Inheritance Mapping in Hibernate

We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

1. Table Per Hierarchy
2. Table Per Concrete class
3. Table Per Subclass

Table Per Hierarchy

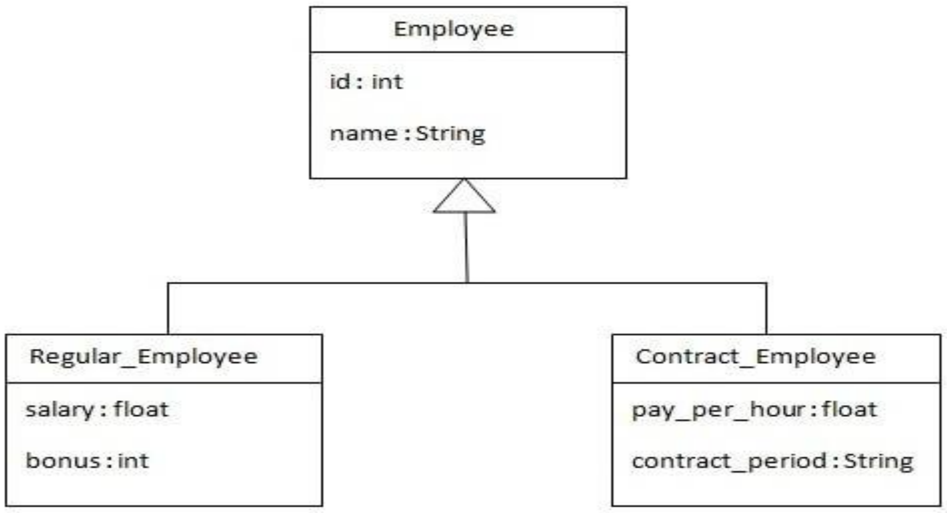
In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table .

a) Table Per Hierarchy using xml file
b) Table Per Hierarchy using Annotation

Table Per Hierarchy using xml file in Hibernate:

By this inheritance strategy, we can map the whole hierarchy by single table only. Here, an extra column (also known as **discriminator column**) is created in the table to identify the class.

Let's understand the problem first. I want to map the whole hierarchy given below into one table of the database.



There are three classes in this hierarchy. Employee is the super class for Regular_Employee and Contract_Employee classes. Let's see the mapping file for this hierarchy.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">


<hibernate-mapping>
```

```
<class name="com.javatpoint.mypackage.Employee" table="emp121" discriminaor-value="emp">
<id name="id">
<generator class="increment"></generator>
</id>

<discriminator column="type" type="string"></discriminator>
<property name="name"></property>

<subclass name="com.javatpoint.mypackage.Regular_Employee" discriminator-value="reg_emp">
<property name="salary"></property>
<property name="bonus"></property>
</subclass>

<subclass name="com.javatpoint.mypackage.Contract_Employee" discriminator-value="con_emp">
<property name="pay_per_hour"></property>
<property name="contract_duration"></property>
</subclass>

</class>

</hibernate-mapping>
```

In case of table per class hierarchy an discriminator column is added by the hibernate framework that specifies the type of the rec distinguish the record. To specify this, **discriminator** subelement of class must be specified.

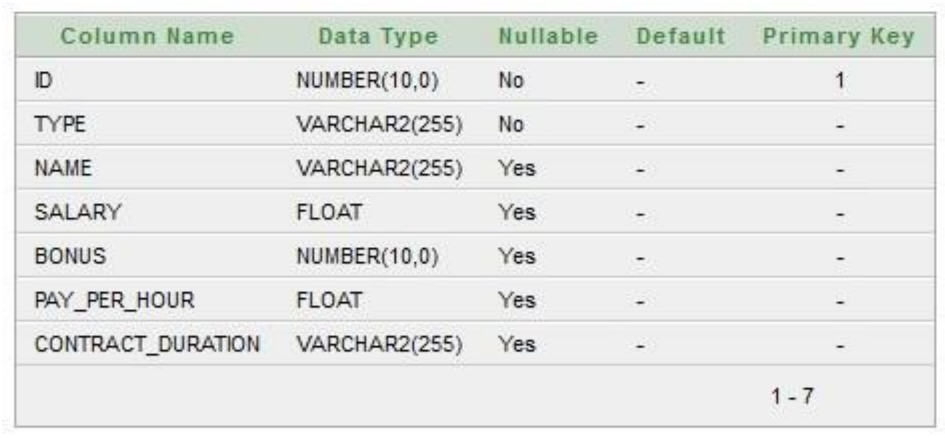
The **subclass** subelement of class, specifies the subclass. In this case, Regular_Employee and Contract_Employee are the subclasses of

The table structure for this hierarchy is as shown below:

47.3M

853

Java Try Catch



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
TYPE	VARCHAR2(255)	No	-	-
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 7

Example of Table per class hierarchy

In this example we are creating the three classes and provide mapping of these classes in the employee.hbm.xml file.

1) Create the Persistent classes

You need to create the persistent classes representing the inheritance. Let's create the three classes for the above hierarchy:

File: Employee.java

