IS502052: Enterprise Systems Development Concepts

Lab 8: Entity Relationship

I. Introduction

In this lab tutorial, we will continue the previous labs, but we will solve the problem when having more than one tables and the relationship between them. There are **3** popular relationships:

- *Many-to-many* relationship;
- *One-to-many* relationship;
- *One-to-one* relationship.

II. Many-to-many Relationship

In systems analysis, a many-to-many relationship is a type of cardinality that refers to the relationship between two entities *A* and *B* in which *A* may contain a parent instance for which there are many children in *B*, and vice versa.

In a relational database management system, such relationships are usually implemented by means of an ***associative table*** (also known as *cross-reference table*). In this case the logical primary key for *AB* is formed from the two foreign keys (*i.e.*, copies of the primary keys of *A* and *B*).

Consider this example, in user account management system, a ***user*** (including *username* and *password*) can have many ***roles*** (*admin*, *moderator*, *normal*, *anonymous*), and a ***role*** can belong to many ***users***.

Technically, when modeling database design, we must have another table, containing the primary keys of *book* and *author* tables, as in **Figure 1**.

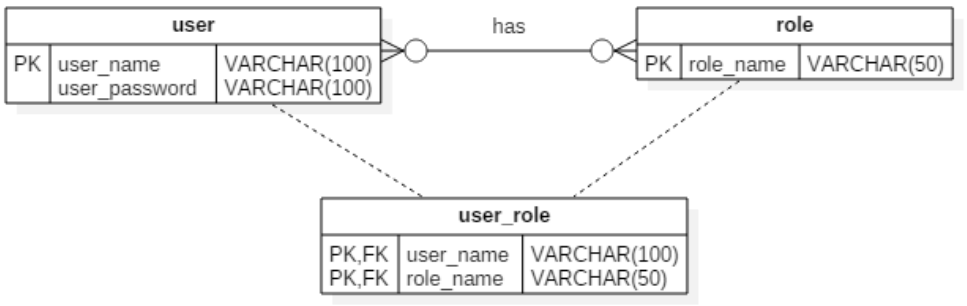

Figure 1 Database model

To illustrate the **many-to-many** relationship, we should do the following steps to implement application:

- Create database and tables in PostgreSQL, then, add to data source on JBoss server;
- Create entity classes (EJB Modules);
- Build and deploy application to JBoss server;
- Create EJB client, a console based application.

1. Create database and tables

Do the following steps, as explained in previous lab, to create database and tables:

- Open **pgAdmin** and start **PostgreSQL** server;
- Create new database, name ***SampleDB3***;
- Create two tables, under ***SampleDB3/Schemas/public/Tables***, named ***user*** and ***role***.
  + In ***user*** table, add two columns ***user_name*** and ***user_password***, the datatype is `text`.
  + In ***role*** table, add two columns ***role_name***, the datatype is `text`.
- Create *associative table* by doing these steps below:
  + Create table name ***user_role***;
  + Next, switch to **Columns** tab, and add two fields: ***user_name*** (`text`), ***role_name*** (`text`);

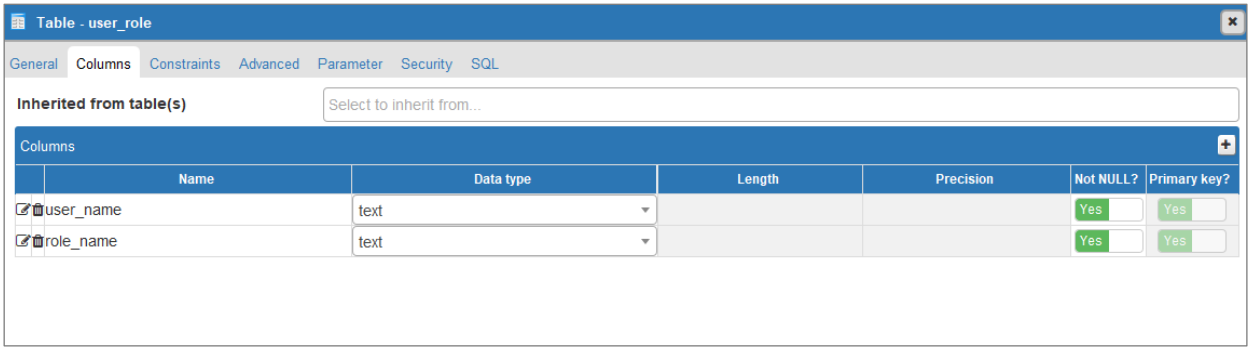

Figure 2 Columns of book_author table

  + Then, switch to **Constraints** tab and select **Foreign Key** in current tab;
  + For ***first*** foreign key, (1) click **Add** button, (2) enter ***user_name_fk*** as name, (3) click **Edit** button, (4) switch to **Columns** tab, (5) select ***user_name*** as local column, (6) select ***public.user*** as references table, (7) select ***user_name*** for referencing, (8) click **Add**.

