

# 实验六

## # 一、相关知识点

1. JDBC基本概念
2. PreparedStatement的用法
3. JDBC数据增、删、改，事务控制等

## # 二、实验目的：

理解Java连接数据库的基本概念。理解利用Statement对象、PreparedStatement对象进行增、删、改操作，理解事务的概念和JDBC编程方式。

## # 三、实验内容：

### 1、利用PreparedStatement进行查询。

第一步：在PublisherManager类中增加方法 `public List<BeanPublisher> searchPublisher(String keyword) throws BaseException` 方法，要求根据关键字在出版社表中查询满足条件的出版社（出版社名称或地址中包含参数中的关键字），参考loadAllPublisher()方法，将查询结果封装为List

第二步：在main函数中编写测试代码进行该方法的调用测试。

```
1 public List<BeanPublisher> searchPublisher(String keyword) throws BaseException{
2     List<BeanPublisher> result = new ArrayList<BeanPublisher>();
3     Connection conn = null;
4     try {
5         conn = DBUtil.getConnection();
6         String sql = "select pubid,publisherName,address" +
7             " from beanpublisher";
8         if (keyword != null && !"".equals(keyword)) {
9             sql += " where publisherName like ? "
10                + "or address like ?";
```

```

11     }
12     sql += " order by pubid";
13     java.sql.PreparedStatement pst = conn.prepareStatement(sql);
14     if (keyword != null && !"".equals(keyword)) {
15         pst.setString(1, "%" + keyword + "%");
16         pst.setString(2, "%" + keyword + "%");
17     }
18     java.sql.ResultSet rs = pst.executeQuery();
19     while (rs.next()) {
20         BeanPublisher r = new BeanPublisher();
21         r.setPubid(rs.getString(1));
22         r.setPublisherName(rs.getString(2));
23         r.setAddress(rs.getString(3));
24         result.add(r);
25     }
26 } catch (SQLException e) {
27     e.printStackTrace();
28     throw new DbException(e);
29 } finally {
30     if (conn != null)
31         try {
32             conn.close();
33         } catch (SQLException e) {
34             // TODO Auto-generated catch block
35             e.printStackTrace();
36         }
37 }
38 return result;
39 }

```

```

1 public static void main(String[] args) {
2     BeanPublisher p = new BeanPublisher();
3     PublisherManager pm = new PublisherManager();
4     try {
5         List<BeanPublisher> lst = pm.searchPublisher("2");
6         for (int i = 0; i < lst.size(); i++) {
7             p = lst.get(i);
8             System.out.println(p.getPubid() + "," + p.getPublisherName() + "," +
p.getAddress());
9         }
10    } catch (BaseException e) {
11        // TODO Auto-generated catch block

```

```
12         e.printStackTrace();
13     }
14     // try {
15     //     pm.deletePublisher("testpubid");
16     // } catch (BaseException e) {
17     //     // TODO Auto-generated catch block
18     //     e.printStackTrace();
19     // }
20 }
```

```
mysql> select * from beanpublisher;
+-----+-----+-----+
| pubid | publisherName | address |
+-----+-----+-----+
| 1      | 11             | 111     |
| 2      | 22             | 222     |
| 3      | 33             | 333     |
| 4      | 12             | 112     |
| 5      | 23             | 223     |
| 6      | 34             | 334     |
| 7      | 16             | 233     |
+-----+-----+-----+
7 rows in set (0.00 sec)
```

```
299 public static void main(String[] args) {
300     BeanPublisher p = new BeanPublisher();
301     PublisherManager pm = new PublisherManager();
302     try {
303         List<BeanPublisher> lst = pm.searchFor();
304         for (int i = 0; i < lst.size(); i++)
305             p = lst.get(i);
306         System.out.println(p.getPubid());
307     }
308     } catch (BaseException e) {
309         // TODO Auto-generated catch block
310         e.printStackTrace();
311     }
312     try {
313         pm.deletePublisher("testpubid");
314     } catch (BaseException e) {
315         // TODO Auto-generated catch block
316         e.printStackTrace();
317     }
318 }
319 }
320
```