```
package com.klef.emp_package;


public class Employee {
private int id;
private String name;


//Generate the getters and setters
}
```

File: Regular_Employee.java

```
package com.klef.emp_package;


public class Regular_Employee extends Employee{
private float salary;
private int bonus;


//getters and setters
}
```

File: Contract_Employee.java

```
package com.klef.emp_package;


public class Contract_Employee extends Employee{
   private float pay_per_hour;
   private String contract_duration;


//getters and setters
}
```

2) Create the mapping file for Persistent class

The mapping has been discussed above for the hierarchy.

File: employee.hbm.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
          "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">
  <hibernate-mapping>
<class name="com.javatpoint.mypackage.Employee" table="emp121" discriminator-value="emp">
<id name="id">
<generator class="increment"></generator>
</id>

<discriminator column="type" type="string"></discriminator>
<property name="name"></property>

<subclass name="com.javatpoint.mypackage.Regular_Employee" discriminator-value="reg_emp">
<property name="salary"></property>
<property name="bonus"></property>
</subclass>

<subclass name="com.javatpoint.mypackage.Contract_Employee" discriminator-value="con_emp">
<property name="pay_per_hour"></property>
<property name="contract_duration"></property>
</subclass>

</class>

</hibernate-mapping>
```

3) Add mapping of hbm file in configuration file

Open the hibernate.cgf.xml file, and add an entry of mapping resource like this:

1. <mapping resource="employee.hbm.xml"/>

Now the configuration file will look like this:

File: hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">
```

```
<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">jtp</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <mapping resource="employee.hbm.xml"/>
    </session-factory>

</hibernate-configuration>
```

The hbm2ddl.auto property is defined for creating automatic table in the database.

---

4) Create the class that stores the persistent object

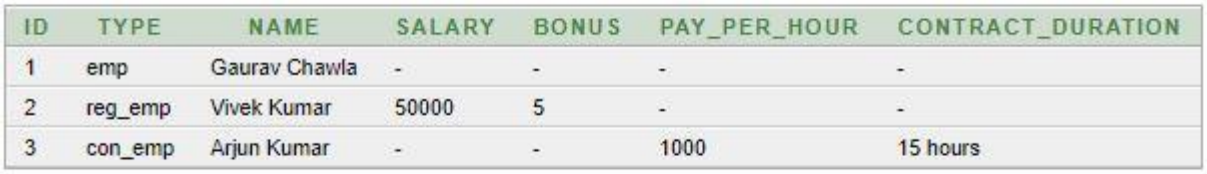
In this class, we are simply storing the employee objects in the database.

File: StoreData.java

```java
package com.klef.emp_package;

    import org.hibernate.Session;
    import org.hibernate.SessionFactory;
    import org.hibernate.Transaction;
    import org.hibernate.boot.Metadata;
    import org.hibernate.boot.MetadataSources;
    import org.hibernate.boot.registry.StandardServiceRegistry;
    import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

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

        StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").buil
    d();
        Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

        SessionFactory factory=meta.getSessionFactoryBuilder().build();
        Session session=factory.openSession();
```