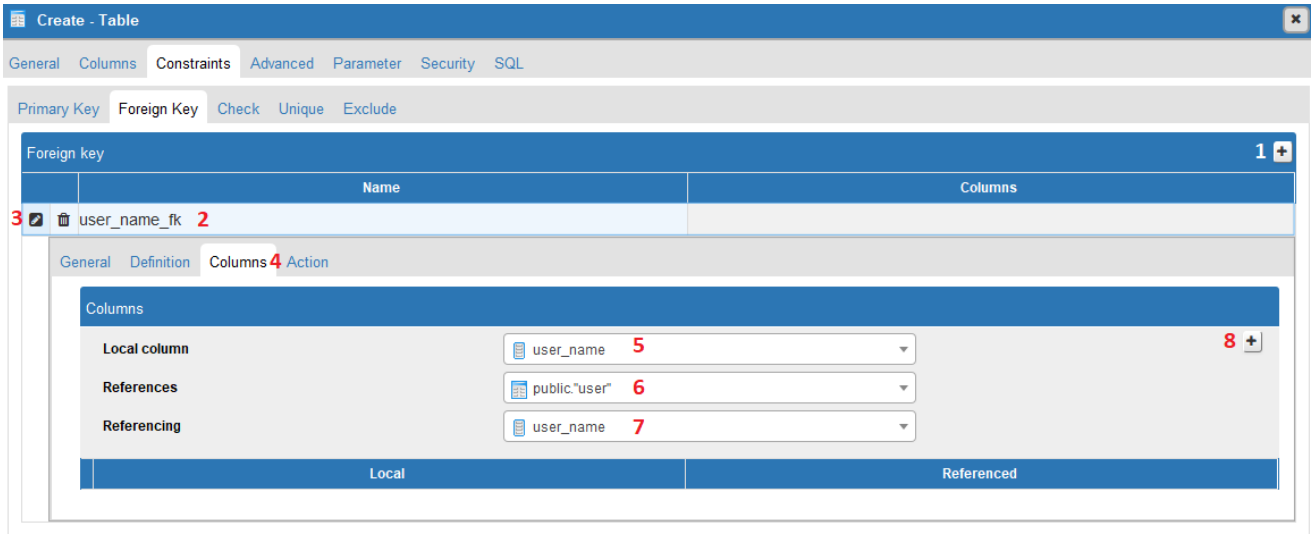

Figure 3 Create constraint

+ Do the above step again to add another constraint for ***role_name***, named ***role_name_fk***.
+ The result will look as **Figure 4**.

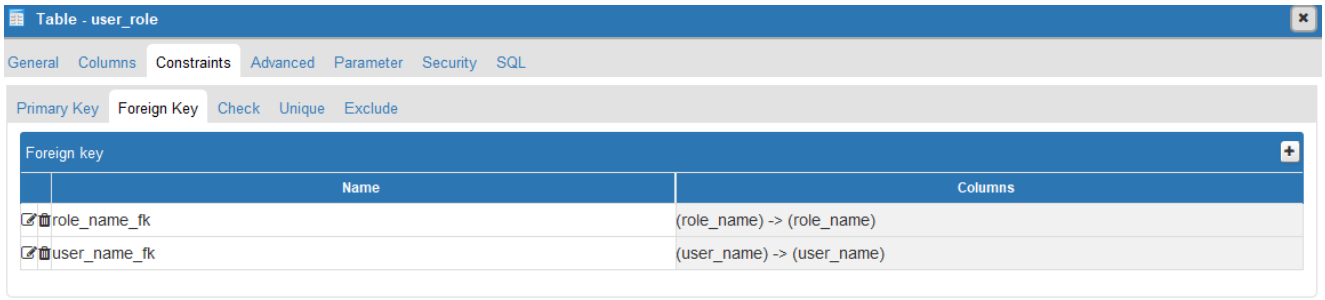

Figure 4 Add constraint on foreign key

- Insert some sample data into tables;
- Finally, configure this database as data source in JBoss server.

