# 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：

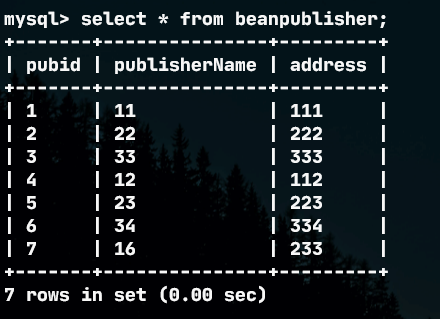
### 1、 利用PreparedStatement进行查询。

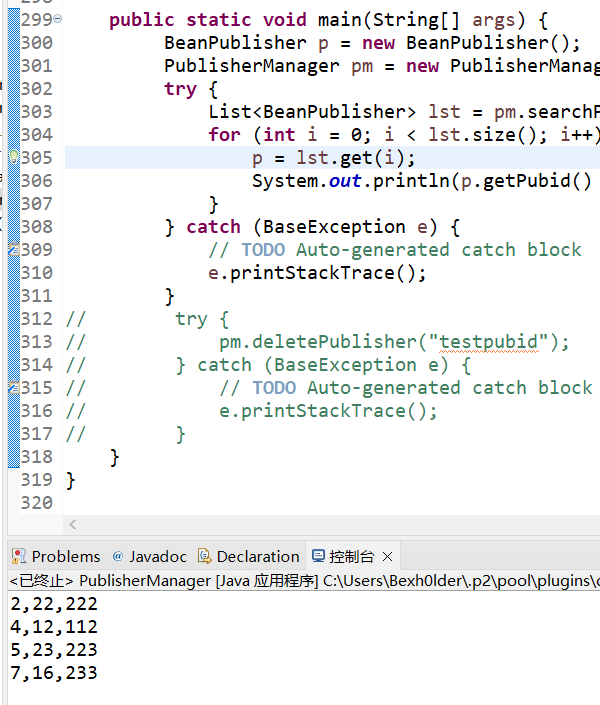
第一步：在PublisherManager类中增加方法public List<BeanPublisher> searchPublisher(String keyword) throws BaseException 方法，要求根据关键字在出版社表中查询满足条件的出版社（出版社名称或地址中包含参数中的关键字），参考loadAllPublisher()方法，将查询结果封装为List

第二步：在main函数中编写测试代码进行该方法的调用测试。

public List<BeanPublisher> searchPublisher(String keyword) throws BaseException{  
 List<BeanPublisher> result = new ArrayList<BeanPublisher>();  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select pubid,publisherName,address" +  
 " from beanpublisher";  
 if (keyword != null && !"".equals(keyword)) {  
 sql += " where publisherName like ? "  
 + "or address like ?";  
 }  
 sql += " order by pubid";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 if (keyword != null && !"".equals(keyword)) {  
 pst.setString(1, "%" + keyword + "%");  
 pst.setString(2, "%" + keyword + "%");  
 }  
 java.sql.ResultSet rs = pst.executeQuery();  
 while (rs.next()) {  
 BeanPublisher r = new BeanPublisher();  
 r.setPubid(rs.getString(1));  
 r.setPublisherName(rs.getString(2));  
 r.setAddress(rs.getString(3));  
 result.add(r);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
 return result;  
}

public static void main(String[] args) {  
 BeanPublisher p = new BeanPublisher();  
 PublisherManager pm = new PublisherManager();  
 try {  
 List<BeanPublisher> lst = pm.searchPublisher("2");  
 for (int i = 0; i < lst.size(); i++) {  
 p = lst.get(i);  
 System.out.println(p.getPubid() + "," + p.getPublisherName() + "," + p.getAddress());  
 }  
 } catch (BaseException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
// try {  
// pm.deletePublisher("testpubid");  
// } catch (BaseException e) {  
// // TODO Auto-generated catch block  
// e.printStackTrace();  
// }  
}





