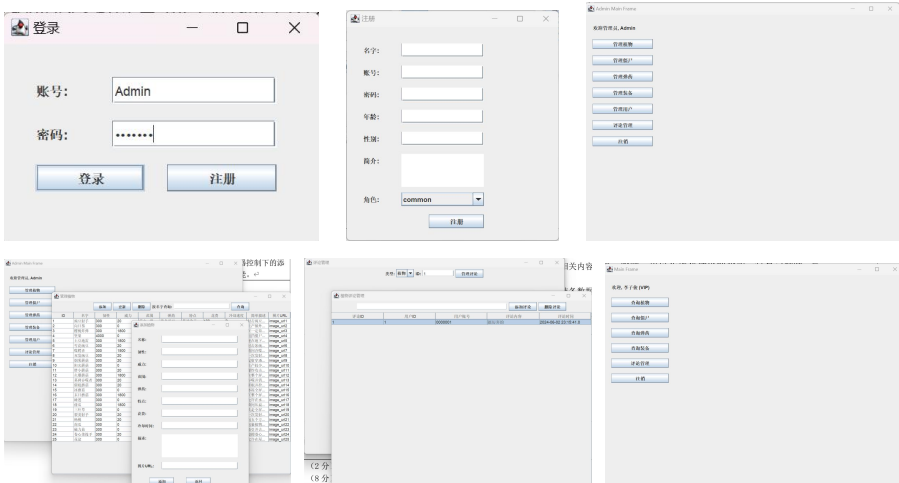
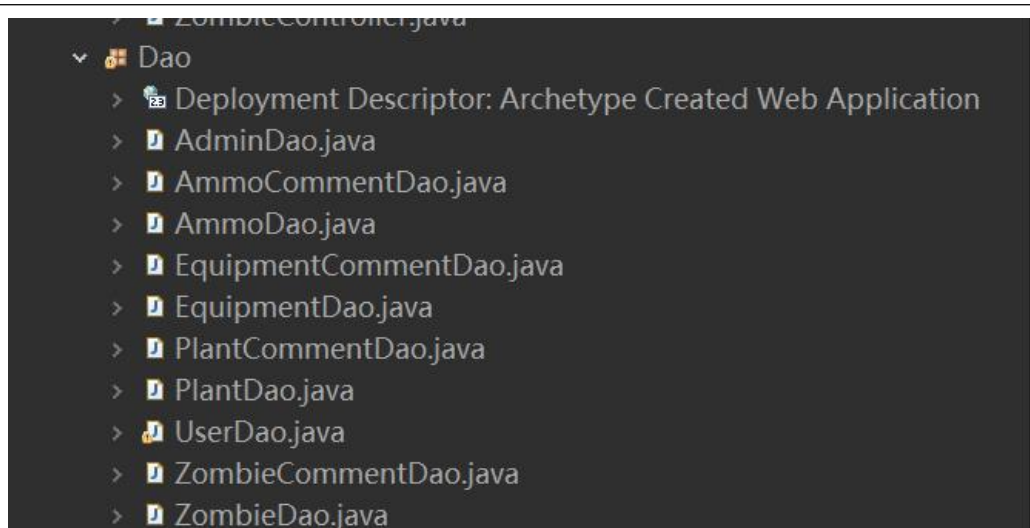


学号	2212000	姓名	宋奕纬	专业	信息安全、法学
项目名称	植物大战僵尸 (PvZ) 信息管理系统				
必备环境	1、Java Development Kit (JDK) 版本: java 22.0.1 2、MySQL 数据库 版本: MySQL 8.0.36 及以上 3、JDBC 驱动 版本: 适用于 MySQL 的 JDBC 驱动 4、集成开发环境 Eclipse (写 Java 项目), VSCode (写 MySQL) 5、Maven 用于管理 Java 项目的依赖关系和构建过程				
系统主要功能简介 (4 分)	近期, 杂交版植物大战僵尸火爆全网, 故萌生出写一个植物大战僵尸数据库管理系统的想法, 该系统的功能如下: 1、使用者分为管理员 (Admin) 和用户 (User) 2、系统实现了登录、注册等功能。 3、用户可以实现对四种内容的查询 (植物 (plant)、僵尸 (zombie)、弹药 (ammo)、装备 (equipment))。用户又分为普通用户 (Common) 与会员用户 (VIP), 普通用户只可以查看所有的信息, 会员用户可以实现通过名字来查询相关信息的信息。用户还可以针对特定的对象添加评论、查看评论。 4、管理员可以实现对用户数据、植物数据、僵尸数据、弹药数据、装备数据的查询、增加、更新、删除, 可以实现对评论的添加、查看与删除。 5、高级功能: 完成了作业要求中的含有事务应用的删除操作、触发器控制下的添加操作、存储过程控制下的更新操作、含有视图的查询操作等功能。				
系统主要页面截图 (6 分)					

	<div></div> <p>以上图片展示了注册、登录、错误注册界面，展示了管理员主界面与部分功能实现界面，展现了用户主界面与部分功能实现界面，并展示了VIP与普通用户查询的区别。</p>
--	--

2. 系统配置（10分）

说明		(2分)请说明系统配置情况（后台数据库，高级语言）； (8分)请使用连接串连接高级语言和数据库，并分析字符串的各个部分。			
配置 步骤 2分	DBMS	主要使用了VSCode进行数据库的编写，具体配置步骤如下：			
		1. 安装对应的插件：主要使用了MySQL插件（写SQL语句）、Database Client JDBC插件（可视化查看数据库）			
		2. 进行数据库的连接操作，通过插件与本地配置好的MySQL实现连接。			
	高级 语言	3. 安装插件Draw.io，绘制ER图			
		主要使用了EclipseIDE进行了Java的编写，具体配置步骤如下			
		1. Marven 框架的安装：在Eclipse中自动安装即可			
连接串 分析 (6分)	高级 语言	2. JDBC的配置：去官网下载“mysql-connector-j-8.4.0.zip”解压后将其中的“mysql-connector-j-8.4.0.jar”粘贴到项目中，右键添加到路径，即可在marven框架的约束文件pom.xml中自动配置。			
		3. 相关包的导入：在Eclipse直接导入即可。			
		序号	名称	功能说明	取值
		1	jdbcURL	JDBC连接URL,用于指定数据库服务器地址和数据库名	"jdbc:mysql://localhost:3306/pvz"
连接串代码 (截屏) (2分)	高级 语言	2	jdbcUsername	数据库连接用户名	"root"
		3	jdbcPassword	数据库连接用户密码	"syw5861668syw"
		1、连接串的定义			
		<pre>private String jdbcURL = "jdbc:mysql://localhost:3306/pvz"; private String jdbcUsername = "root"; private String jdbcPassword = "syw5861668syw";</pre>			
连接串代码 (截屏) (2分)	高级 语言	2、连接函数定义			
		<pre>protected Connection getConnection() { Connection connection = null; try { connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword); } catch (SQLException e) { e.printStackTrace(); } return connection; }</pre>			
		3、主要是通过Dao文件来实现连接，每一个dao文件调用连接函数进行连接（开始写时仅考虑了一个dao，导致了连接串与连接函数的重复定义，之后将针对这个情况进行优化）			