```
     Transaction t=session.beginTransaction();

     Employee e1=new Employee();
     e1.setName("Gaurav Chawla");

     Regular_Employee e2=new Regular_Employee();
     e2.setName("Vivek Kumar");
     e2.setSalary(50000);
     e2.setBonus(5);

     Contract_Employee e3=new Contract_Employee();
     e3.setName("Arjun Kumar");
     e3.setPay_per_hour(1000);
     e3.setContract_duration("15 hours");

     session.persist(e1);
     session.persist(e2);
     session.persist(e3);

     t.commit();
     session.close();
     System.out.println("success");
   }
   }
```

Output:



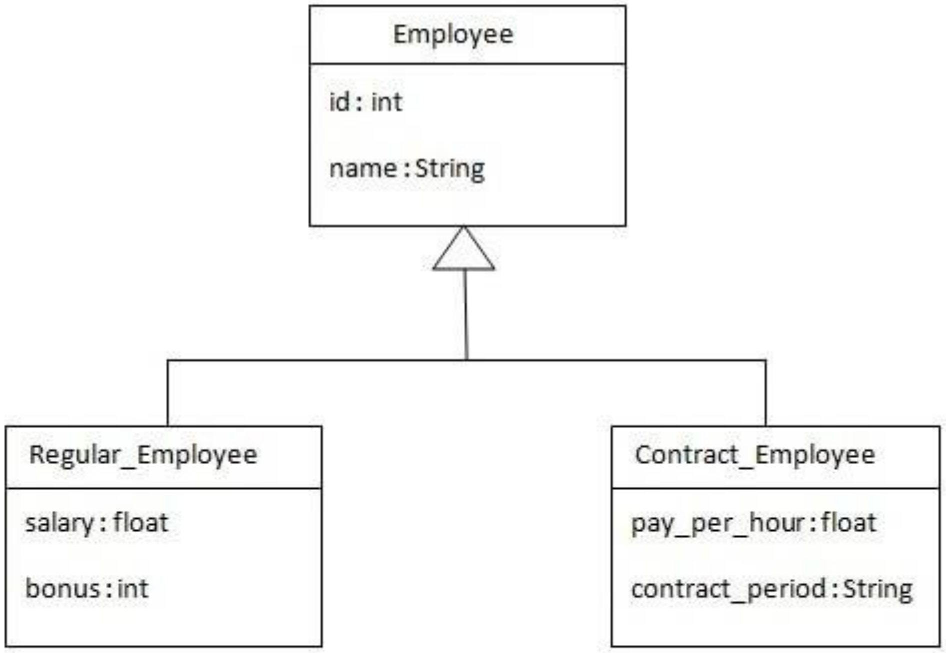
ID	TYPE	NAME	SALARY	BONUS	PAY_PER_HOUR	CONTRACT_DURATION
1	emp	Gaurav Chawla	-	-	-	-
2	reg_emp	Vivek Kumar	50000	5	-	-
3	con_emp	Arjun Kumar	-	-	1000	15 hours

Hibernate Table Per Hierarchy using Annotation

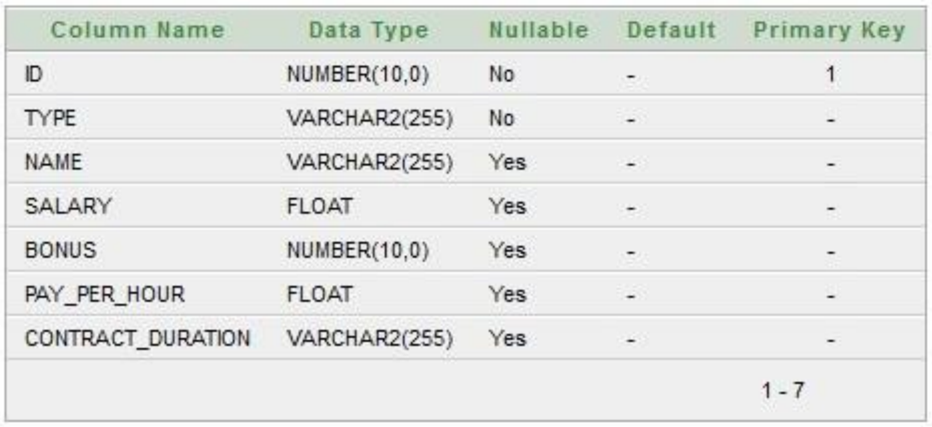
In the previous page, we have mapped the inheritance hierarchy with one table using xml file. Here, we are going to perform this task using annotation. You need to use @Inheritance(strategy=InheritanceType.SINGLE_TABLE), @DiscriminatorColumn and @DiscriminatorValue annotations for mapping table per hierarchy strategy.

In case of table per hierarchy, only one table is required to map the inheritance hierarchy. Here, an extra column (also known as **discriminator column**) is created in the table to identify the class.

Let's see the inheritance hierarchy:



There are three classes in this hierarchy. Employee is the super class for Regular_Employee and Contract_Employee classes.



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
TYPE	VARCHAR2(255)	No	-	-
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 7

Example of Hibernate Table Per Hierarchy using Annotation

You need to follow the following steps to create simple example:

- Create the persistent classes
- Edit pom.xml file
- Create the configuration file
- Create the class to store the fetch the data

1) Create the Persistent classes

You need to create the persistent classes representing the inheritance. Let's create the three classes for the above hierarchy:

File: Employee.java