### 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】

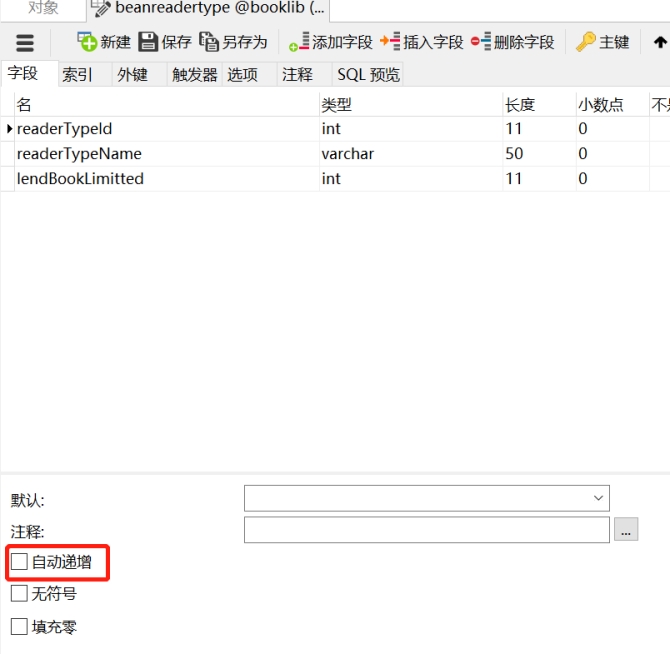
A、 写出替换的代码部分。

public void createPublisher(BeanPublisher p) throws BaseException {  
 if (p.getPubid() == null || "".equals(p.getPubid()) || p.getPubid().length() > 20) {  
 throw new BusinessException("出版社编号必须是1-20个字");  
 }  
 if (p.getPublisherName() == null || "".equals(p.getPublisherName()) || p.getPublisherName().length() > 50) {  
 throw new BusinessException("出版社名称必须是1-50个字");  
 }  
 if (p.getAddress() == null || "".equals(p.getAddress()) || p.getAddress().length() > 100) {  
 throw new BusinessException("出版地址必须是1-100个字");  
 }  
  
  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select \* from BeanPublisher where pubid=?";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setString(1, p.getPubid());  
 java.sql.ResultSet rs = pst.executeQuery();  
 if (rs.next()) throw new BusinessException("出版社编号已经被占用");  
 rs.close();  
 pst.close();  
 sql = "select \* from BeanPublisher where publisherName=?";  
 pst = conn.prepareStatement(sql);  
 pst.setString(1, p.getPublisherName());  
 rs = pst.executeQuery();  
 if (rs.next()) throw new BusinessException("出版社名称已经存在");  
 rs.close();  
 pst.close();  
 //通过PreparedStatement实现  
// sql = "insert into BeanPublisher(pubid,publisherName,address) values(?,?,?)";  
// pst = conn.prepareStatement(sql);  
// pst.setString(1, p.getPubid());  
// pst.setString(2, p.getPublisherName());  
// pst.setString(3, p.getAddress());  
// pst.execute();  
// pst.close();  
 //-----------------------------修改部分-------------------------------------  
 //通过Statement实现  
 sql = "insert into BeanPublisher(pubid,publisherName,address) "  
 + "values("+ p.getPubid() + "," + p.getPublisherName() + "," + p.getAddress() + ")";  
 java.sql.Statement st = conn.createStatement();  
 int i = st.executeUpdate(sql);  
 if(i <= 0) throw new BusinessException("插入失败");  
 //-------------------------------------------------------------------------------  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
}