采取了两种方式实现了通过 java 来操作数据库

(1) 方法一：直接在 java 中定义 SQL 语句（适用于简单操作）

举例：添加植物

```
public class PlantDao {
    private String jdbcURL = "jdbc:mysql://localhost:3306/pvz";
    private String jdbcUsername = "root";
    private String jdbcPassword = "syw5861668syw";

    private static final String INSERT_PLANT_SQL = "INSERT INTO plant (name, toughness, power, 'range', ammo, features, cost, cooldown, 'description', image)";
    private static final String SELECT_PLANT_BY_ID = "SELECT * FROM plant WHERE id = ?";
    private static final String SELECT_ALL_PLANTS = "SELECT * FROM plant";
    private static final String DELETE_PLANT_SQL = "DELETE FROM plant WHERE id = ?";
    private static final String UPDATE_PLANT_SQL = "UPDATE plant SET name = ?, toughness = ?, power = ?, 'range' = ?, ammo = ?, features = ?, cost = ?, cool";
    private static final String SELECT_PLANTS_BY_NAME = "SELECT * FROM plant WHERE name LIKE ?";

    public PlantDao() {}

    protected Connection getConnection() {
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

    public void insertPlant(Plant plant) throws SQLException {
        try (Connection connection = getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(INSERT_PLANT_SQL)) {
            preparedStatement.setString(1, plant.getName());
            preparedStatement.setInt(2, plant.getToughness());
            preparedStatement.setInt(3, plant.getPower());
            preparedStatement.setString(4, plant.getRange());
            preparedStatement.setString(5, plant.getAmmo());
            preparedStatement.setString(6, plant.getFeatures());
            preparedStatement.setInt(7, plant.getCost());
            preparedStatement.setInt(8, plant.getCooldown());
            preparedStatement.setString(9, plant.getDescription());
            preparedStatement.setString(10, plant.getImageUrl());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

(2) 在 MySQL 中写好存储过程，在 Java 中直接调用

举例：删除用户

```
public boolean deleteUser(long id) throws SQLException {
    boolean rowDeleted;
    try (Connection connection = getConnection();
        CallableStatement callableStatement = connection.prepareCall("{call deleteUserWithComments(?)}")) {
        callableStatement.setLong(1, id);
        rowDeleted = callableStatement.executeUpdate() > 0;
    }
    return rowDeleted;
}
```

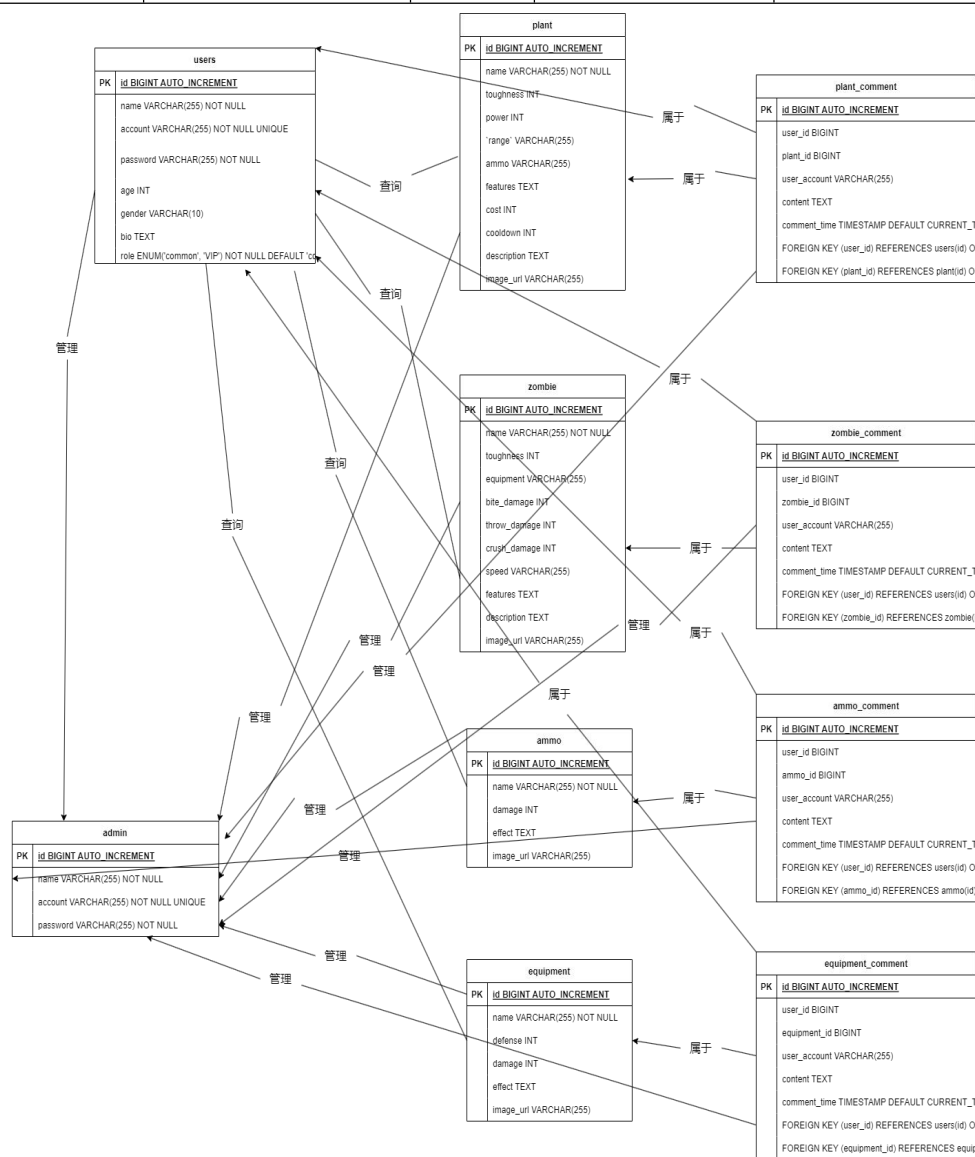
	<pre>Run Copy CREATE PROCEDURE deleteUserWithComments(IN p_user_id BIGINT) BEGIN DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN -- 回滚事务 ROLLBACK; END; -- 开始事务 START TRANSACTION; -- 删除用户评论 DELETE FROM plant_comment WHERE user_id = p_user_id; DELETE FROM zombie_comment WHERE user_id = p_user_id; DELETE FROM ammo_comment WHERE user_id = p_user_id; DELETE FROM equipment_comment WHERE user_id = p_user_id; -- 删除用户 DELETE FROM users WHERE id = p_user_id; -- 提交事务 COMMIT; END //</pre>
备注	<p>1、其他字段（JDBC 驱动类名、连接池最大值、超时时间等均以 connector 默认，未进行设置）</p> <p>2、存在连接函数、连接串重复定义，可以再建一个类实现连接（这样可能也会有一定好处，每次单独进行连接，使系统更加稳定）</p>

3. 数据库设计（14分）

说明	<p>（10分）按照数据表的创建顺序，依次给出所涉及数据表的信息，其中参照字段以“（字段1，字段2，……，字段n）”的形式给出，被参照字段以“表名（字段1，字段2，……，字段n）”的形式给出；</p> <p>（4分）一般DBMS都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。</p>				
数据表 (10)	创建顺序	数据表名称	主键	参照属性	被参照表及属性
	1	admin	id	无	无
	2	users	id	无	无
	3	plant	id	无	无
	4	zombie	id	无	无
	5	ammo	id	无	无
	6	equipment	id	无	无
	7	plant_comment	id	plant_id	plant(id)
				user_id	users(id, account)
				user_account	
	8	zombie_comment	id	zombie_id	zombie(id)
				user_id	

					users(id, account)
				user_account	
	9	ammo_comment	id	ammo_id	ammo(id)
				user_id	
					users(id, account)
	10	equipment_comment	id	equipment_id	equipment(id)
				user_id	
					users(id, account)
				user_account	

关系图
(4)



该图由插件 draw.io 生成，并手动增加了关系。

备注 无

4. 含有事务应用的删除操作（13 分）

说明	<p>（1 分）简要说明该操作所要完成的功能；</p> <p>（2 分）该操作会涉及的表（必须含有两张或两张以上的关系表，同时以“表名”的形式给出）</p> <p>（1 分）表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>（1 分）删除条件涉及的字段描述（以“表名. 属性=? ”形式给出）</p> <p>（4 分）实现该操作的关键代码（高级语言、SQL），截图即可；（其中如果删除语句中不包含任何形式的事务应用将扣除 3 分）</p> <p>（4 分）如何执行该操作，按所述方法能够正常演示程序则给分。</p>	
功能描述 （1 分）	<p>用户与评论息息相关，删除用户时要删除对应用户的全部评论。</p> <p>该操作所要完成的功能是删除用户时，同时删除该用户在植物评论、僵尸评论、弹药评论和装备评论中的所有评论记录。</p>	
涉及的表 （2 分）	users、plant_comment、zombie_comment、ammo_comment、equipment_comment	
表连接涉及字段 （1 分）	<p>users.id = plant_comment.user_id users.id = zombie_comment.user_id users.id = ammo_comment.user_id users.id = equipment_comment.user_id</p> <p>（也可以用：</p> <p>users.account = plant_comment.user_account users.account = zombie_comment.user_account users.account = ammo_comment.user_account users.account = equipment_comment.user_account</p> <p>但是一开始在表中就已经定义好了外键级联，用的是 user 的 id，故采用了 id）</p>	
删除条件字段描述 （1 分）	字段	规则
	users.id = ?	plant_comment.user_id、zombie_comment.user_id、ammo_comment.user_id、equipment_comment.user_id 分别是这几张表的外键。要求删掉用户时，也删掉 user_id 与删掉用户 id 相等的的评论。
代码 （4 分）	<p>这里有一些多余，在开始定义表时就已经加入了外键级联，之后定义删除存储过程时又重新根据字段进行了删除（但更适用于普遍情况，如果 user_id 不是外键也可以实现删除）</p> <p>表定义时加入外键：</p>	

分)