```
package com.klef.emp_package;
import javax.persistence.*;

@Entity
@Table(name = "employee101")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="type",discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue(value="employee")

public class Employee {
@Id
@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name = "id")
private int id;

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

//setters and getters
}
```

File: Regular_Employee.java

```
package com.klef.emp_package;

import javax.persistence.*;

@Entity
@DiscriminatorValue("regularemployee")
public class Regular_Employee extends Employee{

@Column(name="salary")
private float salary;

@Column(name="bonus")
private int bonus;

//setters and getters
}
```

File: Contract_Employee.java

```
package com.klef.emp_package;

import javax.persistence.Column;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("contractemployee")
public class Contract_Employee extends Employee{

    @Column(name="pay_per_hour")
    private float pay_per_hour;

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

    //setters and getters
}
```

---

2) Add project information and configuration in pom.xml file.

Open pom.xml file and click source. Now, add the below dependencies between <dependencies>....</dependencies> tag.

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.1.Final</version>
</dependency>

<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc14</artifactId>
    <version>10.2.0.4.0</version>
</dependency>
```

---

3) Add the persistent classes in configuration file

Open the hibernate.cgf.xml file, and add entries of entity classes like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-5.3.dtd">
<hibernate-configuration>
    <session-factory>

    <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
<property name="connection.username">system</property>
<property name="connection.password">jtp</property>
<property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>

        <mapping class=" com.klef.emp_package "/>
        <mapping class="com.klef.emp_package.Regular_Employee"/>
        <mapping class="com.klef.emp_package.Contract_Employee"/>

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

The hbm2ddl.auto property is defined for creating automatic table in the database.

---

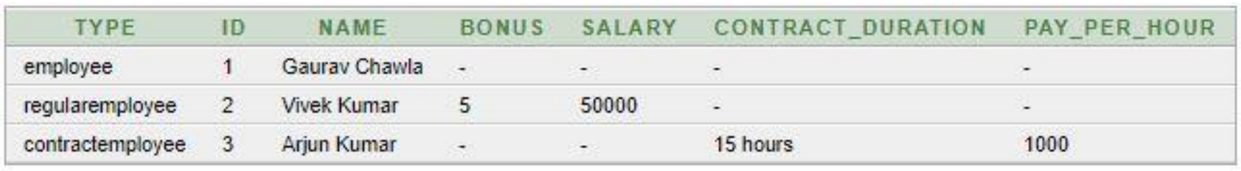
4) Create the class that stores the persistent object

In this class, we are simply storing the employee objects in the database.