<

Problems @ Javadoc Declaration 控制台 ×

<已终止> PublisherManager [Java 应用程序] C:\Users\Bexh0lder\.p2\pool\plugins\c

2,22,222  
4,12,112  
5,23,223  
7,16,233

## 2、 利用Statement对象进行数据添加。

第一步：修改PublisherManager类的createPublisher方法，将其中的insert语言改换成用Statement对象执行；

第二步：运行图书管理系统，进行添加出版社测试。

### 【实验结果与分析】

A、 写出替换的代码部分。

```
1 public void createPublisher(BeanPublisher p) throws BaseException {
2     if (p.getPubid() == null || "".equals(p.getPubid()) || p.getPubid().length() > 20) {
3         throw new BusinessException("出版社编号必须是1-20个字");
4     }
5 }
```

```

5         if (p.getPublisherName() == null || "".equals(p.getPublisherName()) ||
p.getPublisherName().length() > 50) {
6             throw new BusinessException("出版社名称必须是1-50个字");
7         }
8         if (p.getAddress() == null || "".equals(p.getAddress()) || p.getAddress().length()
> 100) {
9             throw new BusinessException("出版地址必须是1-100个字");
10        }
11
12
13        Connection conn = null;
14        try {
15            conn = DBUtil.getConnection();
16            String sql = "select * from BeanPublisher where pubid=?";
17            java.sql.PreparedStatement pst = conn.prepareStatement(sql);
18            pst.setString(1, p.getPubid());
19            java.sql.ResultSet rs = pst.executeQuery();
20            if (rs.next()) throw new BusinessException("出版社编号已经被占用");
21            rs.close();
22            pst.close();
23            sql = "select * from BeanPublisher where publisherName=?";
24            pst = conn.prepareStatement(sql);
25            pst.setString(1, p.getPublisherName());
26            rs = pst.executeQuery();
27            if (rs.next()) throw new BusinessException("出版社名称已经存在");
28            rs.close();
29            pst.close();
30            //通过PreparedStatement实现
31            //      sql = "insert into BeanPublisher(pubid,publisherName,address)
values(?,?,?)";
32            //      pst = conn.prepareStatement(sql);
33            //      pst.setString(1, p.getPubid());
34            //      pst.setString(2, p.getPublisherName());
35            //      pst.setString(3, p.getAddress());
36            //      pst.execute();
37            //      pst.close();
38            //-----修改部分-----
39            //通过Statement实现
40            sql = "insert into BeanPublisher(pubid,publisherName,address) "
41                + "values(" + p.getPubid() + "," + p.getPublisherName() + "," +
p.getAddress() + ")";
42            java.sql.Statement st = conn.createStatement();

```

```

43         int i = st.executeUpdate(sql);
44         if(i <= 0) throw new BusinessException("插入失败");
45         //-----
46     } catch (SQLException e) {
47         e.printStackTrace();
48         throw new DbException(e);
49     } finally {
50         if (conn != null)
51             try {
52                 conn.close();
53             } catch (SQLException e) {
54                 // TODO Auto-generated catch block
55                 e.printStackTrace();
56             }
57     }
58 }

```

```

305 public static void main(String[] args) {
306     try {
307         BeanPublisher p = new BeanPublisher();
308         p.setAddress("666");
309         p.setPubid("8");
310         p.setPublisherName("88");
311         PublisherManager pm = new PublisherManager();
312         pm.createPublisher(p);
313         List<BeanPublisher> lst = pm.loadAllPublisher();
314         for (int i = 0; i < lst.size(); i++) {
315             p = lst.get(i);
316             System.out.println(p.getPubid() + "," + p.getPublisherName() + "," + p.getAddress());
317         }
318     } catch (BaseException e) {
319         // TODO Auto-generated catch block
320         e.printStackTrace();
321     }
322     try {
323         pm.deletePublisher("testpubid");
324     } catch (BaseException e) {
325         // TODO Auto-generated catch block
326         e.printStackTrace();
327     }
328 }
329 }

```

Problems Javadoc Declaration 控制台 ×

<已终止> PublisherManager [Java 应用程序] C:\Users\Bexh0lder\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.2.v20220201-1208\jre\bi

```

1,11,111
2,22,222
3,33,333
4,12,112
5,23,223
6,34,334
7,16,233
8,88,666

```

### 3、利用insert语句添加数据时，未指定字段值处理。

第一步：将数据库表beanreadertype的readerTypeId的自动递增属性去掉。

对象 beanreadertype @booklib (...)

新建 保存 另存为 添加字段 插入字段 删除字段 主键

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不
readerTypeid	int	11	0	
readerTypeName	varchar	50	0	
lendBookLimited	int	11	0	

默认:

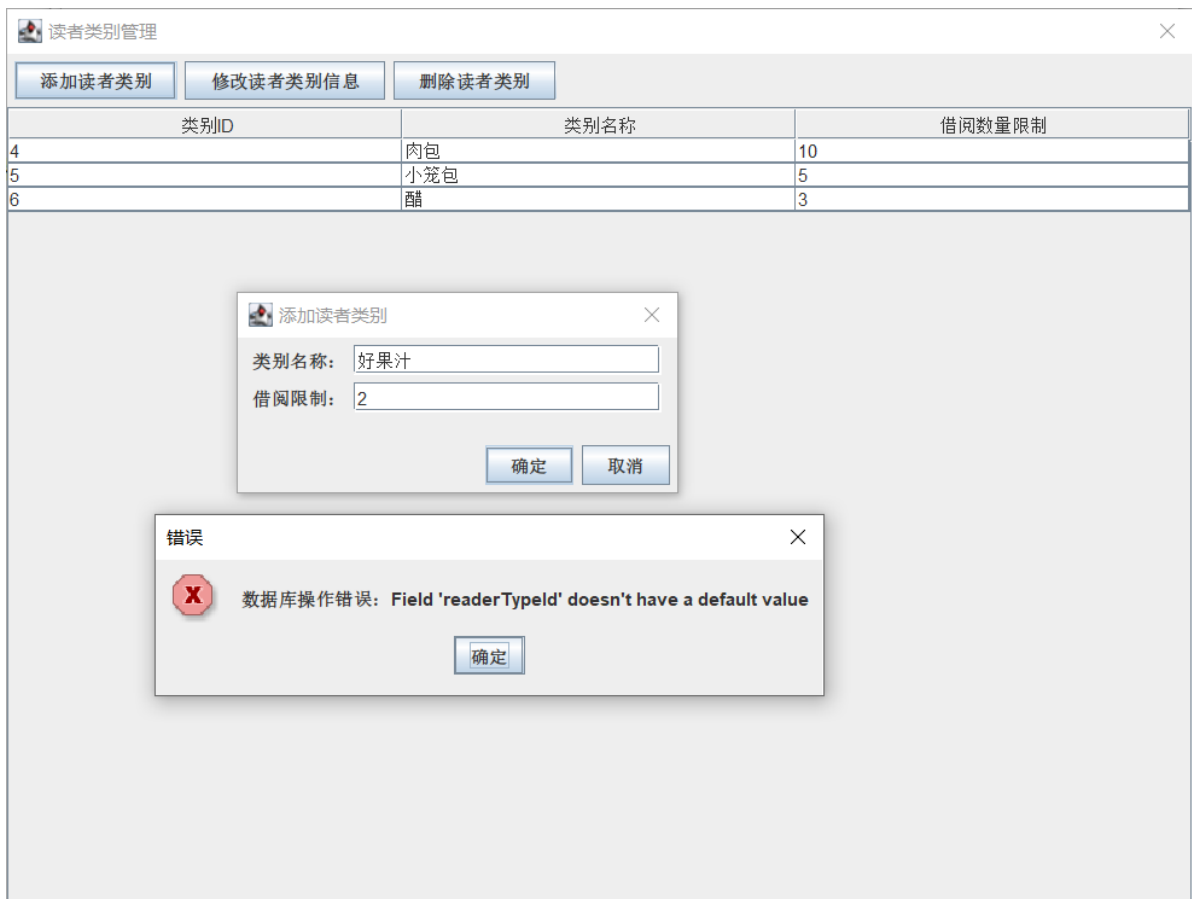
注释:

