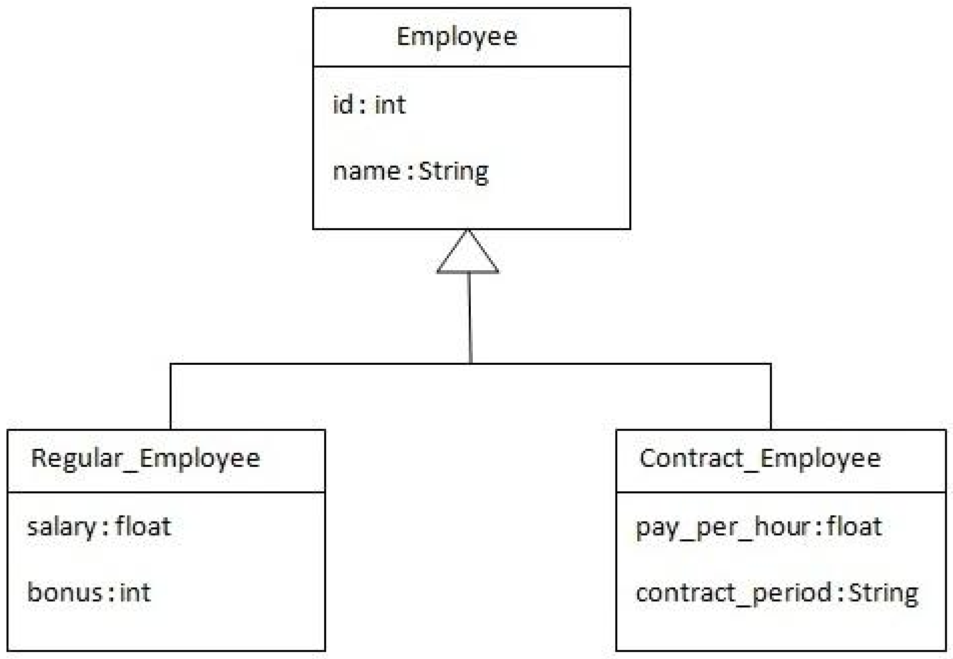
Table Per Concrete class using xml file

In case of Table Per Concrete class, there will be three tables in the database having no relations to each other. There are two ways to map the table with table per concrete class strategy.

- By union-subclass element
- By Self creating the table for each class

Let's understand what hierarchy we are going to map.



Let's see how can we map this hierarchy by union-subclass element:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

  <hibernate-mapping>
  <class name="com.javatpoint.mypackage.Employee" table="emp122">
  <id name="id">
  <generator class="increment"></generator>
  </id>
```

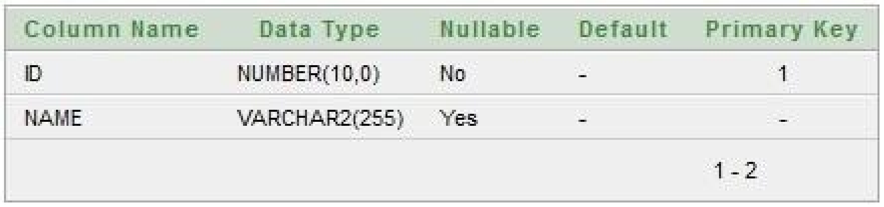
```
 <property name="name"></property>

<union-subclass name="com.javatpoint.mypackage.Regular_Employee" table="regemp122">
 <property name="salary"></property>
 <property name="bonus"></property>
 </union-subclass>

<union-subclass name="com.javatpoint.mypackage.Contract_Employee" table="contemp122">
 <property name="pay_per_hour"></property>
 <property name="contract_duration"></property>
 </union-subclass>

 </class>

 </hibernate-mapping>
```

In case of table per concrete class, there will be three tables in the database, each representing a particular class.

The **union-subclass** subelement of class, specifies the subclass. It adds the columns of parent table into this table. In other words, it is working as a union.

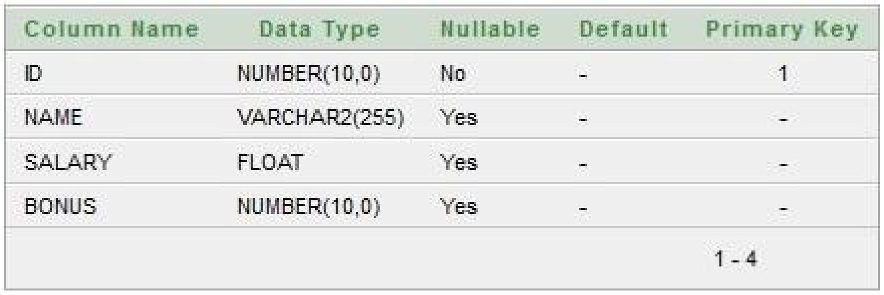
The table structure for each table will be as follows:

Table structure for Employee class



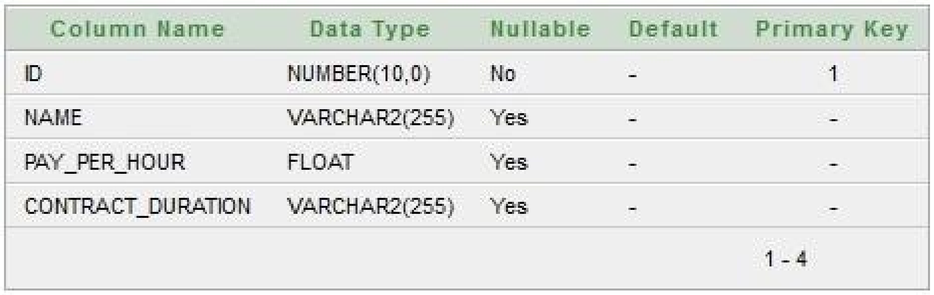
Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
				1 - 2

Table structure for Regular_Employee class



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
			1 - 4	

Table structure for Contract_Employee class



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
			1 - 4	

Example of Table per concrete class

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.javatpoint.mypackage;


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


//getters and setters
}
```

File: Regular_Employee.java

```
package com.javatpoint.mypackage;
```

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

//getters and setters
}
```

File: Contract_Employee.java

```
package com.javatpoint.mypackage;

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 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">



  <hibernate-mapping>
  <class name="com.javatpoint.mypackage.Employee" table="emp122">
  <id name="id">
  <generator class="increment"></generator>
  </id>

  <property name="name"></property>
```

```
                                                                    <union-
subclass name="com.javatpoint.mypackage.Regular_Employee" table="regemp122">
  <property name="salary"></property>
  <property name="bonus"></property>
  </union-subclass>

                                                                    <union-
subclass name="com.javatpoint.mypackage.Contract_Employee" table="contemp122">
  <property name="pay_per_hour"></property>
  <property name="contract_duration"></property>
  </union-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:

```
<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 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.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">oracle</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

```
package com.javatpoint.mypackage;

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

public class StoreData {
public static void main(String[] args) {
    Session session=new Configuration().configure("hibernate.cfg.xml")
                        .buildSessionFactory().openSession();

    Transaction t=session.beginTransaction();

    Employee e1=new Employee();
    e1.setName("sonoo");

    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");
}
}
```

download this example developed using Myeclipse IDE
download this example developed using Eclipse IDE

← prev next →

Share 0