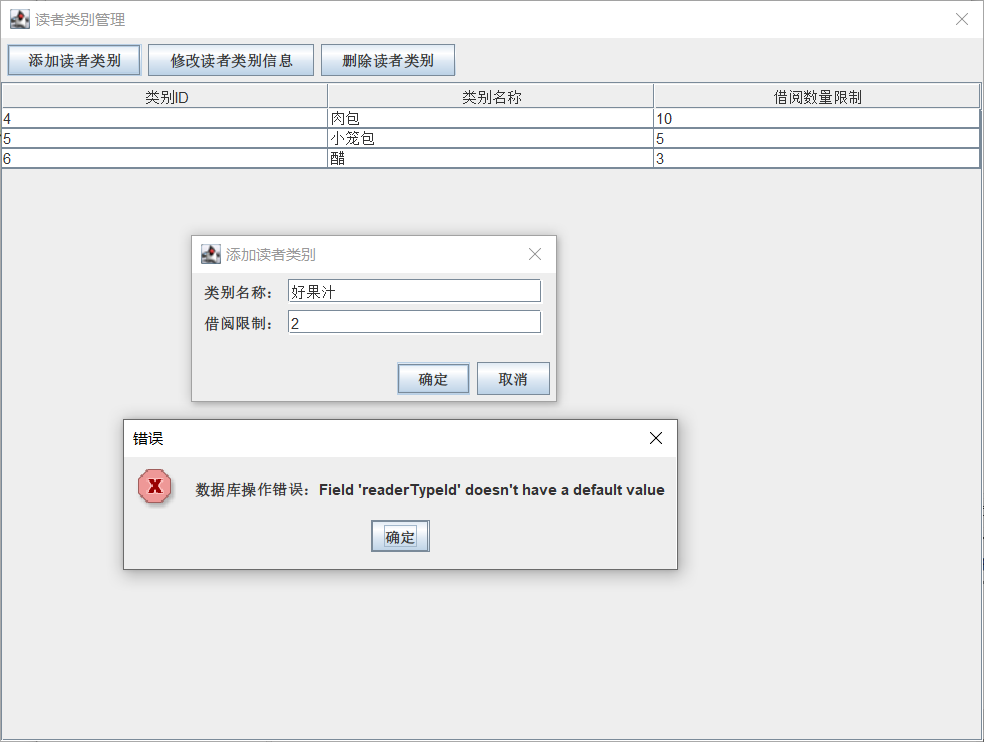


### 3、 利用insert语句添加数据时，未指定字段值处理。

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



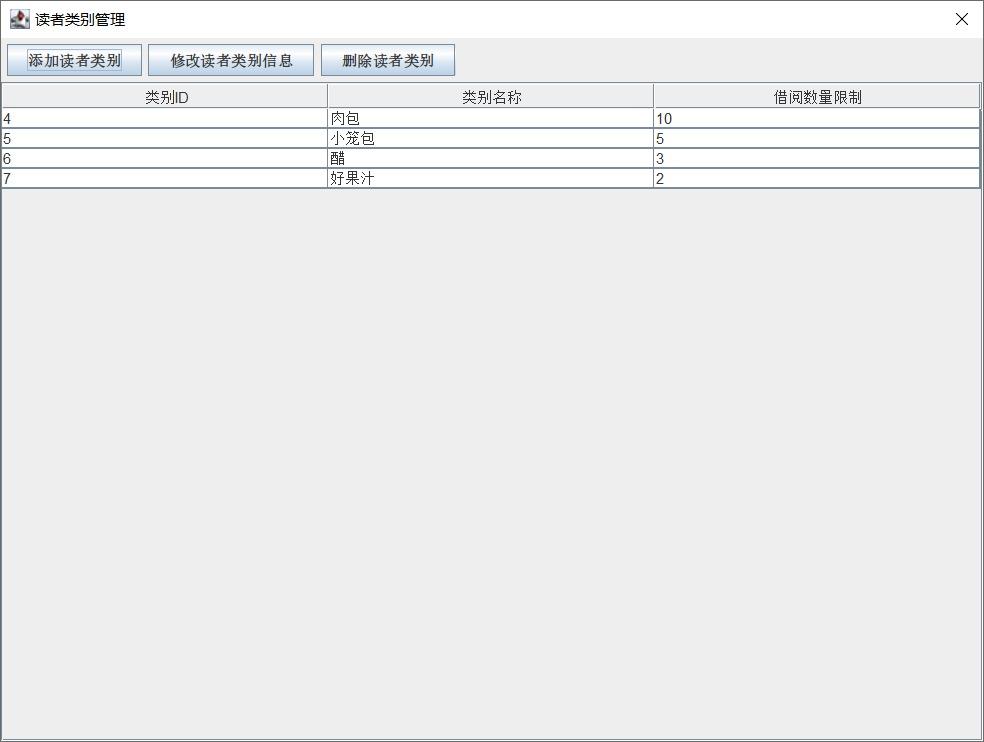
第二步：运行图书管理系统，打开读者类别管理界面，并尝试添加一个读者类别；系统将会报一个错误，请分析说明错误原因。



错误原因：报错内容信息是readerTypeld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readerTypeld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。

public void createReaderType(BeanReaderType rt) throws BaseException {  
 if (rt.getReaderTypeName() == null || "".equals(rt.getReaderTypeName()) || rt.getReaderTypeName().length() > 20) {  
 throw new BusinessException("读者类别名称必须是1-20个字");  
 }  
 if (rt.getLendBookLimitted() < 0 || rt.getLendBookLimitted() > 100) {  
 throw new BusinessException("借阅图书数量必须在0-100之间");  
 }  
 Connection conn = null;  
 int maxReaderTypeId = 0;//修改处  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select \* from BeanReaderType where readerTypeName=?";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setString(1, rt.getReaderTypeName());  
 java.sql.ResultSet rs = pst.executeQuery();  
 if (rs.next()) throw new BusinessException("读者类别名称已经被占用");  
 rs.close();  
 pst.close();  
 //---------------------------修改处--------------------------------  
 sql = "select max(readerTypeId) from beanreadertype";  
 pst = conn.prepareStatement(sql);  
 rs = pst.executeQuery();  
 if (rs.next()) maxReaderTypeId = rs.getInt(1);  
 //--------------------------------------------------------------------  
 sql = "insert into BeanReaderType(readerTypeId,readerTypeName,lendBookLimitted) values(?,?,?)";//修改处  
 pst = conn.prepareStatement(sql);  
 //---------------------------修改处--------------------------------  
 pst.setInt(1, maxReaderTypeId + 1);  
 pst.setString(2, rt.getReaderTypeName());  
 pst.setInt(3, rt.getLendBookLimitted());  
 //--------------------------------------------------------------------  
 pst.execute();  
 pst.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
}