☐ 自动递增

☐ 无符号

☐ 填充零

第二步：运行图书管理系统，打开读者类别管理界面，并尝试添加一个读者类别；系统将会报一个错误，请分析说明错误原因。



错误原因：报错内容信息是readerTypeId字段没有默认值，在将字段的自增属性取消后，插入新记录时,不会再自动地创建readerTypeId字段的值，所以报错

第三步：应该如何修改程序，使新增读者类别的ID为表中现有数据的最大ID值+1。

```

1 public void createReaderType(BeaReaderType rt) throws BaseException {
2     if (rt.getReaderTypeName() == null || "".equals(rt.getReaderTypeName()) ||
rt.getReaderTypeName().length() > 20) {
3         throw new BusinessException("读者类别名称必须是1-20个字");
4     }
5     if (rt.getLendBookLimited() < 0 || rt.getLendBookLimited() > 100) {
6         throw new BusinessException("借阅图书数量必须在0-100之间");
7     }
8     Connection conn = null;
9     int maxReaderTypeId = 0; //修改处
10    try {
11        conn = DBUtil.getConnection();
12        String sql = "select * from BeaReaderType where readerTypeName=?";
13        java.sql.PreparedStatement pst = conn.prepareStatement(sql);
14        pst.setString(1, rt.getReaderTypeName());
15        java.sql.ResultSet rs = pst.executeQuery();
16        if (rs.next()) throw new BusinessException("读者类别名称已经被占用");
17        rs.close();
18        pst.close();

```



```

19 //-----修改处-----
20 sql = "select max(readerTypeId) from beanreadertype";
21 pst = conn.prepareStatement(sql);
22 rs = pst.executeQuery();
23 if (rs.next()) maxReaderTypeId = rs.getInt(1);
24 //-----
25 sql = "insert into
BeanReaderType(readerTypeId,readerTypeName,lendBookLimited)
values(?,?,?);"//修改处
26 pst = conn.prepareStatement(sql);
27 //-----修改处-----
28 pst.setInt(1, maxReaderTypeId + 1);
29 pst.setString(2, rt.getReaderTypeName());
30 pst.setInt(3, rt.getLendBookLimited());
31 //-----
32 pst.execute();
33 pst.close();
34 } catch (SQLException e) {
35     e.printStackTrace();
36     throw new DbException(e);
37 } finally {
38     if (conn != null)
39         try {
40             conn.close();
41         } catch (SQLException e) {
42             // TODO Auto-generated catch block
43             e.printStackTrace();
44         }
45     }
46 }

```

读者类别管理		
<input type="button" value="添加读者类别"/> <input type="button" value="修改读者类别信息"/> <input type="button" value="删除读者类别"/>		
类别ID	类别名称	借阅数量限制
4	肉包	10
5	小笼包	5
6	醋	3
7	好果汁	2

#### 4、利用PreparedStatement对象进行数据修改。

在SystemUserManager类中，新建一个modifyUserName方法，实现用户名称（username字段）的修改功能。并修改其main函数，将admin用户的名称改为：超级管理员。

##### 【实验结果与分析】

A、请提供方法代码和main函数代码。

```

1  public void modifyUserName(String userid, String newName) throws BaseException
2  {
3      Connection conn = null;
4      try {
5          conn = DBUtil.getConnection();
6          String sql = "select * from BeanSystemUser where userid=?";
7          java.sql.PreparedStatement pst = conn.prepareStatement(sql);
8          pst.setString(1, userid);
9          java.sql.ResultSet rs = pst.executeQuery();
10         if (!rs.next()) throw new BusinessException("账号不存在");
11         rs.close();
12         pst.close();
13         sql = "update BeanSystemUser set username=? where userid=?";
14         pst = conn.prepareStatement(sql);
15         pst.setString(1, newName);

```

```

15         pst.setString(2, userid);
16         pst.execute();
17         pst.close();
18     } catch (SQLException e) {
19         e.printStackTrace();
20         throw new DbException(e);
21     } finally {
22         if (conn != null)
23             try {
24                 conn.close();
25             } catch (SQLException e) {
26                 // TODO Auto-generated catch block
27                 e.printStackTrace();
28             }
29     }
30 }