```
▷ Run | New Tab | Copy
CREATE TABLE plant_comment (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  plant_id BIGINT,
  user_account VARCHAR(255),
  content TEXT,
  comment_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (plant_id) REFERENCES plant(id) ON DELETE CASCADE
);
▷ Run | New Tab | Copy
CREATE TABLE zombie_comment (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  zombie_id BIGINT,
  user_account VARCHAR(255),
  content TEXT,
  comment_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (zombie_id) REFERENCES zombie(id) ON DELETE CASCADE
);
▷ Run | New Tab | Copy
CREATE TABLE ammo_comment (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  ammo_id BIGINT,
  user_account VARCHAR(255),
  content TEXT,
  comment_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (ammo_id) REFERENCES ammo(id) ON DELETE CASCADE
);
▷ Run | New Tab | Copy
CREATE TABLE equipment_comment (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  equipment_id BIGINT,
  user_account VARCHAR(255),
  content TEXT,
  comment_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (equipment_id) REFERENCES equipment(id) ON DELETE CASCADE
);
```

定义带事务的删除存储过程:

```
--含有事务应用的删除操作:
▷ Run | New Tab
DELIMITER //

▷ Run | Copy
CREATE PROCEDURE deleteUserWithComments(
  IN p_user_id BIGINT
)
BEGIN
  DECLARE EXIT HANDLER FOR SQL_EXCEPTION
  BEGIN
    -- 回滚事务
    ROLLBACK;
  END;

  -- 开始事务
  START TRANSACTION;

  -- 删除用户评论(在表的定义写了外键, 这里不写也行)
  DELETE FROM plant_comment WHERE user_id = p_user_id;
  DELETE FROM zombie_comment WHERE user_id = p_user_id;
  DELETE FROM ammo_comment WHERE user_id = p_user_id;
  DELETE FROM equipment_comment WHERE user_id = p_user_id;

  -- 删除用户
  DELETE FROM users WHERE id = p_user_id;

  -- 提交事务
  COMMIT;
END //

▷ Run | New Tab
DELIMITER ;
```

在 UserDao 中的调用:

```
public boolean deleteUser(long id) throws SQLException {
  boolean rowDeleted;
  try (Connection connection = getConnection();
      CallableStatement callableStatement = connection.prepareCall("{call deleteUserWithComments(?)}")) {
    callableStatement.setLong(1, id);
    rowDeleted = callableStatement.executeUpdate() > 0;
  }
  return rowDeleted;
}
```

(其他部分略去)

程序演示（4分）

一开始的用户与评论：

管理用户							
添加 更新 删除 按名字查询: 查询							
ID	名字	账号	密码	年龄	性别	个性签名	角色
1	刘思灼	0000001	password1	25	男	喜欢打篮球	common
2	李子俊	0000002	password2	30	女	喜欢读书	VIP
3	姬耀飞	0000003	password3	22	男	喜欢游泳	common
4	宋茂凯	0000004	password4	28	女	喜欢旅行	VIP
5	陈俊林	0000005	password5	26	男	喜欢音乐	common
6	关忠吉	0000006	password6	24	女	喜欢看电影	VIP
7	卢东飞	0000007	password7	27	男	喜欢健身	common
8	左祥宇	0000008	password8	32	女	喜欢烹饪	VIP
9	焦钰程	0000009	password9	29	男	喜欢跑步	common
10	刘欣君	0000010	password10	23	女	喜欢画画	VIP
11	宋永坤	0000011	password11	31	男	喜欢摄影	common
12	陈雨欣	0000012	password12	28	女	喜欢写作	VIP
13	贾文浩	0000013	password13	26	男	喜欢集邮	common
14	王韵之	0000014	password14	24	女	喜欢舞蹈	VIP
15	李成凯	0000015	password15	29	男	喜欢钓鱼	common
16	陈星宇	0000016	password16	33	女	喜欢园艺	VIP
17	朱振洋	0000017	password17	21	男	喜欢露营	common
18	刘曹雨	0000018	password18	30	女	喜欢烘焙	VIP
19	杨硕	0000019	password19	27	男	喜欢健身	common
20	姬耀康	0000020	password20	27	男	喜欢睡觉	common
21	李登科	0000021	password21	27	男	喜欢吃饭	common
22	梅宇皓	0000022	password22	26	女	喜欢瑜伽	VIP
23	李美丽	0000028	11223344	24	female	我爱读书	VIP
24	鹏哥	0000037	lovelyayaj	15	男	好吃	common

植物评论管理				
添加评论 删除评论				
评论ID	用户ID	用户账号	评论内容	评论时间
2	1	0000001	植物真有意思	2024-06-03 16:11:10.0
3	1	0000001	绿萝真是版本之子	2024-06-03 16:11:18.0
4	1	0000001	太好玩了	2024-06-03 16:11:24.0
5	1	0000001	感觉每个植物的实力都很强	2024-06-03 16:11:47.0
6	4	0000004	确实啊	2024-06-03 16:13:19.0
7	4	0000004	确实好用	2024-06-03 16:13:24.0

现在删掉进行过评论的 ID 为 1 的用户：

管理用户

添加 更新 删除 按名字查询: 刘 查询

ID	名字	账号	密码	年龄	性别	个性签名	角色
1	刘思灼	0000001	password1	25	男	喜欢打篮球	common
10	刘候君	0000010	password10	23	女	喜欢画画	VIP
18	刘曹雨	0000018	password18	30	女	喜欢烘焙	VIP

删除确认
确认删除用户 ID 为 1 的用户吗?
是(Y) 否(N)

现在的用户与评论：

管理用户

添加

更新

删除

按名字查询:

查询

ID	名字	账号	密码	年龄	性别	个性签名	角色
2	李子俊	0000002	password2	30	女	喜欢读书	VIP
3	谢鹏飞	0000003	password3	22	男	喜欢游泳	common
4	宋思冉	0000004	password4	28	女	喜欢旅行	VIP
5	陈俊林	0000005	password5	26	男	喜欢音乐	common
6	关忠吉	0000006	password6	24	女	喜欢看电影	VIP
7	卢振飞	0000007	password7	27	男	喜欢健身	common
8	左洋宇	0000008	password8	32	女	喜欢烹饪	VIP
9	焦钰程	0000009	password9	29	男	喜欢跑步	common
10	刘敏君	0000010	password10	23	女	喜欢画画	VIP
11	宋成鹏	0000011	password11	31	男	喜欢摄影	common
12	陈雨辰	0000012	password12	28	女	喜欢写作	VIP
13	姜文清	0000013	password13	26	男	喜欢集邮	common
14	王杨之	0000014	password14	24	女	喜欢跳舞	VIP
15	李嘉凯	0000015	password15	29	男	喜欢钓鱼	common
16	陈星宇	0000016	password16	33	女	喜欢园艺	VIP
17	宋振洋	0000017	password17	21	男	喜欢烹饪	common
18	刘青雨	0000018	password18	30	女	喜欢唱歌	VIP
19	杨照	0000019	password19	27	男	喜欢极限	common
20	靳冠朋	0000020	password20	27	男	喜欢睡觉	common
21	李登科	0000021	password21	27	男	喜欢电竞	common
22	杨宇坤	0000022	password22	26	女	喜欢瑜伽	VIP
23	李美丽	0000028	11223344	24	female	我爱读书	VIP
24	顾司	0000037	lovelyjaql	15		好吃	common

植物评论管理

添加评论

删除评论

评论ID	用户ID	用户账号	评论内容	评论时间
6	4	0000004	很强烈	2024-06-03 16:13:19.0
7	4	0000004	非常好用	2024-06-03 16:13:24.0

我们可以看到删除 1 号用户的同时也删掉了 1 号用户的评论。

备注无。

5. 触发器控制下的添加操作（20 分）

说明	(1 分) 简要说明该操作所要完成的功能; (2 分) 简要说明该触发器所要完成的功能 (1 分) 该操作会涉及的表 (以 “表名” 的形式给出)。 (2 分) 该操作输入数据以及输入数据应该满足的条件, 如: 数值范围、是否为空; (6 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可; (8 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。	
功能描述 (1 分)	为确保系统的安全性, 用户注册或管理员在添加用户时, 不能让用户获得管理员权限。 本系统中用户不能使用账号 “Admin” 进行注册, 也不使用已经存在的账号进行注册。 管理员在进行用户添加时不能使用 “Admin” 作为用户账号, 不能使用已经存在的账号作为用户账号。	
触发器描述 (2 分)	触发器 1 在用户注册或管理员添加用户时触发, 如果尝试使用 “Admin” 作为账号名称, 将触发错误, 防止用户使用 “Admin” 作为账号。 触发器 2 在用户注册或管理员添加用户时触发, 如果尝试使用已经存在的账号名称, 将触发错误, 防止重复注册相同的账号。	
涉及的表 (1 分)	users (name, account, password, age, gender, bio, role)	
输入数据 (2 分)	字段	规则
	name	字符串, 不能为空
	account	字符串, 不能为空, 不能为 Admin, 不能与已有账号重复
	password	字符串, 不能为空
	age	整数, 可以为空
	gender	字符串, 可以为空
	bio	文本, 可以为空
	role	枚举值 (common 和 VIP), 默认为 common, 不能为空
插入操作源码 (3 分)	分为两部分: UserDao 中与数据库存储过程进行连接, UserService、UserController 中具体实现, ui 界面进行使用: 1、UserDao.java 截图 实现了对存储过程(插入用户)的调用	

```

public void insertUser(User user) throws SQLException {
    String sql = "{CALL insertUser(?, ?, ?, ?, ?, ?, ?)}";
    try (Connection connection = getConnection();
        CallableStatement callableStatement = connection.prepareCall(sql)) {
        callableStatement.setString(1, user.getName());
        callableStatement.setString(2, user.getAccount());
        callableStatement.setString(3, user.getPassword());
        callableStatement.setInt(4, user.getAge());
        callableStatement.setString(5, user.getGender());
        callableStatement.setString(6, user.getBio());
        callableStatement.setString(7, user.getRole());
        callableStatement.executeUpdate();
    } catch (SQLException e) {
        throw new SQLException(e);
    }
}

```

2、UserService.java 截图

生成弹窗中的字符串，

触发器抛出“Account cannot be Admin”，返回值为“不能使用‘Admin’作为账号名称”；

触发器抛出“Account already exists”，返回值为“账号已存在”。

```
public String registerUser(User user) {
    try {
        // 直接调用存储过程插入用户
        userDao.insertUser(user);
    } catch (SQLException e) {
        // 检查是否是触发器抛出的错误
        if (e.getMessage().contains("Account cannot be Admin")) {
            return "不能使用 'Admin' 作为账号名称";
        } else if (e.getMessage().contains("Account already exists")) {
            return "账号已存在";
        } else {
            e.printStackTrace();
            return "注册失败";
        }
    }
    return "注册成功";
}
```

3、UserController.java 截图

```
public String registerUser(User user) {
    return userService.registerUser(user);
}
```

4、UI 界面

AddUserFrame.java

```
package ui;

import javax.swing.*;

public class AddUserFrame extends JFrame {
    private JTextField nameField;
    private JTextField accountField;
    private JPasswordField passwordField;
    private JTextField ageField;
    private JTextField genderField;
    private JTextArea bioField;
    private JComboBox<String> roleComboBox;
    private JButton addButton;
    private UserController userController;

    public AddUserFrame(UserController userController) {
        this.userController = userController;
        setTitle("添加用户");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(null);

        JLabel nameLabel = new JLabel("姓名:");
        nameLabel.setBounds(30, 30, 60, 25);
        add(nameLabel);

        nameField = new JTextField();
        nameField.setBounds(100, 30, 150, 25);
        add(nameField);