### 4、 利用PreparedStatement对象进行数据修改。

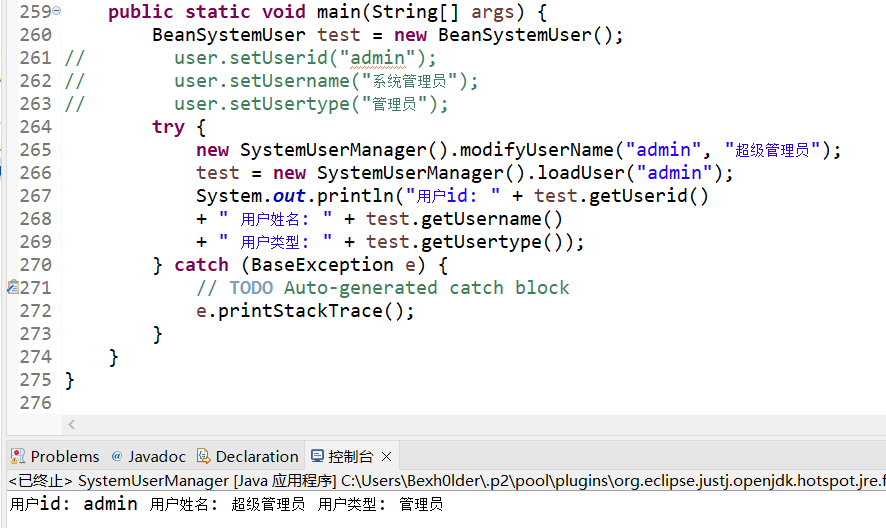
在SystemUserManager类中，新建一个modifyUserName方法，实现用户名称（username字段）的修改功能。并修改其main函数，将admin用户的名称改为：超级管理员。

【实验结果与分析】

A、 请提供方法代码和main函数代码。

public void modifyUserName(String userid, String newName) throws BaseException {  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select \* from BeanSystemUser where userid=?";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 java.sql.ResultSet rs = pst.executeQuery();  
 if (!rs.next()) throw new BusinessException("账号不存在");  
 rs.close();  
 pst.close();  
 sql = "update BeanSystemUser set username=? where userid=?";  
 pst = conn.prepareStatement(sql);  
 pst.setString(1, newName);  
 pst.setString(2, userid);  
 pst.execute();  
 pst.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
}

public static void main(String[] args) {  
 BeanSystemUser test = new BeanSystemUser();  
// user.setUserid("admin");  
// user.setUsername("系统管理员");  
// user.setUsertype("管理员");  
 try {  
 new SystemUserManager().modifyUserName("admin", "超级管理员");  
 test = new SystemUserManager().loadUser("admin");  
 System.out.println("用户id: " + test.getUserid()   
 + " 用户姓名: " + test.getUsername()   
 + " 用户类型: " + test.getUsertype());  
 } catch (BaseException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
}





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。

public boolean modifyUserName(String userid, String newName) throws BaseException {  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
// String sql = "select \* from BeanSystemUser where userid=?";  
// java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
// pst.setString(1, userid);  
// java.sql.ResultSet rs = pst.executeQuery();  
// if (!rs.next()) throw new BusinessException("账号不存在");  
// rs.close();  
// pst.close();  
// sql = "update BeanSystemUser set username=? where userid=?";  
// pst = conn.prepareStatement(sql);  
// pst.setString(1, newName);  
// pst.setString(2, userid);  
// pst.execute();  
// pst.close();  
 String sql = "update BeanSystemUser set username= \""+ newName + "\" where userid=\"" + userid + "\"";  
 java.sql.Statement st = conn.createStatement();  
 int i = st.executeUpdate(sql);  
 if(i <= 0) return false;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
 return true;  
}

public static void main(String[] args) {  
 BeanSystemUser test = new BeanSystemUser();  
 try {  
 List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();  
 testList = new SystemUserManager().loadAllUsers(true);  
 if(new SystemUserManager().modifyUserName("111", "超级管理员"))  
 System.out.println("修改成功");  
 else  
 System.out.println("修改失败");  
 for(int i = 0; i < testList.size(); ++i)  
 {  
 test = testList.get(i);  
 System.out.println("用户id: " + test.getUserid()   
 + " 用户姓名: " + test.getUsername()   
 + " 用户类型: " + test.getUsertype());  
 }  
 if(new SystemUserManager().modifyUserName("admin", "超级管理员"))  
 System.out.println("修改成功");  
 else  
 System.out.println("修改失败");  
 testList = new SystemUserManager().loadAllUsers(true);  
 for(int i = 0; i < testList.size(); ++i)  
 {  
 test = testList.get(i);  
 System.out.println("用户id: " + test.getUserid()   
 + " 用户姓名: " + test.getUsername()   
 + " 用户类型: " + test.getUsertype());  
 }  
 } catch (BaseException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
}

executeUpdate 的返回值是一个整数（int），指示受影响的行数（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