2. Create entity classes (EJB Modules)

Do the following steps, as explained in previous lab, to create entity classes:

- In NetBeans, switch to **Services** tab and create connection to ***SampleDB3*** database;
- Create new Java EE project, name ***SampleEJB-04***;
- Right-click on source packages and create **Entity Classes from Database**;
- Select ***SampleDB3-DS*** as data source;
- Instead of adding all columns to create entity classes, we just add only ***user_role*** table, and all referenced tables (***user*** and ***role***) will be added automatically (**Figure 5**);

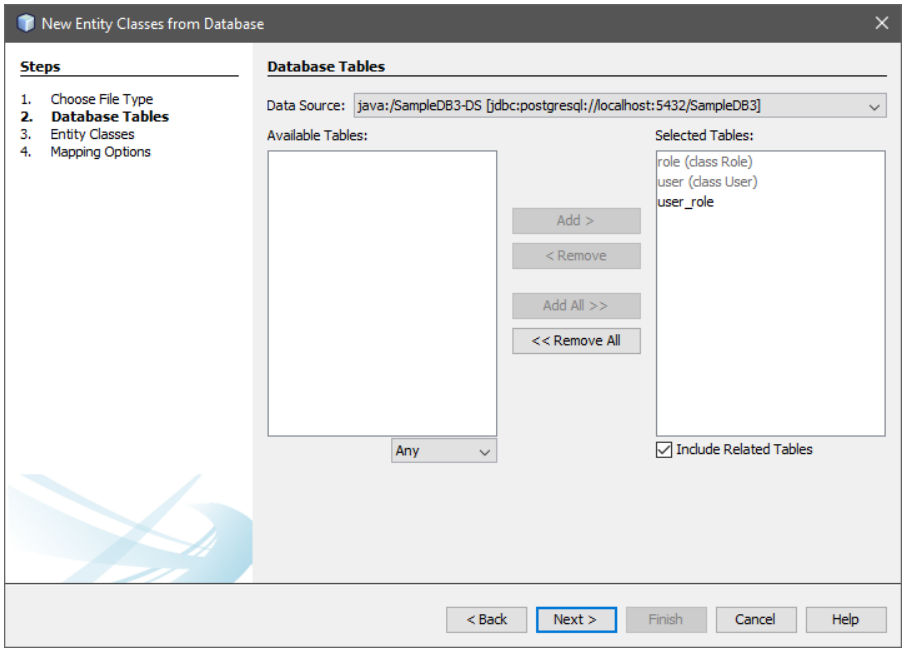

Figure 5 Crate new entity classes from database (1)

- Next, enter ***edu.tdt.entity*** as package name (**Figure 6**);

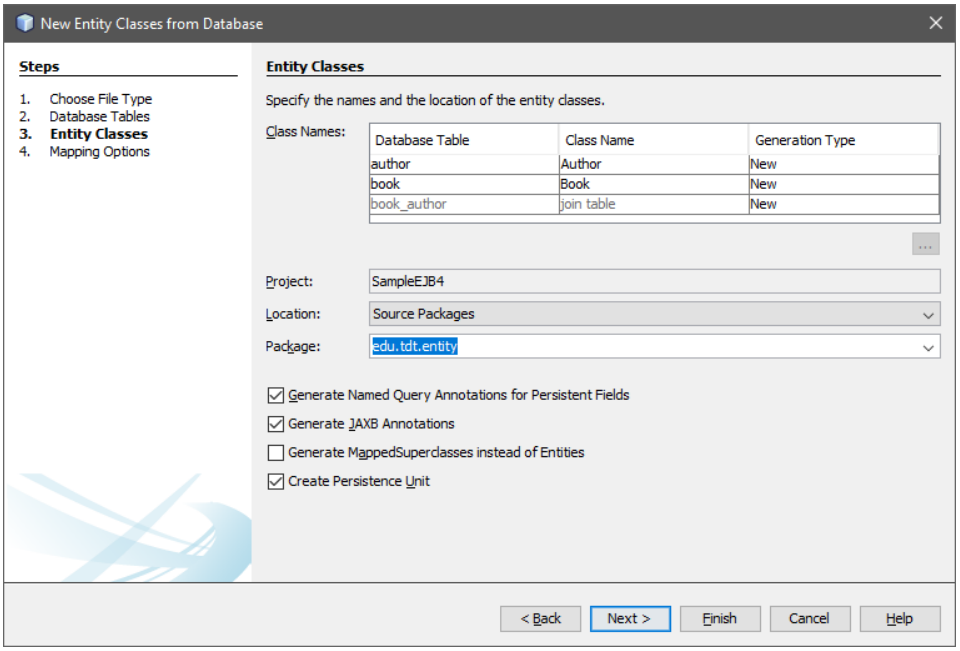

Figure 6 Crate new entity classes from database (2)

- In final step, you need to select the two top options (**Figure 7**).