```

```

1 public static void main(String[] args) {
2     BeanSystemUser test = new BeanSystemUser();
3     // user.setUserId("admin");
4     // user.setUsername("系统管理员");
5     // user.setUserType("管理员");
6     try {
7         new SystemUserManager().modifyUserName("admin", "超级管理员");
8         test = new SystemUserManager().loadUser("admin");
9         System.out.println("用户id: " + test.getUserId()
10             + " 用户姓名: " + test.getUsername()
11             + " 用户类型: " + test.getUserType());
12     } catch (BaseException e) {
13         // TODO Auto-generated catch block
14         e.printStackTrace();
15     }
16 }

```

```
259 public static void main(String[] args) {
260     BeanSystemUser test = new BeanSystemUser();
261     // user.setUserId("admin");
262     // user.setUsername("系统管理员");
263     // user.setUserType("管理员");
264     try {
265         new SystemUserManager().modifyUserName("admin", "超级管理员");
266         test = new SystemUserManager().loadUser("admin");
267         System.out.println("用户id: " + test.getUserId()
268             + " 用户姓名: " + test.getUsername()
269             + " 用户类型: " + test.getUserType());
270     } catch (BaseException e) {
271         // TODO Auto-generated catch block
272         e.printStackTrace();
273     }
274 }
275 }
276
```

Problems @ Javadoc Declaration 控制台 ×

<已终止> SystemUserManager [Java 应用程序] C:\Users\Bexh0lder\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f

用户id: admin 用户姓名: 超级管理员 用户类型: 管理员

```
mysql> select * from beansystemuser;
+-----+-----+-----+-----+-----+-----+
| userid | username | pwd   | usertype | createDate          | removeDate |
+-----+-----+-----+-----+-----+-----+
| admin  | 管理员   | admin | 管理员   | 2013-01-01 00:00:00 | NULL       |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from beansystemuser;
+-----+-----+-----+-----+-----+-----+
| userid | username   | pwd   | usertype | createDate          | removeDate |
+-----+-----+-----+-----+-----+-----+
| admin  | 超级管理员 | admin | 管理员   | 2013-01-01 00:00:00 | NULL       |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

B、思考：如果上述方法的返回值为布尔类型，即如果成功修改了用户名称，则返回true，如果用户不存在或修改失败返回false。应该如何完善代码。提示：主要statement或PreparedStatement对象的execute方法和executeUpdate方法的区别。

```
1 public boolean modifyUserName(String userid, String newName) throws
   BaseException {
2     Connection conn = null;
3     try {
4         conn = DBUtil.getConnection();
5         // String sql = "select * from BeanSystemUser where userid=?";
6         // java.sql.PreparedStatement pst = conn.prepareStatement(sql);
7         // pst.setString(1, userid);
8         // java.sql.ResultSet rs = pst.executeQuery();
9         // if (!rs.next()) throw new BusinessException("账号不存在");
```

```

10 // rs.close();
11 // pst.close();
12 // sql = "update BeanSystemUser set username=? where userid=?";
13 // pst = conn.prepareStatement(sql);
14 // pst.setString(1, newName);
15 // pst.setString(2, userid);
16 // pst.execute();
17 // pst.close();
18 String sql = "update BeanSystemUser set username= \"" + newName + "\"
where userid=\"" + userid + "\"";
19 java.sql.Statement st = conn.createStatement();
20 int i = st.executeUpdate(sql);
21 if(i <= 0) return false;
22 } catch (SQLException e) {
23     e.printStackTrace();
24     throw new DbException(e);
25 } finally {
26     if (conn != null)
27         try {
28             conn.close();
29         } catch (SQLException e) {
30             // TODO Auto-generated catch block
31             e.printStackTrace();
32         }
33 }
34 return true;
35 }

```

```
1 public static void main(String[] args) {
2     BeanSystemUser test = new BeanSystemUser();
3     try {
4         List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();
5         testList = new SystemUserManager().loadAllUsers(true);
6         if(new SystemUserManager().modifyUserName("111", "超级管理员"))
7             System.out.println("修改成功");
8         else
9             System.out.println("修改失败");
10        for(int i = 0; i < testList.size(); ++i)
11        {
12            test = testList.get(i);
13            System.out.println("用户id: " + test.getUserid()
14                + " 用户姓名: " + test.getUsername());
15        }
16    }
17 }
```

```

15         + " 用户类型: " + test.getUsertype());
16     }
17     if(new SystemUserManager().modifyUserName("admin", "超级管理员"))
18         System.out.println("修改成功");
19     else
20         System.out.println("修改失败");
21     testList = new SystemUserManager().loadAllUsers(true);
22     for(int i = 0; i < testList.size(); ++i)
23     {
24         test = testList.get(i);
25         System.out.println("用户id: " + test.getUserid()
26             + " 用户姓名: " + test.getUsername()
27             + " 用户类型: " + test.getUsertype());
28     }
29 } catch (BaseException e) {
30     // TODO Auto-generated catch block
31     e.printStackTrace();
32 }
33 }

```

executeUpdate 的返回值是一个整数（int），指示受影响行数（即更新计数），如果没有修改成功则返回值不会大于0，所以可以通过返回值来确定是否修改成功