### 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。

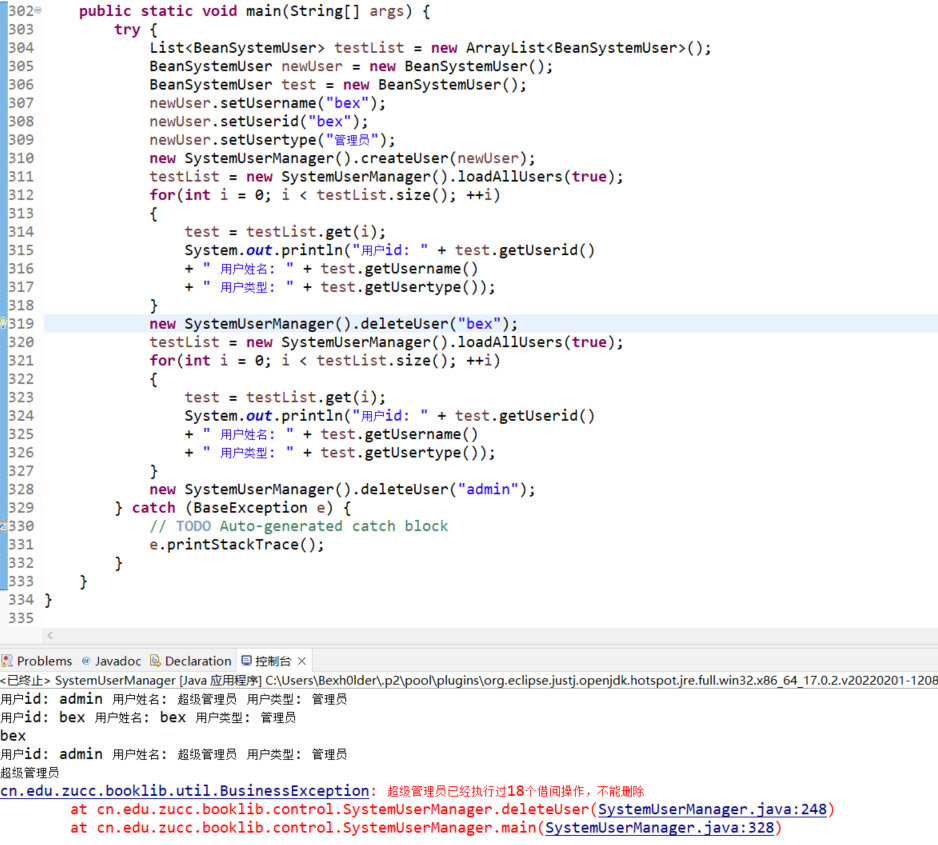
public void deleteUser(String userid) throws BaseException {  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select removeDate from BeanSystemUser where userid=?";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 java.sql.ResultSet rs = pst.executeQuery();  
 if (!rs.next()) throw new BusinessException("登陆账号不存在或已被删除");  
 rs.close();  
 pst.close();  
 //-----------------修改后的sql语句---------------------  
 sql = "delete from beansystemuser where userid=?";  
 //------------------------------------------------------------  
 pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 pst.execute();  
 pst.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
}



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。

public void deleteUser(String userid) throws BaseException {  
 Connection conn = null;  
 try {  
 conn = DBUtil.getConnection();  
 String sql = "select username from BeanSystemUser where userid=?";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 java.sql.ResultSet rs = pst.executeQuery();  
 if (!rs.next()) throw new BusinessException("账号不存在或已被删除");  
 String deleteName = rs.getString(1);  
 System.out.println(deleteName);  
 rs.close();  
 pst.close();  
 sql = "select count(\*) from beanbooklendrecord where lendOperUserid=?";  
 pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 rs = pst.executeQuery();  
 rs.next();  
 int n = rs.getInt(1);  
 if (n > 0) throw new BusinessException(deleteName + "已经执行过" + n + "个借阅操作，不能删除");  
 sql = "delete from beansystemuser where userid=?";  
 pst = conn.prepareStatement(sql);  
 pst.setString(1, userid);  
 pst.execute();  
 pst.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 throw new DbException(e);  
 } finally {  
 if (conn != null)  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
}