        JLabel accountLabel = new JLabel("账号:");
        accountLabel.setBounds(30, 70, 60, 25);
        add(accountLabel);

        accountField = new JTextField();
        accountField.setBounds(100, 70, 150, 25);
        add(accountField);

        JLabel passwordLabel = new JLabel("密码:");
        passwordLabel.setBounds(30, 110, 60, 25);
        add(passwordLabel);

        passwordField = new JPasswordField();
        passwordField.setBounds(100, 110, 150, 25);
        add(passwordField);

        JLabel ageLabel = new JLabel("年龄:");
        ageLabel.setBounds(30, 150, 60, 25);
        add(ageLabel);

        ageField = new JTextField();
        ageField.setBounds(100, 150, 150, 25);
        add(ageField);
    }
}
```

```

64 JLabel genderLabel = new JLabel("性别:");
65 genderLabel.setBounds(30, 190, 60, 25);
66 add(genderLabel);
67
68 genderField = new JTextField();
69 genderField.setBounds(100, 190, 150, 25);
70 add(genderField);
71
72 JLabel bioLabel = new JLabel("年龄:");
73 bioLabel.setBounds(30, 230, 60, 25);
74 add(bioLabel);
75
76 bioField = new JTextArea();
77 bioField.setBounds(100, 230, 150, 60);
78 add(bioField);
79
80 JLabel roleLabel = new JLabel("角色:");
81 roleLabel.setBounds(30, 300, 60, 25);
82 add(roleLabel);
83
84 String[] roles = { "Common", "VIP" };
85 roleComboBox = new JComboBox<>(roles);
86 roleComboBox.setBounds(100, 300, 150, 25);
87 add(roleComboBox);
88
89 JButton registerButton = new JButton("注册");
90 registerButton.setBounds(150, 340, 100, 25);
91 add(registerButton);
92
93 registerButton.addActionListener(new ActionListener() {
94     @Override
95     public void actionPerformed(ActionEvent e) {
96         String name = nameField.getText();
97         String account = accountField.getText();
98         String password = new String(passwordField.getPassword());
99         int age = Integer.parseInt(ageField.getText());
100        String gender = genderField.getText();
101        String bio = bioField.getText();
102        String role = (String) roleComboBox.getSelectedItem();
103
104        User user = new User();
105        user.setName(name);
106        user.setAccount(account);
107        user.setPassword(password);
108        user.setAge(age);
109        user.setGender(gender);
110        user.setBio(bio);
111        user.setRole(role);
112
113        String result = userController.registerUser(user);
114        if ("注册成功".equals(result)) {
115            JOptionPane.showMessageDialog(null, "添加成功", "成功", JOptionPane.INFORMATION_MESSAGE);
116            dispose();
117        } else {
118            JOptionPane.showMessageDialog(null, result, "错误", JOptionPane.ERROR_MESSAGE);
119        }
120    }
121 });

```

RegisterFrame. java