File: StoreTest.java

```
package com.klef.emp_package;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreTest {

    public static void main(String args[])
    {
        StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configue("hibernate.cfg.xml").b
uild();
        Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

        SessionFactory factory=meta.getSessionFactoryBuilder().build();
        Session session=factory.openSession();

         Transaction t=session.beginTransaction();

            Employee e1=new Employee();
            e1.setName("Gaurav Chawla");

            Regular_Employee e2=new Regular_Employee();
            e2.setName("Vivek Kumar");
            e2.setSalary(50000);
            e2.setBonus(5);

            Contract_Employee e3=new Contract_Employee();
            e3.setName("Arjun Kumar");
            e3.setPay_per_hour(1000);
            e3.setContract_duration("15 hours");

            session.persist(e1);
            session.persist(e2);
            session.persist(e3);

            t.commit();
            session.close();
            System.out.println("success");
    }
}
```

Output:



TYPE	ID	NAME	BONUS	SALARY	CONTRACT_DURATION	PAY_PER_HOUR
employee	1	Gaurav Chawla	-	-	-	-
regularemployee	2	Vivek Kumar	5	50000	-	-
contractemployee	3	Arjun Kumar	-	-	15 hours	1000

Table Per Concrete class

In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

- Table Per Concrete class using xml file
- Table Per Concrete class using Annotation

2) Example for Table per concrete class using Annotations

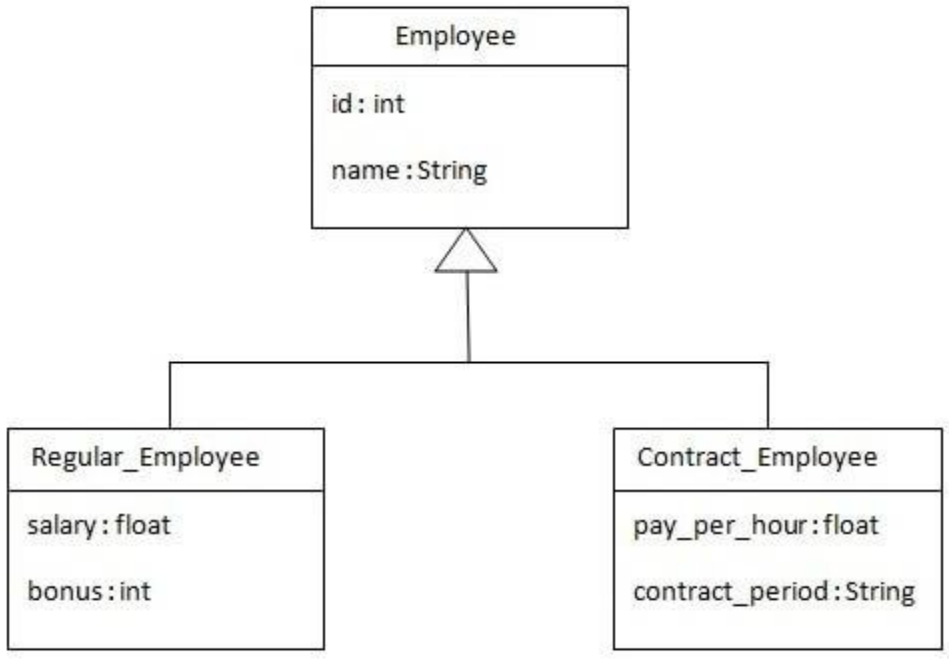
In case of Table Per Concrete class, tables are created per class. So there are no nullable values in the table. Disadvantage of this approach is that duplicate columns are created in the subclass tables.

Here, we need to use @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) annotation in the parent class and @AttributeOverrides annotation in the subclasses.

@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) specifies that we are using table per concrete class strategy. It should be specified in the parent class only.

@AttributeOverrides defines that parent class attributes will be overriden in this class. In table structure, parent class table columns will be added in the subclass table.