public static void main(String[] args) {  
 try {  
 List<BeanSystemUser> testList = new ArrayList<BeanSystemUser>();  
 BeanSystemUser newUser = new BeanSystemUser();  
 BeanSystemUser test = new BeanSystemUser();  
 newUser.setUsername("bex");  
 newUser.setUserid("bex");  
 newUser.setUsertype("管理员");  
 new SystemUserManager().createUser(newUser);  
 testList = new SystemUserManager().loadAllUsers(true);  
 for(int i = 0; i < testList.size(); ++i)  
 {  
 test = testList.get(i);  
 System.out.println("用户id: " + test.getUserid()   
 + " 用户姓名: " + test.getUsername()   
 + " 用户类型: " + test.getUsertype());  
 }  
 new SystemUserManager().deleteUser("bex");  
 testList = new SystemUserManager().loadAllUsers(true);  
 for(int i = 0; i < testList.size(); ++i)  
 {  
 test = testList.get(i);  
 System.out.println("用户id: " + test.getUserid()   
 + " 用户姓名: " + test.getUsername()   
 + " 用户类型: " + test.getUsertype());  
 }  
 new SystemUserManager().deleteUser("admin");  
 } catch (BaseException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
}

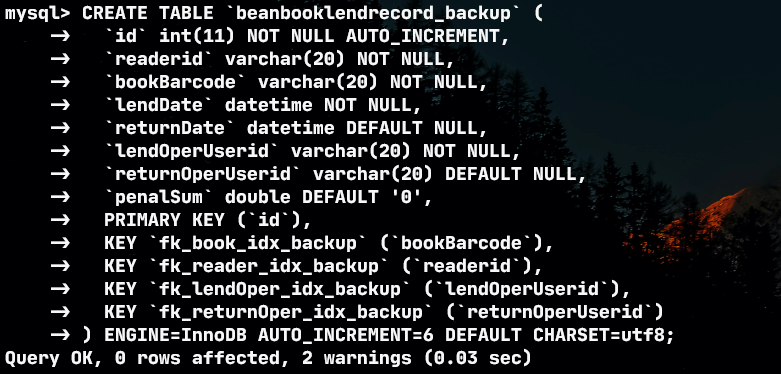


### 6、 **(修改)**在数据库中建立一张BeanBookLendRecord\_backup表，用于保存已经归还图书的借阅记录。其表结构与BeanBookLendRecord表完全一致。要求在借阅管理类中，增加方法，实现已经归还数据的备份功能（备份完成后，在原表中删除备份成功的数据）。提示：注意事务控制。

【实验结果与分析】

A 请提供备份表的建表语句

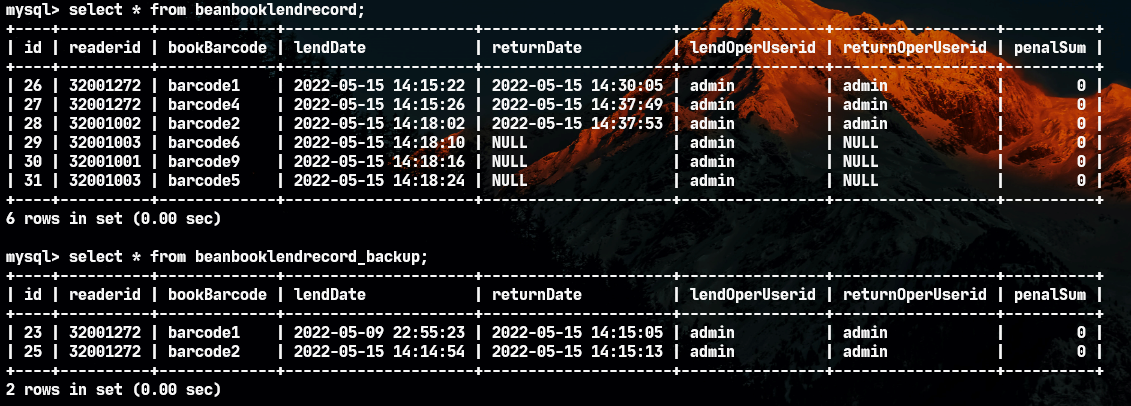
CREATE TABLE `beanbooklendrecord\_backup` (  
 `id` int(11) NOT NULL AUTO\_INCREMENT,  
 `readerid` varchar(20) NOT NULL,  
 `bookBarcode` varchar(20) NOT NULL,  
 `lendDate` datetime NOT NULL,  
 `returnDate` datetime DEFAULT NULL,  
 `lendOperUserid` varchar(20) NOT NULL,  
 `returnOperUserid` varchar(20) 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;