```

1 package ui;
2
3 import javax.swing.*;
4
5 import controller.UserController;
6 import model.User;
7
8 import java.awt.event.ActionEvent;
9 import java.awt.event.ActionListener;
10
11
12
13 public class RegisterFrame extends JFrame {
14     private JTextField nameField;
15     private JTextField accountField;
16     private JPasswordField passwordField;
17     private JTextField ageField;
18     private JTextArea genderField;
19     private JTextArea bioField;
20     private JComboBox<String> roleComboBox;
21     private JButton registerButton;
22     private UserController userController;
23
24     public RegisterFrame(UserController userController) {
25         this.userController = userController;
26         setTitle("注册");
27         setSize(400, 300);
28         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
29         setLocationRelativeTo(null);
30         setLayout(null);
31
32         JLabel nameLabel = new JLabel("名字:");
33         nameLabel.setBounds(30, 30, 60, 25);
34         add(nameLabel);
35
36         nameField = new JTextField();
37         nameField.setBounds(100, 30, 150, 25);
38         add(nameField);
39
40         JLabel accountLabel = new JLabel("账号:");
41         accountLabel.setBounds(30, 70, 60, 25);
42         add(accountLabel);
43
44         accountField = new JTextField();
45         accountField.setBounds(100, 70, 150, 25);
46         add(accountField);
47
48         JLabel passwordLabel = new JLabel("密码:");
49         passwordLabel.setBounds(30, 110, 60, 25);
50         add(passwordLabel);
51
52         passwordField = new JPasswordField();
53         passwordField.setBounds(100, 110, 150, 25);
54         add(passwordField);
55
56         JLabel ageLabel = new JLabel("年龄:");
57         ageLabel.setBounds(30, 150, 60, 25);
58         add(ageLabel);

```

	<pre> 66 add(genderLabel); 67 68 genderField = new JTextField(); 69 genderField.setBounds(100, 190, 150, 25); 70 add(genderField); 71 72 JLabel bioLabel = new JLabel("简介:"); 73 bioLabel.setBounds(30, 230, 60, 25); 74 add(bioLabel); 75 76 bioField = new JTextArea(); 77 bioField.setBounds(100, 230, 150, 60); 78 add(bioField); 79 80 JLabel roleLabel = new JLabel("角色:"); 81 roleLabel.setBounds(30, 300, 60, 25); 82 add(roleLabel); 83 84 String[] roles = { "common", "VIP" }; 85 roleComboBox = new JComboBox<>(roles); 86 roleComboBox.setBounds(100, 300, 150, 25); 87 add(roleComboBox); 88 89 registerButton = new JButton("注册"); 90 registerButton.setBounds(150, 340, 100, 25); 91 add(registerButton); 92 registerButton.addActionListener(new ActionListener() { 93 @Override 94 public void actionPerformed(ActionEvent e) { 95 String name = nameField.getText(); 96 String account = accountField.getText(); 97 String password = new String(passwordField.getPassword()); 98 int age = Integer.parseInt(ageField.getText()); 99 String gender = genderField.getText(); 100 String bio = bioField.getText(); 101 String role = (String) roleComboBox.getSelectedItem(); 102 103 User user = new User(); 104 user.setName(name); 105 user.setAccount(account); 106 user.setPassword(password); 107 user.setAge(age); 108 user.setGender(gender); 109 user.setBio(bio); 110 user.setRole(role); 111 112 String result = userController.registerUser(user); 113 if ("注册成功".equals(result)) { 114 JOptionPane.showMessageDialog(null, "注册成功", "成功", JOptionPane.INFORMATION_MESSAGE); 115 dispose(); 116 } else { 117 JOptionPane.showMessageDialog(null, result, "错误", JOptionPane.ERROR_MESSAGE); 118 } 119 } 120 }); 121 } 122 } </pre>
	在注册与管理员添加中都调用了 registerUser，实现弹窗对字符串的输出。
触发器源码（3分）	写了两个触发器，一个用于判断输入的账号是不是 Admin，一个用于判断账号是否已经存在。

--触发器控制下的添加操作

▷ Run | New Tab

DELIMITER //

▷ Run | Copy

CREATE TRIGGER before_insert_user

BEFORE INSERT ON users

FOR EACH ROW

BEGIN

-- 检查账号是否为 'Admin'

IF NEW.account = 'Admin' THEN

| SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Account cannot be Admin';

END IF;

END //

▷ Run | Copy

CREATE TRIGGER before_insert_user2

BEFORE INSERT ON users

FOR EACH ROW

BEGIN

-- 检查账号是否已经存在

DECLARE account_count INT DEFAULT 0;

SELECT COUNT(*) INTO account_count FROM users WHERE account = NEW.account;

IF account_count > 0 THEN

| SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Account already exists';

END IF;

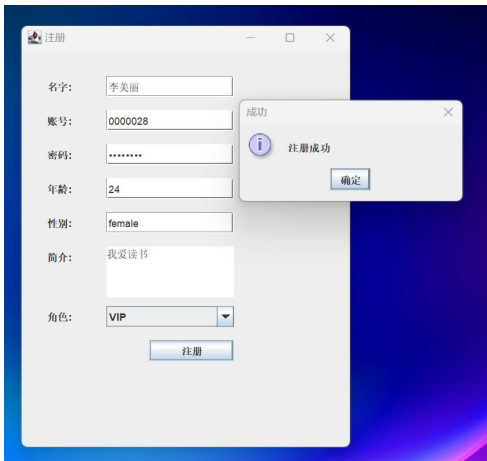
END //

▷ Run | New Tab

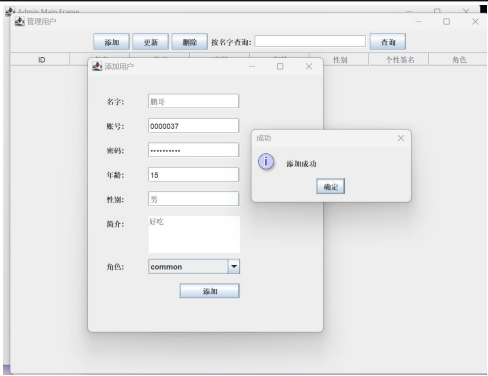
DELIMITER ;

说明：不违背触发器能够执行插入操作。

程序
演示
(4
分)



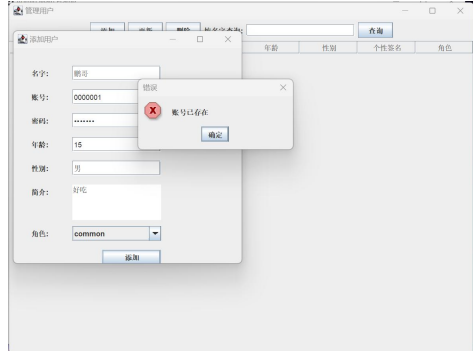
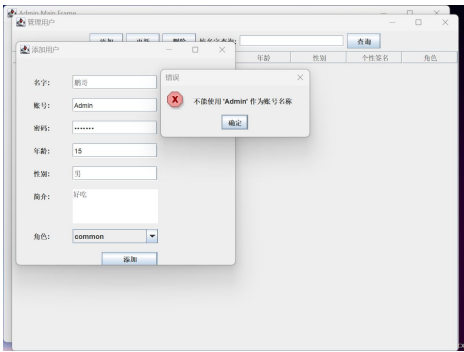
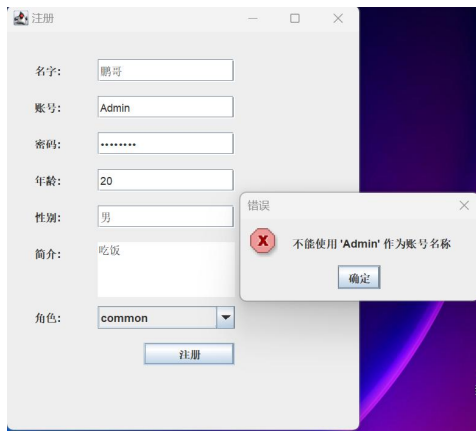
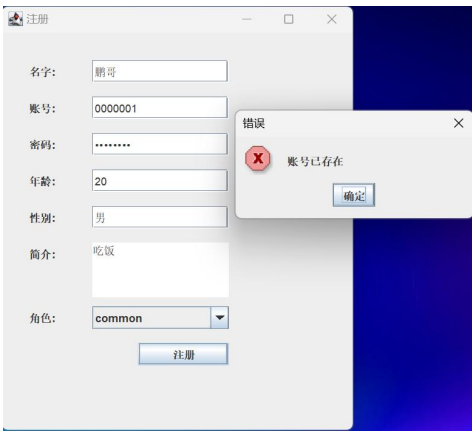
	Q	id bigint	name varchar(255)	account varchar(255)	password varchar(255)	age int	gender varchar(10)	bio text	role enum('common','VIP')
> 1	1		刘明约	0000001	password1	25	男	喜欢打篮球	common
> 2	2		李子俊	0000002	password2	30	女	喜欢读书	VIP
> 3	3		陈耀飞	0000003	password3	22	男	喜欢游泳	common
> 4	4		宋瑞禹	0000004	password4	28	女	喜欢旅行	VIP
> 5	5		易松林	0000005	password5	26	男	喜欢音乐	common
> 6	6		关组合	0000006	password6	24	女	喜欢看电影	VIP
> 7	7		卢乐飞	0000007	password7	27	男	喜欢健身	common
> 8	8		左泽宇	0000008	password8	32	女	喜欢烹饪	VIP
> 9	9		焦钰程	0000009	password9	29	男	喜欢跑步	common
> 10	10		刘威君	0000010	password10	23	女	喜欢画画	VIP
> 11	11		宋志辉	0000011	password11	31	男	喜欢摄影	common
> 12	12		韩福顺	0000012	password12	28	女	喜欢写作	VIP
> 13	13		董文浩	0000013	password13	26	男	喜欢宠物	common
> 14	14		王彬之	0000014	password14	24	女	喜欢舞蹈	VIP
> 15	15		李超凯	0000015	password15	29	男	喜欢钓鱼	common
> 16	16		陈耀宇	0000016	password16	33	女	喜欢园艺	VIP
> 17	17		朱振萍	0000017	password17	21	男	喜欢露营	common
> 18	18		刘雨南	0000018	password18	30	女	喜欢烘焙	VIP
> 19	19		杨威	0000019	password19	27	男	喜欢阅读	common
> 20	20		陈瑞康	0000020	password20	27	男	喜欢编程	common
> 21	21		李煜科	0000021	password21	27	男	喜欢吃饭	common
> 22	22		杨宇坤	0000022	password22	25	女	喜欢瑜伽	VIP
> 23	23		李美珊	0000023	11222344	24	female	喜欢读书	VIP



	id	name	account	password	age	gender	bio	role
	bigint	varchar(255)	varchar(255)	varchar(255)	int	varchar(10)	text	enum: common
> 2	2	李美丽	0000002	password2	30	女	我爱读书	VIP
> 3	3	鹏哥	0000003	password3	22	男	我爱读书	common
> 4	4	李美丽	0000004	password4	28	女	我爱旅行	VIP
> 5	5	李美丽	0000005	password5	26	男	我爱音乐	common
> 6	6	李美丽	0000006	password6	24	女	我爱旅行	VIP
> 7	7	李美丽	0000007	password7	27	男	我爱旅行	common
> 8	8	李美丽	0000008	password8	32	女	我爱旅行	VIP
> 9	9	李美丽	0000009	password9	29	男	我爱旅行	common
> 10	10	李美丽	0000010	password10	23	女	我爱旅行	VIP
> 11	11	李美丽	0000011	password11	31	男	我爱旅行	common
> 12	12	李美丽	0000012	password12	28	女	我爱旅行	VIP
> 13	13	李美丽	0000013	password13	26	男	我爱旅行	common
> 14	14	李美丽	0000014	password14	24	女	我爱旅行	VIP
> 15	15	李美丽	0000015	password15	29	男	我爱旅行	common
> 16	16	李美丽	0000016	password16	33	女	我爱旅行	VIP
> 17	17	李美丽	0000017	password17	21	男	我爱旅行	common
> 18	18	李美丽	0000018	password18	30	女	我爱旅行	VIP
> 19	19	李美丽	0000019	password19	27	男	我爱旅行	common
> 20	20	李美丽	0000020	password20	27	男	我爱旅行	common
> 21	21	李美丽	0000021	password21	27	男	我爱旅行	common
> 22	22	李美丽	0000022	password22	25	女	我爱旅行	VIP
> 23	23	李美丽	0000023	password23	24	female	我爱旅行	VIP
> 24	24	李美丽	0000024	password24	15	男	我爱旅行	common

注册插入数据（“李美丽”，“0000028”，“11223344”，24，“female”，“我爱读书”，“VIP”）
管理员加入数据（“鹏哥”，“0000037”，“ilovemysql”，15，“男”，“好吃”，“common”）
符合要求，注册成功/添加成功；查看 users 表，成功插入。

说明：违背触发器要求，不能够执行插入操作，系统报错。



在注册与管理员添加用户时，若账号为 Admin 或账号已经存在，均无法执行插入操作。

备注

无

6. 存储过程控制下的更新操作（18分）

说明	<p>（1分）简要说明该操作所要完成的功能；</p> <p>（1分）简要说明该存储过程所要完成的功能；</p> <p>（2分）说明该操作涉及操作的表（必须包含两张或两张以上的关系表，以“表名形式”描述）</p> <p>（1分）表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>（2分）该操作会修改字段（以“表名. 字段名”的形式给出），以及修改规则，如新数值的计算方法、在何种条件下予以修改等；</p> <p>（6分）实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>（5分）如何执行该操作，按所述方法能够正常演示程序则给分。</p>	
功能描述 （1分）	<p>VIP 用户较为尊贵，对账号有特殊要求（比如喜欢靓号等），故管理员不能随便更改 VIP 用户的账号（当然可以先将 VIP 用户改成 common 用户再进行修改）。</p> <p>功能主要有两点：</p> <p>1、修改用户账号时，对相关评论中用户的账号也进行修改；</p> <p>2、管理员可以正常修改普通用户账号，但是直接修改 VIP 用户账号时抛出错误。</p>	
存储过程功能描述 （1分）	创建存储过程，传入参数	
涉及的关系表 （2分）	users、plant_comment、zombie_comment、ammo_comment、equipment_comment	
表连接涉及字段 （1分）		
更改字段 （2分）	字段	规则

更新代码 （3分）	（截屏）	

创建
存储
过程
源码
（3
分）