The class hierarchy is given below:



```
package com.klef.emppackage;
import javax.persistence.*;


@Entity
@Table(name = "employee102")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)


public class Employee {
```

```
@Id
@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name = "id")
private int id;

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

//setters and getters
}
```

File: Regular_Employee.java

```
package com.klef.emppackage;
import javax.persistence.*;

@Entity
@Table(name="regularemployee102")
@AttributeOverrides({
   @AttributeOverride(name="id", column=@Column(name="id")),
   @AttributeOverride(name="name", column=@Column(name="name"))
})
public class Regular_Employee extends Employee{

@Column(name="salary")
private float salary;

@Column(name="bonus")
private int bonus;

//setters and getters
}
```

File: Contract_Employee.java

```
package com.klef.emppackage;
import javax.persistence.*;
@Entity
@Table(name="contractemployee102")
@AttributeOverrides({
   @AttributeOverride(name="id", column=@Column(name="id")),
   @AttributeOverride(name="name", column=@Column(name="name"))
})
```

```
public class Contract_Employee extends Employee{

    @Column(name="pay_per_hour")
    private float pay_per_hour;

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

    public float getPay_per_hour() {
        return pay_per_hour;
    }
    public void setPay_per_hour(float payPerHour) {
        pay_per_hour = payPerHour;
    }
    public String getContract_duration() {
        return contract_duration;
    }
    public void setContract_duration(String contractDuration) {
        contract_duration = contractDuration;
    }
}
```

Pom.xml

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.1.Final</version>
</dependency>

<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc14</artifactId>
    <version>10.2.0.4.0</version>
</dependency>
```

File: hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<!-- Generated by MyEclipse Hibernate Tools.                   -->
```

```
<hibernate-configuration>
    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">scott</property>
        <property name="connection.password">tiger</property>
<property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>

        <mapping class="com.klef.emppackage.Employee"/>
        <mapping class="com.klef.emppackage.Contract_Employee"/>
        <mapping class="com.klef.emppackage.Regular_Employee"/>
    </session-factory>
</hibernate-configuration>
```

File: StoreData.java

```
package com.klef.emppackage;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreData {

    public static void main(String[] args) {

        StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configur("hibernate.cfg.xml").bui
ld();
        Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

        SessionFactory factory=meta.getSessionFactoryBuilder().build();
        Session session=factory.openSession();

        Transaction t=session.beginTransaction();

        Employee e1=new Employee();
        e1.setName("Gaurav Chawla");
```

```
            Regular_Employee e2=new Regular_Employee();
            e2.setName("Vivek Kumar");
            e2.setSalary(50000);
            e2.setBonus(5);

            Contract_Employee e3=new Contract_Employee();
            e3.setName("Arjun Kumar");
            e3.setPay_per_hour(1000);
            e3.setContract_duration("15 hours");

            session.persist(e1);
            session.persist(e2);
            session.persist(e3);

            t.commit();
            session.close();
            System.out.println("success");
          }
        }
```

Table Per Subclass

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.eatures of Java - Javatpoint

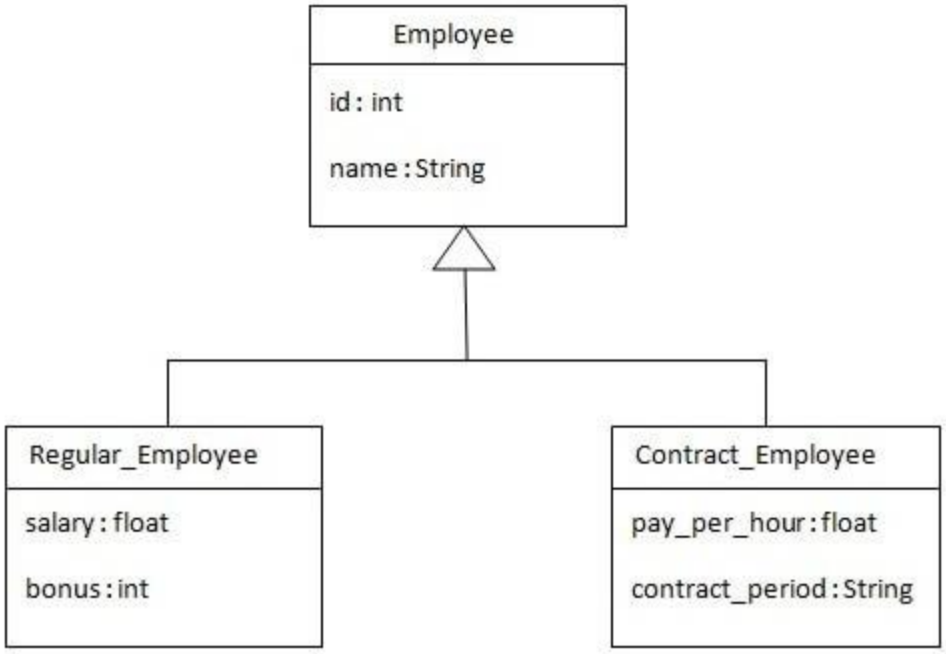
- Table Per Subclass using xml file
- Table Per Subclass using Annotation