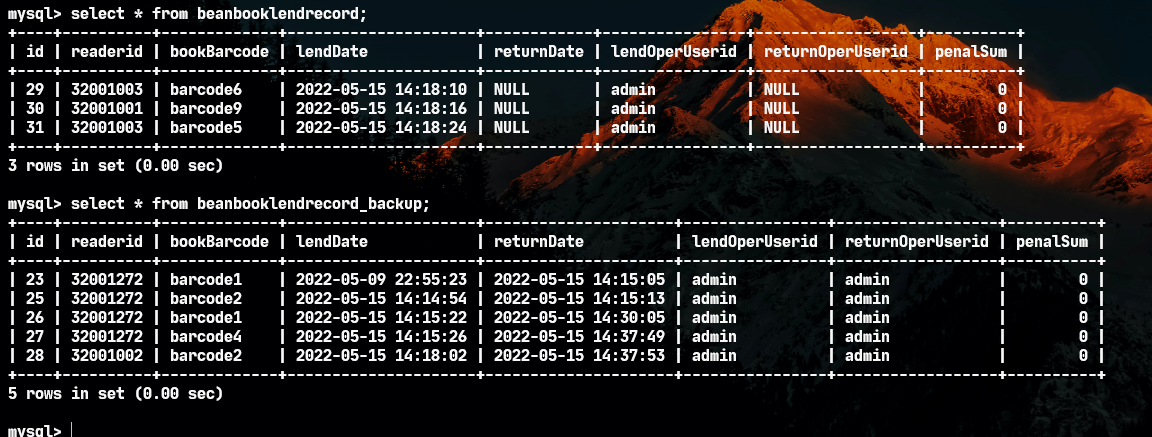
B 请提供备份函数代码

//三金指定very优雅版
  
 public void BackupReturnedRecord()throws DbException, SQLException{
  
 Connection conn = null;
  
 String sql = null;
  
 try {
  
 conn = DBUtil.getConnection();
  
 //关闭事务自动提交
  
 conn.setAutoCommit(false);
  
  
 sql = "insert beanbooklendrecord\_backup select \* from beanbooklendrecord where returnDate is not null";
  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);
  
 pst.execute();
  
 pst.close();
  
 sql = "delete from beanbooklendrecord where id in (select id from beanbooklendrecord\_backup)";
  
 pst = conn.prepareStatement(sql);
  
 pst.execute();
  
 pst.close();
  
 //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
  
 conn.commit();
  
 }catch(SQLException e) {
  
 //出错就回滚到第一步之前
  
 conn.rollback();
  
 e.printStackTrace();
  
 }finally {
  
 if (conn != null)
  
 try {
  
 conn.close();
  
 } catch (SQLException e) {
  
 // TODO Auto-generated catch block
  
 e.printStackTrace();
  
 }
  
 }
  
}

备份前



备份后

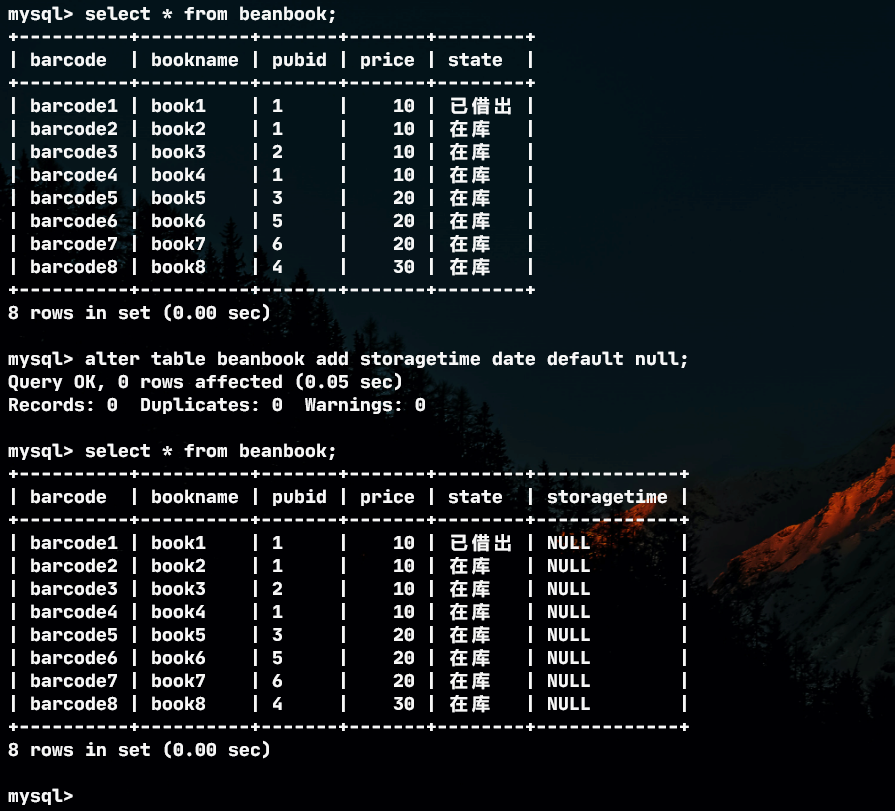


### 7、 如果需要记录图书的入库时间（需要包含时分秒），应如何修改数据库表结构和相关代码？

【实验结果与分析】

添加字段

alter table beanbook add storagetime datetime default null;