```

264 public static void main(String[] args) {
265     BeanSystemUser test = new BeanSystemUser();
266     try {
267         List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();
268         testList = new SystemUserManager().loadAllUsers(true);
269         if(new SystemUserManager().modifyUserName("111", "超级管理员"))
270             System.out.println("修改成功");
271         else
272             System.out.println("修改失败");
273         for(int i = 0; i < testList.size(); ++i)
274         {
275             test = testList.get(i);
276             System.out.println("用户id: " + test.getUserid()
277                 + " 用户姓名: " + test.getUsername()
278                 + " 用户类型: " + test.getUsertype());
279         }
280         if(new SystemUserManager().modifyUserName("admin", "超级管理员"))
281             System.out.println("修改成功");
282         else
283             System.out.println("修改失败");
284         testList = new SystemUserManager().loadAllUsers(true);
285         for(int i = 0; i < testList.size(); ++i)
286         {
287             test = testList.get(i);
288             System.out.println("用户id: " + test.getUserid()
289                 + " 用户姓名: " + test.getUsername()
290                 + " 用户类型: " + test.getUsertype());
291         }
292     } catch (BaseException e) {
293         // TODO Auto-generated catch block

```

Problems Javadoc Declaration 控制台 ×

<已终止> SystemUserManager [Java 应用程序] C:\Users\Bexh0lder\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.2.v2022020

修改失败  
 用户id: admin 用户姓名: 管理员 用户类型: 管理员  
 修改成功  
 用户id: admin 用户姓名: 超级管理员 用户类型: 管理员

## 5、Delete语句的执行。修改用户管理类中的用户删除方法，用删除数据库表中数据的形式代替现有软删除模式。

### 【实验结果与分析】

A、修改后的sql语句部分是。

```
1 public void deleteUser(String userid) throws BaseException {
2     Connection conn = null;
3     try {
4         conn = DBUtil.getConnection();
5         String sql = "select removeDate from BeanSystemUser where userid=?";
6         java.sql.PreparedStatement pst = conn.prepareStatement(sql);
7         pst.setString(1, userid);
8         java.sql.ResultSet rs = pst.executeQuery();
9         if (!rs.next()) throw new BusinessException("登陆账号不存在或已被删除");
10        rs.close();
11        pst.close();
12        //-----修改后的sql语句-----
13        sql = "delete from beansystemuser where userid=?";
14        //-----
15        pst = conn.prepareStatement(sql);
16        pst.setString(1, userid);
17        pst.execute();
18        pst.close();
19    } catch (SQLException e) {
20        e.printStackTrace();
21        throw new DbException(e);
22    } finally {
23        if (conn != null)
24            try {
25                conn.close();
26            } catch (SQLException e) {
27                // TODO Auto-generated catch block
28                e.printStackTrace();
29            }
30    }
31 }
```

```

293 public static void main(String[] args) {
294     try {
295         List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();
296         BeanSystemUser newUser = new BeanSystemUser();
297         BeanSystemUser test = new BeanSystemUser();
298         newUser.setUsername("bex");
299         newUser.setUserid("bex");
300         newUser.setUsertype("管理员");
301         new SystemUserManager().createUser(newUser);
302         testList = new SystemUserManager().loadAllUsers(true);
303         for(int i = 0; i < testList.size(); ++i)
304         {
305             test = testList.get(i);
306             System.out.println("用户id: " + test.getUserid()
307                 + " 用户姓名: " + test.getUsername()
308                 + " 用户类型: " + test.getUsertype());
309         }
310         new SystemUserManager().deleteUser("bex");
311         testList = new SystemUserManager().loadAllUsers(true);
312         for(int i = 0; i < testList.size(); ++i)
313         {
314             test = testList.get(i);
315             System.out.println("用户id: " + test.getUserid()
316                 + " 用户姓名: " + test.getUsername()
317                 + " 用户类型: " + test.getUsertype());
318         }
319     } catch (BaseException e) {
320         // TODO Auto-generated catch block
321         e.printStackTrace();
322     }
}

```

Problems @ Javadoc Declaration 控制台 ×

<已终止> SystemUserManager [Java 应用程序] C:\Users\Bexh0lder\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win

用户id: admin 用户姓名: 超级管理员 用户类型: 管理员  
 用户id: bex 用户姓名: bex 用户类型: 管理员  
 用户id: admin 用户姓名: 超级管理员 用户类型: 管理员

B、如果对删除函数进行限制，要求不能删除已经有过借阅操作的用户。应如何修改代码。提示：可参考读者管理类中的读者类别删除方法。

```

1 public void deleteUser(String userid) throws BaseException {
2     Connection conn = null;
3     try {
4         conn = DBUtil.getConnection();
5         String sql = "select username from BeanSystemUser where userid=?";
6         java.sql.PreparedStatement pst = conn.prepareStatement(sql);
7         pst.setString(1, userid);
8         java.sql.ResultSet rs = pst.executeQuery();
9         if (!rs.next()) throw new BusinessException("账号不存在或已被删除");
10        String deleteName = rs.getString(1);
11        System.out.println(deleteName);
12        rs.close();
13        pst.close();
14        sql = "select count(*) from beanbooklendrecord where lendOperUserid=?";
15        pst = conn.prepareStatement(sql);
16        pst.setString(1, userid);
17        rs = pst.executeQuery();

```



```

18         rs.next();
19         int n = rs.getInt(1);
20         if (n > 0) throw new BusinessException(deleteName + "已经执行过" + n + "个
借阅操作, 不能删除");
21         sql = "delete from beansystemuser where userid=?";
22         pst = conn.prepareStatement(sql);
23         pst.setString(1, userid);
24         pst.execute();
25         pst.close();
26     } catch (SQLException e) {
27         e.printStackTrace();
28         throw new DbException(e);
29     } finally {
30         if (conn != null)
31             try {
32                 conn.close();
33             } catch (SQLException e) {
34                 // TODO Auto-generated catch block
35                 e.printStackTrace();
36             }
37     }
38 }

```