Example for Table per Subclass using Annotations:

In case of table per subclass strategy, tables are created as per persistent classes but they are treated using primary and foreign key. So there will not be any duplicate column in the relation.

We need to specify **@Inheritance(strategy=InheritanceType.JOINED)** in the parent class and **@PrimaryKeyJoinColumn** annotation in the subclasses.

Let's see the hierarchy of classes that we are going to map.



File: Employee.java

```
package com.klef.emppackage;
import javax.persistence.*;

@Entity
@Table(name = "employee103")
@Inheritance(strategy=InheritanceType.JOINED)

public class Employee {
@Id
@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name = "id")
private int id;

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

//setters and getters
}
```

File: Regular_Employee.java

```
package com.klef.emppackage;

import javax.persistence.*;
```

```
@Entity
@Table(name="regularemployee103")
@PrimaryKeyJoinColumn(name="ID")
public class Regular_Employee extends Employee{

@Column(name="salary")
private float salary;

@Column(name="bonus")
private int bonus;

//setters and getters
}
```

File: Contract_Employee.java

```
package com.klef.emppackage;

import javax.persistence.*;

@Entity
@Table(name="contractemployee103")
@PrimaryKeyJoinColumn(name="ID")
public class Contract_Employee extends Employee{

   @Column(name="pay_per_hour")
   private float pay_per_hour;

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

   //setters and getters
}
```

---

2) Add project information and configuration in pom.xml file.

Open pom.xml file and click source. Now, add the below dependencies between <dependencies>....</dependencies> tag.

```
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-core</artifactId>
```

```
    <version>5.3.1.Final</version>
</dependency>

<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc14</artifactId>
    <version>10.2.0.4.0</version>
</dependency>
```

---

3)Create the configuration file

Open the hibernate.cgf.xml file, and add an entry of mapping resource like this:

```
<mapping class="com.klef.emppackage.Employee"/>
<mapping class="com.klef.emppackage.Contract_Employee"/>
<mapping class="com.klef.emppackage.Regular_Employee"/>
```

Now the configuration file will look like this:

File: hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
       <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">jtp</property>
<property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>

        <mapping class="com.klef.emppackage.Employee"/>
       <mapping class="com.klef.emppackage.Contract_Employee"/>
       <mapping class="com.klef.emppackage.Regular_Employee"/>
    </session-factory>
```

```
</hibernate-configuration>
```

The hbm2ddl.auto property is defined for creating automatic table in the database.

---

3) Create the class that stores the persistent object

In this class, we are simply storing the employee objects in the database.

File: StoreData.java

```
package com.klef.emppackage;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreData {

  public static void main(String args[])
  {
    StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").
build();
    Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

    SessionFactory factory=meta.getSessionFactoryBuilder().build();
    Session session=factory.openSession();

     Transaction t=session.beginTransaction();

      Employee e1=new Employee();
      e1.setName("Gaurav Chawla");

      Regular_Employee e2=new Regular_Employee();
      e2.setName("Vivek Kumar");
      e2.setSalary(50000);
      e2.setBonus(5);
```

```
            Contract_Employee e3=new Contract_Employee();
            e3.setName("Arjun Kumar");
            e3.setPay_per_hour(1000);
            e3.setContract_duration("15 hours");

            session.persist(e1);
            session.persist(e2);
            session.persist(e3);

            t.commit();
            session.close();
            System.out.println("success");
    }
}
```