修改代码

* BookManager.createBook(BeanBook b)
* public void createBook(BeanBook b) throws BaseException {  
    
    
   if (b.getBarcode() == null || "".equals(b.getBarcode()) || b.getBarcode().length() > 20) {  
   throw new BusinessException("条码必须是1-20个字");  
   }  
   if (b.getBookname() == null || "".equals(b.getBookname()) || b.getBookname().length() > 50) {  
   throw new BusinessException("图书名称必须是1-50个字");  
   }  
   Connection conn = null;  
   try {  
   conn = DBUtil.getConnection();  
   String sql = "select \* from BeanBook where barcode=?";  
   java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
   pst.setString(1, b.getBarcode());  
   java.sql.ResultSet rs = pst.executeQuery();  
   if (rs.next()) throw new BusinessException("条码已经被占用");  
   rs.close();  
   pst.close();  
   sql = "insert into BeanBook(barcode,bookname,pubid,price,state,storagetime) values(?,?,?,?,'在库',?)";//修改处  
   pst = conn.prepareStatement(sql);  
   pst.setString(1, b.getBarcode());  
   pst.setString(2, b.getBookname());  
   pst.setString(3, b.getPubid());  
   pst.setDouble(4, b.getPrice());  
   pst.setTimestamp(5, new java.sql.Timestamp(System.currentTimeMillis()));//修改处  
   pst.execute();  
   pst.close();  
   } catch (SQLException e) {  
   e.printStackTrace();  
   throw new DbException(e);  
   } finally {  
   if (conn != null)  
   try {  
   conn.close();  
   } catch (SQLException e) {  
   // TODO Auto-generated catch block  
   e.printStackTrace();  
   }  
   }  
  }
* BookManager.loadBook(String barcode)
* public BeanBook loadBook(String barcode) throws DbException {  
   Connection conn = null;  
   try {  
   conn = DBUtil.getConnection();  
   String sql = "select b.barcode,b.bookname,b.pubid,b.price,b.state,b.storagetime,p.publishername " + //修改处  
   " from beanbook b left outer join beanpublisher p on (b.pubid=p.pubid)" +  
   " where b.barcode=? ";  
   java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
   pst.setString(1, barcode);  
   java.sql.ResultSet rs = pst.executeQuery();  
   if (rs.next()) {  
   BeanBook b = new BeanBook();  
   b.setBarcode(rs.getString(1));  
   b.setBookname(rs.getString(2));  
   b.setPubid(rs.getString(3));  
   b.setPrice(rs.getDouble(4));  
   b.setState(rs.getString(5));  
   b.setStorageTime(rs.getDate(6));//修改处  
   b.setPubName(rs.getString(7));  
   return b;  
   }  
   } catch (SQLException e) {  
   e.printStackTrace();  
   throw new DbException(e);  
   } finally {  
   if (conn != null)  
   try {  
   conn.close();  
   } catch (SQLException e) {  
   // TODO Auto-generated catch block  
   e.printStackTrace();  
   }  
   }  
   return null;  
  }
* BeanBook
* package cn.edu.zucc.booklib.model;  
    
  import java.util.Date;  
    
  public class BeanBook {  
   private String barcode;  
   private String bookname;  
   private String pubid;  
   private double price;  
   private String state;//状态：已借出,在库,已删除  
   private Date storageTime;  
     
   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  
    
   public String getBarcode() {  
   return barcode;  
   }  
    
   public void setBarcode(String barcode) {  
   this.barcode = barcode;  
   }  
    
   public String getBookname() {  
   return bookname;  
   }  
    
   public void setBookname(String bookname) {  
   this.bookname = bookname;  
   }  
    
   public String getPubid() {  
   return pubid;  
   }  
    
   public void setPubid(String pubid) {  
   this.pubid = pubid;  
   }  
    
   public double getPrice() {  
   return price;  
   }  
    
   public void setPrice(double price) {  
   this.price = price;  
   }  
    
   public String getState() {  
   return state;  
   }  
    
   public void setState(String state) {  
   this.state = state;  
   }  
     
   //-------------修改处-----------------  
   public void setStorageTime(Date storageTime) {  
   this.storageTime = storageTime;  
   }  
     
   public Date getStroageTime() {  
   return storageTime;  
   }  
   //-----------------------------------  
     
   public String getPubName() {  
   return pubName;  
   }  
    
   public void setPubName(String pubName) {  
   this.pubName = pubName;  
   }  
  }
* 