```

1 public static void main(String[] args) {
2     try {
3         List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();
4         BeanSystemUser newUser = new BeanSystemUser();
5         BeanSystemUser test = new BeanSystemUser();
6         newUser.setUsername("bex");
7         newUser.setUserid("bex");
8         newUser.setUserType("管理员");
9         new SystemUserManager().createUser(newUser);
10        testList = new SystemUserManager().loadAllUsers(true);
11        for(int i = 0; i < testList.size(); ++i)
12        {
13            test = testList.get(i);
14            System.out.println("用户id: " + test.getUserid()
15                + " 用户姓名: " + test.getUsername()
16                + " 用户类型: " + test.getUserType());
17        }
18        new SystemUserManager().deleteUser("bex");
19        testList = new SystemUserManager().loadAllUsers(true);

```

```

20         for(int i = 0; i < testList.size(); ++i)
21         {
22             test = testList.get(i);
23             System.out.println("用户id: " + test.getUserid()
24                 + " 用户姓名: " + test.getUsername()
25                 + " 用户类型: " + test.getUsertype());
26         }
27         new SystemUserManager().deleteUser("admin");
28     } catch (BaseException e) {
29         // TODO Auto-generated catch block
30         e.printStackTrace();
31     }
32 }

```

```

302= public static void main(String[] args) {
303     try {
304         List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();
305         BeanSystemUser newUser = new BeanSystemUser();
306         BeanSystemUser test = new BeanSystemUser();
307         newUser.setUsername("bex");
308         newUser.setUserid("bex");
309         newUser.setUsertype("管理员");
310         new SystemUserManager().createUser(newUser);
311         testList = new SystemUserManager().loadAllUsers(true);
312         for(int i = 0; i < testList.size(); ++i)
313         {
314             test = testList.get(i);
315             System.out.println("用户id: " + test.getUserid()
316                 + " 用户姓名: " + test.getUsername()
317                 + " 用户类型: " + test.getUsertype());
318         }
319         new SystemUserManager().deleteUser("bex");
320         testList = new SystemUserManager().loadAllUsers(true);
321         for(int i = 0; i < testList.size(); ++i)
322         {
323             test = testList.get(i);
324             System.out.println("用户id: " + test.getUserid()
325                 + " 用户姓名: " + test.getUsername()
326                 + " 用户类型: " + test.getUsertype());
327         }
328         new SystemUserManager().deleteUser("admin");
329     } catch (BaseException e) {
330         // TODO Auto-generated catch block
331         e.printStackTrace();
332     }
333 }
334 }
335

```

Problems Javadoc Declaration 控制台 ×

<已终止> SystemUserManager [Java 应用程序] C:\Users\Bexh0lder\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.2.v20220201-1208

用户id: admin 用户姓名: 超级管理员 用户类型: 管理员

用户id: bex 用户姓名: bex 用户类型: 管理员

bex

用户id: admin 用户姓名: 超级管理员 用户类型: 管理员

超级管理员

[cn.edu.zucc.booklib.util.BusinessException](#): 超级管理员已经执行过18个借阅操作，不能删除

at cn.edu.zucc.booklib.control.SystemUserManager.deleteUser(SystemUserManager.java:248)

at cn.edu.zucc.booklib.control.SystemUserManager.main(SystemUserManager.java:328)

6、(修改)在数据库中建立一张 **BeanBookLendRecord\_backup** 表，用于保存已经归还图书的借阅记录。其表结构与 **BeanBookLendRecord** 表完全一致。要求在借阅管理类中，增加方法，实现已经归还数据的备份功能（备份完成后，在原表中删除备份成功的数据）。提示：注意事务控制。

【实验结果与分析】

A 请提供备份表的建表语句

```
1 CREATE TABLE `beanbooklendrecord_backup` (  
2   `id` int(11) NOT NULL AUTO_INCREMENT,  
3   `readerid` varchar(20) NOT NULL,  
4   `bookBarcode` varchar(20) NOT NULL,  
5   `lendDate` datetime NOT NULL,  
6   `returnDate` datetime DEFAULT NULL,  
7   `lendOperUserid` varchar(20) NOT NULL,  
8   `returnOperUserid` varchar(20) DEFAULT NULL,  
9   `penalSum` double DEFAULT '0',  
10  PRIMARY KEY (`id`),  
11  KEY `fk_book_idx_backup` (`bookBarcode`),  
12  KEY `fk_reader_idx_backup` (`readerid`),  
13  KEY `fk_lendOper_idx_backup` (`lendOperUserid`),  
14  KEY `fk_returnOper_idx_backup` (`returnOperUserid`)  
15 ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;
```

```
mysql> CREATE TABLE `beanbooklendrecord_backup` (  
→   `id` int(11) NOT NULL AUTO_INCREMENT,  
→   `readerid` varchar(20) NOT NULL,  
→   `bookBarcode` varchar(20) NOT NULL,  
→   `lendDate` datetime NOT NULL,  
→   `returnDate` datetime DEFAULT NULL,  
→   `lendOperUserid` varchar(20) NOT NULL,  
→   `returnOperUserid` varchar(20) DEFAULT NULL,  
→   `penalSum` double DEFAULT '0',  
→   PRIMARY KEY (`id`),  
→   KEY `fk_book_idx_backup` (`bookBarcode`),  
→   KEY `fk_reader_idx_backup` (`readerid`),  
→   KEY `fk_lendOper_idx_backup` (`lendOperUserid`),  
→   KEY `fk_returnOper_idx_backup` (`returnOperUserid`)  
→ ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;  
Query OK, 0 rows affected, 2 warnings (0.03 sec)
```

B 请提供备份函数代码