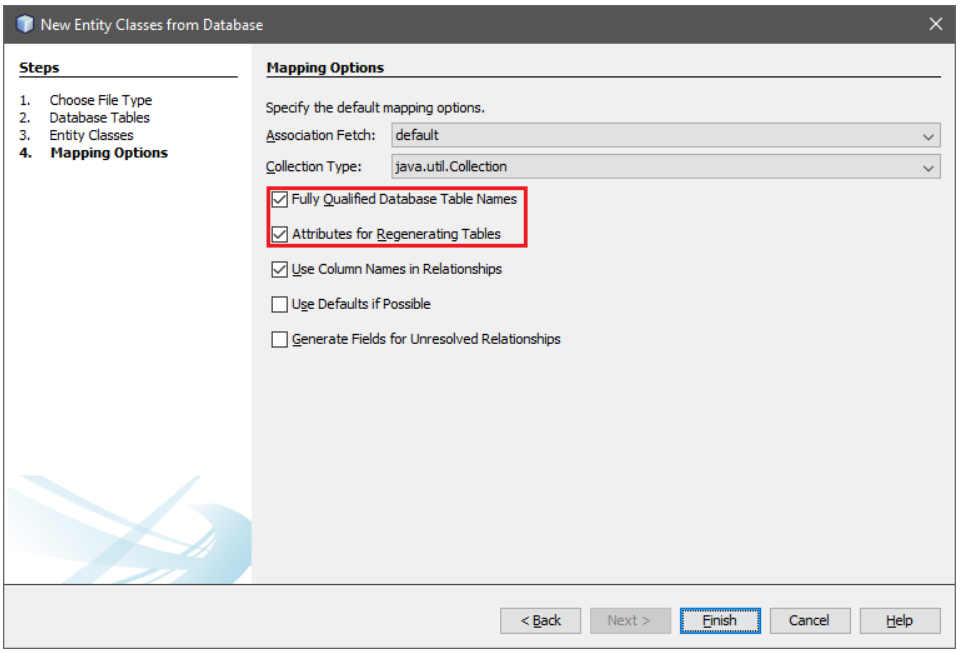

Figure 7 Crate new entity classes from database (3)

- Create session bean class, name ***SystemManagerRemote*** and ***SystemManager***;

```java
package edu.tdt.entity;

import java.util.ArrayList;
import javax.ejb.Remote;

@Remote
public interface SystemManagementRemote {

    void insertUser(String userName, String userPassword);
    void insertRole(String roleName);
    void insertUserRole(String userName, String roleName);
    ArrayList<String> searchRole(String rolename);
    int getUserPassword(String username);

}
```

```java
package edu.tdt.entity;

import java.util.ArrayList;
import java.util.Collection;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateful
public class SystemManagement implements SystemManagementRemote {

    @PersistenceContext(unitName = "SampleEJB-04PU")
    private EntityManager em;

    public SystemManagement()
    {

    }

    @Override
    public void insertUser(String userName, String userPassword)
    {
        User user = new User(userName, userPassword);
        em.persist(user);
    }

    @Override
    public void insertRole(String roleName)
    {
        Role role = new Role(roleName);
        em.persist(role);
    }

    @Override
    public void insertUserRole(String userName, String roleName)
    {
        User user = em.find(User.class, userName);
        Role role = em.find(Role.class, roleName);

        user.getRoleCollection().add(role);
        role.getUserCollection().add(user);
    }

    @Override
    public ArrayList<String> searchRole(String rolename)
    {
        Role role = em.find(Role.class, rolename);

        if(role != null)
        {
            ArrayList<String> arrOutput = new ArrayList<String>();
            for(User user : role.getUserCollection())
            {
                arrOutput.add(user.getUserName());
            }
            return arrOutput;
        }
        return null;
    }
```

```
    @Override
    public int getUserPassword(String username)
    {
        User user = em.find(User.class, username);
        if(user != null)
            return user.getUserPassword().hashCode();
        return -1;
    }

}
```

3. Build and deploy application to JBoss server

After completing **Step 2**, you need to build and deploy application to JBoss server, as explained in previous tutorial.