```
CREATE PROCEDURE updateUserAndComments(  
    IN p_id BIGINT,  
    IN p_name VARCHAR(255),  
    IN p_account VARCHAR(255),  
    IN p_password VARCHAR(255),  
    IN p_age INT,  
    IN p_gender VARCHAR(10),  
    IN p_bio TEXT,  
    IN p_role ENUM('common', 'VIP')  
)  
  
BEGIN  
    DECLARE user_role ENUM('common', 'VIP');  
    DECLARE old_account VARCHAR(255);  
  
    DECLARE EXIT HANDLER FOR SQLEXCEPTION  
    BEGIN  
        -- 回滚事务  
        ROLLBACK;  
        RESIGNAL;  
    END;  
  
    -- 获取旧账号对应的用户角色和旧账号  
    SELECT role, account INTO user_role, old_account FROM users WHERE id = p_id;  
  
    -- 如果用户是VIP且账号被修改, 则引发错误  
    IF user_role = 'VIP' AND p_account != old_account THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot modify VIP user account';  
    END IF;  
  
    -- 开始事务  
    START TRANSACTION;  
  
    -- 更新用户表  
    UPDATE users  
    SET name = p_name, account = p_account, password = p_password, age = p_age, gender = p_gender, bio = p_bio, role = p_role  
    WHERE id = p_id;  
  
    -- 更新评论表中的 user_account  
    UPDATE plant_comment  
    SET user_account = p_account  
    WHERE user_account = old_account;  
  
    UPDATE zombie_comment  
    SET user_account = p_account  
    WHERE user_account = old_account;  
  
    UPDATE ammo_comment  
    SET user_account = p_account  
    WHERE user_account = old_account;  
  
    UPDATE equipment_comment  
    SET user_account = p_account  
    WHERE user_account = old_account;  
  
