clas2);

      //Create Clas Set3
      Set<Clas> classSet3 = new HashSet();
      classSet3.add(clas2);
      classSet3.add(clas3);
      classSet3.add(clas1);

      //Create Teacher Entity
      Teacher teacher1 = new Teacher(0, "Satish","Java",classSet1);
      Teacher teacher2 = new Teacher(0, "Krishna","Adv Java",classSet2);
      Teacher teacher3 = new Teacher(0, "Masthanvali","DB2",classSet3);

      //Store Teacher
      entitymanager.persist(teacher1);
      entitymanager.persist(teacher2);
      entitymanager.persist(teacher3);


      entitymanager.getTransaction( ).commit( );
      entitymanager.close( );
      emfactory.close( );
      }
   }
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output, check MySQL workbench as follows. In this example project, three tables are created.

Pass the following query in MySQL interface and the result of **teacher_clas** table in a tabular format is shown as follows in the query.

```
Select * form teacher_clas;

Teacher _tid    Classet_cid
```

```
354             351
355             351
356             351
354             352
355             352
356             352
354             353
355             353
356             353
```

In the above table teacher_tid is the foreign key from teacher table, and classet_cid is the foreign key from class table. Therefore different teachers are allotted to different class.

Pass the following query in MySQL interface and the result of teacher table in a tabular format is shown as follows in the query:

```
Select * from teacher;

Tid     Subject     Tname
354     Java        Satish
355     Adv Java    Krishna
356     DB2         Masthanvali
```

Pass the following query in MySQL interface and the result of **clas** table in a tabular format is shown as follows in the query:

```
Select * from clas;

cid     Cname
351     1st
352     2nd
353     3rd
```