4. Create EJB client

Do the following steps, as explained in previous lab, to create EJB client:

- Create Java project, name ***SampleEJB5Test***;
- Add ***SampleEJB5*** and **JBoss Client API** to project's classpath;
- Create ***jndi.properties*** and ***jboss-ejb-client.properties***;
- Create Java application, named ***EJBTester.java***;

```
package edu.tdt.test;

import edu.tdt.entity.*;
import java.io.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.*;

public class EJBTester {

    private Properties props;
    private InitialContext ctx;

    public EJBTester()
    {
        readJNDI();
    }

    /**
     * Read the JNDI properties file
     */
    private void readJNDI()
    {
        props = new Properties();

        try
        {
            props.load(new FileInputStream("jndi.properties"));
        } catch(IOException e)
        {
            e.getMessage();
        }

        try
        {
            ctx = new InitialContext(props);
        } catch (NamingException ex)
        {
            ex.getMessage();
        }
    }
```

```
    /**
     * Construct and return the JNDI address of called class
     * @return String
     */
    private String getJNDI()
    {
        String appName = "";
        String moduleName = "SampleEJB-04";
        String distinctName = "";
        String sessionBeanName = SystemManagement.class.getSimpleName();
        String viewClassName = SystemManagementRemote.class.getName() + "?stateful";

        return "ejb:"+appName+"/"+moduleName+"/"+distinctName+"/"+sessionBeanName+"!"+viewClassName;
    }

    /**
     * Show the GUI in console window
     */
    private void showGUI()
    {
        System.out.println("\n========================");
        System.out.println("TDT System Management");
        System.out.println("========================");
        System.out.print("Options: \n1. Add User \n2. Add Role \n3. Search Users by Role \n4. Exit
\nEnter Choice: ");
    }

    /**
     * Test the Stateless EJB
     */
    public void testEntityEJB()
    {
        try
        {
            Scanner sc = new Scanner(System.in);

            // Lookup the SystemManagementRemote
            SystemManagementRemote session = (SystemManagementRemote) ctx.lookup(getJNDI());

            // Log-in
            String _username, _password;
            System.out.println("Log-in to TDT System Management");
            System.out.print("Username: ");
            _username = sc.nextLine();
            System.out.print("Password: ");
            _password = sc.nextLine();

            if(_password.hashCode() != session.getUserPassword(_username))
            {
                System.err.println("Wrong username/password!");
                return;
            }

            // Show GUI
            int choice;
            String username, password, rolename;
            do
            {
                showGUI();
                choice = Integer.parseInt(sc.nextLine());

                switch(choice)
                {
                    case 1: // Insert new user with created role
                        System.out.print("Enter username  : ");
                        username = sc.nextLine();
                        System.out.print("Enter password  : ");
                        password = sc.nextLine();
                        System.out.print("Enter role name : ");
                        rolename = sc.nextLine();

                        session.insertUser(username, password);
                        session.insertUserRole(username, rolename);

                        System.out.println("Done!");
                        break;
```

```
                    case 2: // Insert new role
                        System.out.print("Enter role: ");
                        rolename = sc.nextLine();

                        session.insertRole(rolename);

                        System.out.println("Done!");
                        break;

                    case 3: // Search user by role
                        System.out.print("Enter role name: ");
                        rolename = sc.nextLine();

                        for(String s : session.searchRole(rolename))
                        {
                            System.out.println(s);
                        }

                        System.out.println("Done!");
                        break;

                    default:
                        break;
                }

            } while(choice != 4);

        } catch (NamingException ex)
        {
            Logger.getLogger(EJBTester.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

- Then, create ***main*** function to test this application.

In this tutorial program, you should notice that we implement a ***log-in function***, which prompt the user to enter username and password to validate on the database. Since we must keep user's password privately, we use ***hashCode()*** method to encode the password string.

III. Exercise

1. Continue the sample program, as we have log-in function, we can now extend the system as follows:
   - If user log-in as admin, run the *System Management GUI*.
   - If user log-in as moderator, run the *TDTU Bookstore GUI*.
   - If user log-in as anonymous, run the *TDTU Bookstore GUI*, but only have search book/author function.