```
1 //三金指定very优雅版
```

```

2      public void BackupReturnedRecord()throws DbException, SQLException{
3          Connection conn = null;
4          String sql = null;
5          try {
6              conn = DBUtil.getConnection();
7              //关闭事务自动提交
8              conn.setAutoCommit(false);
9
10             sql = "insert beanbooklendrecord_backup select * from beanbooklendrecord
where returnDate is not null";
11             java.sql.PreparedStatement pst = conn.prepareStatement(sql);
12             pst.execute();
13             pst.close();
14             sql = "delete from beanbooklendrecord where id in (select id from
beanbooklendrecord_backup)";
15             pst = conn.prepareStatement(sql);
16             pst.execute();
17             pst.close();
18             //提交存档，如果第一步成功而第二步失败时方便回到第一步执行之前
19             conn.commit();
20         }catch(SQLException e) {
21             //出错就回滚到第一步之前
22             conn.rollback();
23             e.printStackTrace();
24         }finally {
25             if (conn != null)
26                 try {
27                     conn.close();
28                 } catch (SQLException e) {
29                     // TODO Auto-generated catch block
30                     e.printStackTrace();
31                 }
32         }
33     }

```

备份前

```
mysql> select * from beanbooklendrecord;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | readerid | bookBarcode | lendDate | returnDate | lendOperUserid | returnOperUserid | penalSum |
+----+-----+-----+-----+-----+-----+-----+-----+
| 26 | 32001272 | barcode1 | 2022-05-15 14:15:22 | 2022-05-15 14:30:05 | admin | admin | 0 |
| 27 | 32001272 | barcode4 | 2022-05-15 14:15:26 | 2022-05-15 14:37:49 | admin | admin | 0 |
| 28 | 32001002 | barcode2 | 2022-05-15 14:18:02 | 2022-05-15 14:37:53 | admin | admin | 0 |
| 29 | 32001003 | barcode6 | 2022-05-15 14:18:10 | NULL | admin | NULL | 0 |
| 30 | 32001001 | barcode9 | 2022-05-15 14:18:16 | NULL | admin | NULL | 0 |
| 31 | 32001003 | barcode5 | 2022-05-15 14:18:24 | NULL | admin | NULL | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> select * from beanbooklendrecord_backup;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | readerid | bookBarcode | lendDate | returnDate | lendOperUserid | returnOperUserid | penalSum |
+----+-----+-----+-----+-----+-----+-----+-----+
| 23 | 32001272 | barcode1 | 2022-05-09 22:55:23 | 2022-05-15 14:15:05 | admin | admin | 0 |
| 25 | 32001272 | barcode2 | 2022-05-15 14:14:54 | 2022-05-15 14:15:13 | admin | admin | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

备份后

```
mysql> select * from beanbooklendrecord;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | readerid | bookBarcode | lendDate | returnDate | lendOperUserid | returnOperUserid | penalSum |
+----+-----+-----+-----+-----+-----+-----+-----+
| 29 | 32001003 | barcode6 | 2022-05-15 14:18:10 | NULL | admin | NULL | 0 |
| 30 | 32001001 | barcode9 | 2022-05-15 14:18:16 | NULL | admin | NULL | 0 |
| 31 | 32001003 | barcode5 | 2022-05-15 14:18:24 | NULL | admin | NULL | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from beanbooklendrecord_backup;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | readerid | bookBarcode | lendDate | returnDate | lendOperUserid | returnOperUserid | penalSum |
+----+-----+-----+-----+-----+-----+-----+-----+
| 23 | 32001272 | barcode1 | 2022-05-09 22:55:23 | 2022-05-15 14:15:05 | admin | admin | 0 |
| 25 | 32001272 | barcode2 | 2022-05-15 14:14:54 | 2022-05-15 14:15:13 | admin | admin | 0 |
| 26 | 32001272 | barcode1 | 2022-05-15 14:15:22 | 2022-05-15 14:30:05 | admin | admin | 0 |
| 27 | 32001272 | barcode4 | 2022-05-15 14:15:26 | 2022-05-15 14:37:49 | admin | admin | 0 |
| 28 | 32001002 | barcode2 | 2022-05-15 14:18:02 | 2022-05-15 14:37:53 | admin | admin | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

7、如果需要记录图书的入库时间（需要包含时分秒），应如何修改数据库表结构和相关代码？

【实验结果与分析】

添加字段

```
1 | alter table beanbook add storagetime datetime default null;
```

```
mysql> select * from beanbook;
+-----+-----+-----+-----+-----+
| barcode | bookname | pubid | price | state |
+-----+-----+-----+-----+
| barcode1 | book1 | 1 | 10 | 已借出 |
| barcode2 | book2 | 1 | 10 | 在库 |
| barcode3 | book3 | 2 | 10 | 在库 |
| barcode4 | book4 | 1 | 10 | 在库 |
| barcode5 | book5 | 3 | 20 | 在库 |
| barcode6 | book6 | 5 | 20 | 在库 |
| barcode7 | book7 | 6 | 20 | 在库 |
| barcode8 | book8 | 4 | 30 | 在库 |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> alter table beanbook add storagetime date default null;
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> select * from beanbook;
+-----+-----+-----+-----+-----+-----+
| barcode | bookname | pubid | price | state | storagetime |
+-----+-----+-----+-----+-----+-----+
| barcode1 | book1 | 1 | 10 | 已借出 | NULL |
| barcode2 | book2 | 1 | 10 | 在库 | NULL |
| barcode3 | book3 | 2 | 10 | 在库 | NULL |
| barcode4 | book4 | 1 | 10 | 在库 | NULL |
| barcode5 | book5 | 3 | 20 | 在库 | NULL |
| barcode6 | book6 | 5 | 20 | 在库 | NULL |
| barcode7 | book7 | 6 | 20 | 在库 | NULL |
| barcode8 | book8 | 4 | 30 | 在库 | NULL |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