    -- 提交事务  
    COMMIT;  
END //
```

存储
过程
执行
源码
（1
分）

（截屏）

程序
演示
（2
分）

说明：不违背存储过程，能够执行更新操作

这是原来的用户与评论：看到 id 为 4 的 vip 用户与 id 为 5 的用户都进行了评论

管理用户

添加 更新 删除

按名字查询: 查询

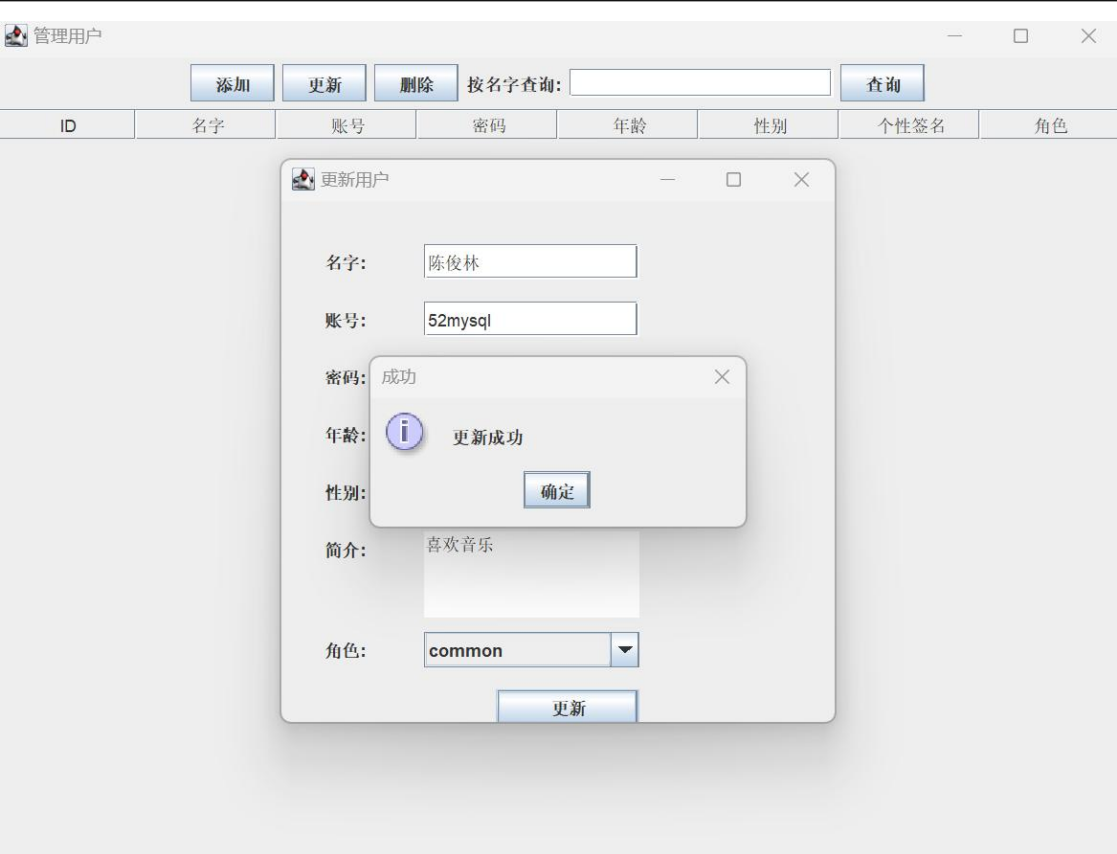
ID	名字	账号	密码	年龄	性别	个性签名	角色
2	李子俊	0000002	password2	30	女	喜欢读书	VIP
3	张耀飞	0000003	password3	22	男	喜欢游泳	common
4	张耀耀	0000004	password4	28	女	喜欢旅行	VIP
5	唐俊林	0000005	password5	26	男	喜欢音乐	common
6	关忠亮	0000006	password6	24	女	喜欢看电影	VIP
7	卢东飞	0000007	password7	27	男	喜欢健身	common
8	左洋宇	0000008	password8	32	女	喜欢烹饪	VIP
9	熊伟程	0000009	password9	29	男	喜欢跑步	common
10	冯晓君	0000010	password10	23	女	喜欢画画	VIP
11	宋永静	0000011	password11	31	男	喜欢摄影	common
12	陈雨辰	0000012	password12	28	女	喜欢写作	VIP
13	梁文浩	0000013	password13	26	男	喜欢电影	common
14	王松之	0000014	password14	24	女	喜欢舞蹈	VIP
15	李嘉凯	0000015	password15	29	男	喜欢钓鱼	common
16	范见宇	0000016	password16	33	女	喜欢园艺	VIP
17	朱福祥	0000017	password17	21	男	喜欢音乐	common
18	刘晋南	0000018	password18	30	女	喜欢烘焙	VIP
19	杨明	0000019	password19	27	男	喜欢摄影	common
20	陈淑康	0000020	password20	27	男	喜欢编程	common
21	李俊科	0000021	password21	27	男	喜欢吃饭	common
22	梅宇坤	0000022	password22	25	女	喜欢瑜伽	VIP
23	李永福	0000023	11223344	24	female	喜欢读书	VIP
24	曹岩	0000037	lcwmytqj	15	男	好吃	common
25	李子俊	0000001	password2	30	女	喜欢读书	VIP

植物评论管理

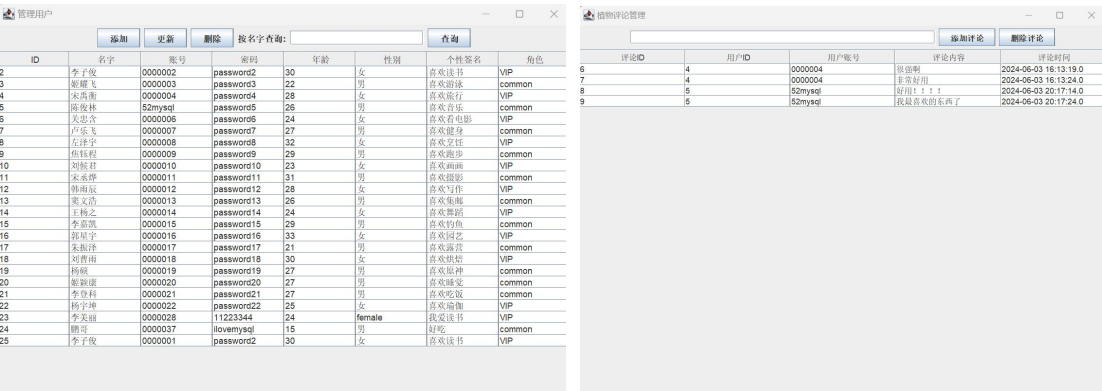
添加评论 删除评论

评论ID	用户ID	用户账号	评论内容	评论时间
6	4	0000004	很棒啊	2024-06-03 16:13:19.0
7	4	0000004	非常好用	2024-06-03 16:13:24.0
8	5	0000005	好自！+！+！	2024-06-03 20:17:14.0
9	5	0000005	我最喜欢的东西了	2024-06-03 20:17:24.0

修改 id 为 5 的用户的账号为：52mysql

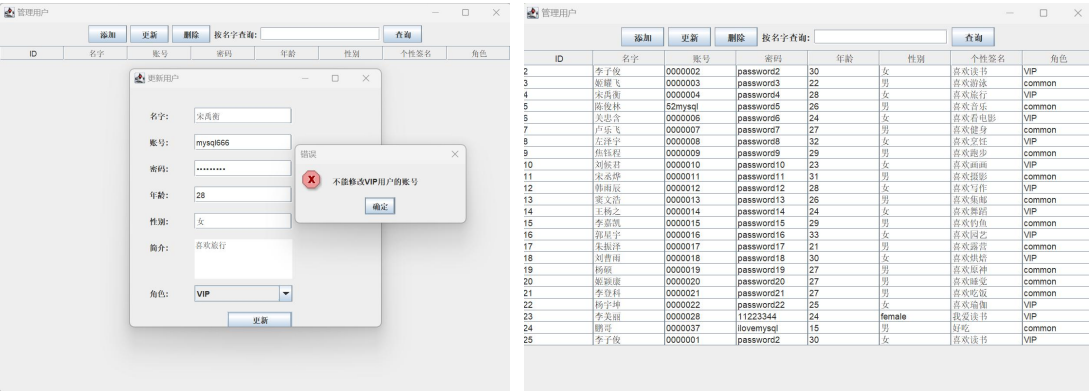


查看现在的用户与评论：



我们看到该用户的账号与对应评论中的账号都修改成功。

说明：违背存储过程，系统报错：



程序
演示
(2
分)

	我们看到修改 VIP 用户的账号时，系统报错，无法对该用户的账号进行修改！
备注	无。

7. 含有视图的查询操作（15 分）

说明	（1 分）简要说明该操作所要完成的功能； （1 分）简要说明建立的该视图的功能； （2 分）简要说明该操作涉及的关系数据表（以“表名”的形式给出） （1 分）简要说明表连接涉及的字段（以“表 1. 属性=表 2. 属性”） （6 分）实现该操作的关键代码（高级语言、SQL），截图即可； （4 分）如何执行该操作，按所述方法能够正常演示程序则给分。
操作功能描述(1 分)	
视图功能描述(1 分)	
涉及的关系表(2 分)	
表连接字段（1 分）	
创建视图代码(3 分)	（截屏）
查询代码（3 分）	（截屏）
程序演示（4 分）	
备注	