修改代码

- BookManager.createBook(BeaBook b)

```
1 public void createBook(BeaBook b) throws BaseException {
2
3
4     if (b.getBarcode() == null || "".equals(b.getBarcode()) ||
        b.getBarcode().length() > 20) {
5         throw new BusinessException("条码必须是1-20个字");
6     }
7     if (b.getBookname() == null || "".equals(b.getBookname()) ||
        b.getBookname().length() > 50) {
8         throw new BusinessException("图书名称必须是1-50个字");
9     }
10    Connection conn = null;
11    try {
12        conn = DBUtil.getConnection();
13        String sql = "select * from BeanBook where barcode=?";
14        java.sql.PreparedStatement pst = conn.prepareStatement(sql);
```



```

15         pst.setString(1, b.getBarcode());
16         java.sql.ResultSet rs = pst.executeQuery();
17         if (rs.next()) throw new BusinessException("条码已经被占用");
18         rs.close();
19         pst.close();
20         sql = "insert into
BeanBook(barcode,bookname,pubid,price,state,storagetime) values(?,?,?,?,'在
库',?)";//修改处
21         pst = conn.prepareStatement(sql);
22         pst.setString(1, b.getBarcode());
23         pst.setString(2, b.getBookname());
24         pst.setString(3, b.getPubid());
25         pst.setDouble(4, b.getPrice());
26         pst.setTimestamp(5, new
java.sql.Timestamp(System.currentTimeMillis()));//修改处
27         pst.execute();
28         pst.close();
29     } catch (SQLException e) {
30         e.printStackTrace();
31         throw new DbException(e);
32     } finally {
33         if (conn != null)
34             try {
35                 conn.close();
36             } catch (SQLException e) {
37                 // TODO Auto-generated catch block
38                 e.printStackTrace();
39             }
40     }
41 }

```

- BookManager.loadBook(String barcode)

```

1 public BeanBook loadBook(String barcode) throws DbException {
2     Connection conn = null;
3     try {
4         conn = DBUtil.getConnection();
5         String sql = "select
b.barcode,b.bookname,b.pubid,b.price,b.state,b.storagetime,p.publishername "
+ //修改处
6         " from beanbook b left outer join beanpublisher p on
(b.pubid=p.pubid)" +

```

```

7         " where b.barcode=? ";
8     java.sql.PreparedStatement pst = conn.prepareStatement(sql);
9     pst.setString(1, barcode);
10    java.sql.ResultSet rs = pst.executeQuery();
11    if (rs.next()) {
12        BeanBook b = new BeanBook();
13        b.setBarcode(rs.getString(1));
14        b.setBookname(rs.getString(2));
15        b.setPubid(rs.getString(3));
16        b.setPrice(rs.getDouble(4));
17        b.setState(rs.getString(5));
18        b.setStorageTime(rs.getDate(6)); //修改处
19        b.setPubName(rs.getString(7));
20        return b;
21    }
22    } catch (SQLException e) {
23        e.printStackTrace();
24        throw new DbException(e);
25    } finally {
26        if (conn != null)
27            try {
28                conn.close();
29            } catch (SQLException e) {
30                // TODO Auto-generated catch block
31                e.printStackTrace();
32            }
33    }
34    return null;
35 }

```

- BeanBook

```

1    package cn.edu.zucc.booklib.model;
2
3    import java.util.Date;
4
5    public class BeanBook {
6        private String barcode;
7        private String bookname;
8        private String pubid;
9        private double price;
10       private String state; //状态: 已借出,在库,已删除

```



```
11     private Date storageTime;
12
13     private String pubName;//出版社名称，在图书表中不存储名称，只存储出版社
    ID
14
15     public String getBarcode() {
16         return barcode;
17     }
18
19     public void setBarcode(String barcode) {
20         this.barcode = barcode;
21     }
22
23     public String getBookname() {
24         return bookname;
25     }
26
27     public void setBookname(String bookname) {
28         this.bookname = bookname;
29     }
30
31     public String getPubid() {
32         return pubid;
33     }
34
35     public void setPubid(String pubid) {
36         this.pubid = pubid;
37     }
38
39     public double getPrice() {
40         return price;
41     }
42
43     public void setPrice(double price) {
44         this.price = price;
45     }
46
47     public String getState() {
48         return state;
49     }
50
51     public void setState(String state) {
```

```

52         this.state = state;
53     }
54
55     //-----修改处-----
56     public void setStorageTime(Date storageTime) {
57         this.storageTime = storageTime;
58     }
59
60     public Date getStroageTime() {
61         return storageTime;
62     }
63     //-----
64
65     public String getPubName() {
66         return pubName;
67     }
68
69     public void setPubName(String pubName) {
70         this.pubName = pubName;
71     }
72 }

```

```
mysql> select * from beanbook;
```

barcode	bookname	pubid	price	state	storagetime
barcode1	book1	1	10	已借出	NULL
barcode10	book10	4	90	在庫	2022-05-13 21:30:08
barcode2	book2	1	10	在庫	NULL
barcode3	book3	2	10	在庫	NULL
barcode4	book4	1	10	在庫	NULL
barcode5	book5	3	20	在庫	NULL
barcode6	book6	5	20	在庫	NULL
barcode7	book7	6	20	在庫	NULL
barcode8	book8	4	30	在庫	NULL
barcode9	book9	1	10	在庫	2022-05-13 21:24:28

```
10 rows in set (0.00 sec)
```

```
mysql> |
```