数据库工程作业

要求：

1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
3. 报告中所列举的四种操作，每种操作举一个例子即可。
4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

工程作业报告

1. 项目信息（10 分）

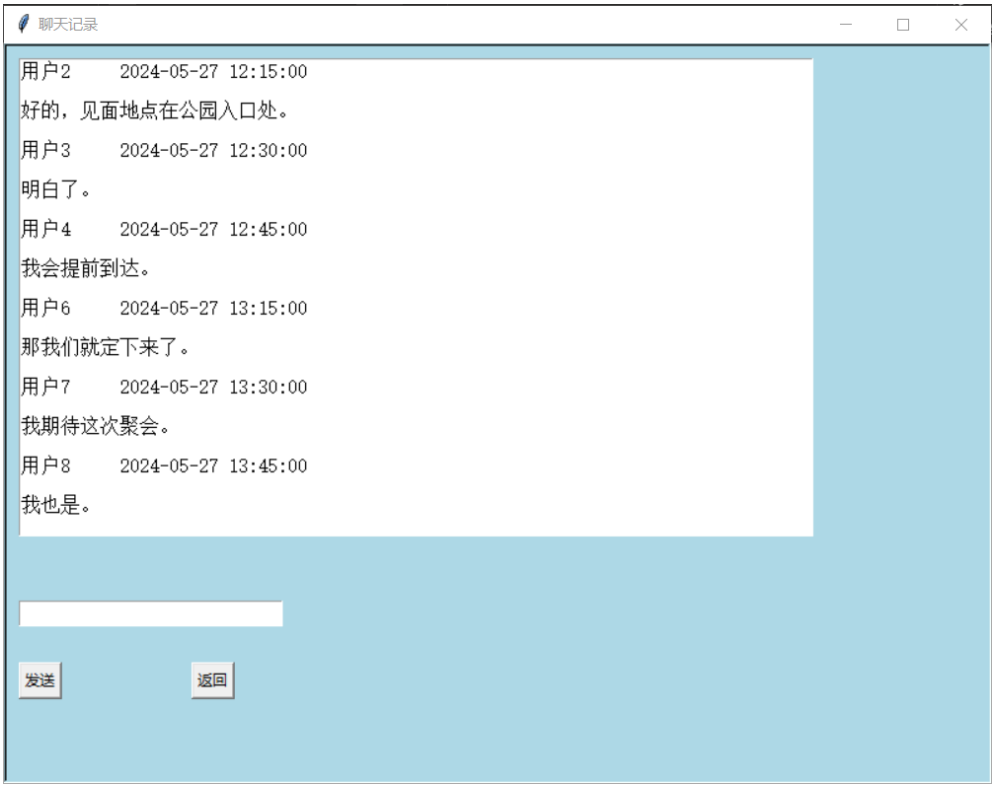
学号	2210351	姓名	赵一凡	专业	计算机科学与技术
项目名称	IOCQ（简易聊天系统）				
必备环境	Python+mysql+tkinter				
系统主要功能简介（4 分）	一个简易的聊天系统，其功能包括： 1.用户可以注册账号，设置密码，昵称，性别，通过管理员将其入群后，就可以自由发送消息，被管理员移出群聊的用户不能再进入群聊。 2.用户可以修改其账号信息。 3.管理员可以查看群聊中所有成员的基本信息，可以将选中成员移出群聊，可以向群中添加用户。 4.管理员可以查看群聊中所有聊天记录，可以删除指定记录。				

系统主要页面截图（6 分）	 


管理员操作界面
群成员信息:
账号: 234567
昵称: 用户2
入群时间: 2024-05-25 10:15
性别: 女
操作:
添加群成员
移除群成员
账号 昵称 入群时间 性别 管理员
111111 用户11 2024-06-02 23:1 男 0
123456 用户1 2024-06-03 20:2 男 0
234567 用户2 2024-05-25 10:1 女 0
345678 用户3 2024-05-26 10:3 男 0
456789 用户4 2024-05-26 10:4 女 0
567890 用户5 2024-06-02 23:1 男 0
678901 用户6 2024-05-26 11:1 女 0
789012 用户7 2024-05-26 11:3 男 0
890123 用户8 2024-05-26 11:4 女 0
901234 用户9 2024-05-26 12:0 男 0
1122112 管理员1 2024-05-24 10:1 女 1
2211221 管理员2 2024-05-24 11:0 男 1


聊天记录
编号: REC012 用户2 234567 2024-05-27 12:15:00
好的，见面地点在公园入口处。
编号: REC013 用户3 345678 2024-05-27 12:30:00
明白了。
编号: REC014 用户4 456789 2024-05-27 12:45:00
我会提前到达。
编号: REC016 用户6 678901 2024-05-27 13:15:00
那我们就定下来了。
编号: REC017 用户7 789012 2024-05-27 13:30:00
我期待这次聚会。
编号: REC018 用户8 890123 2024-05-27 13:45:00
我也是。
删除记录
返回


新用户注册
新用户注册
账号：
12343513
昵称：
小明
密码：
*****
性别：
男
注册
返回首页

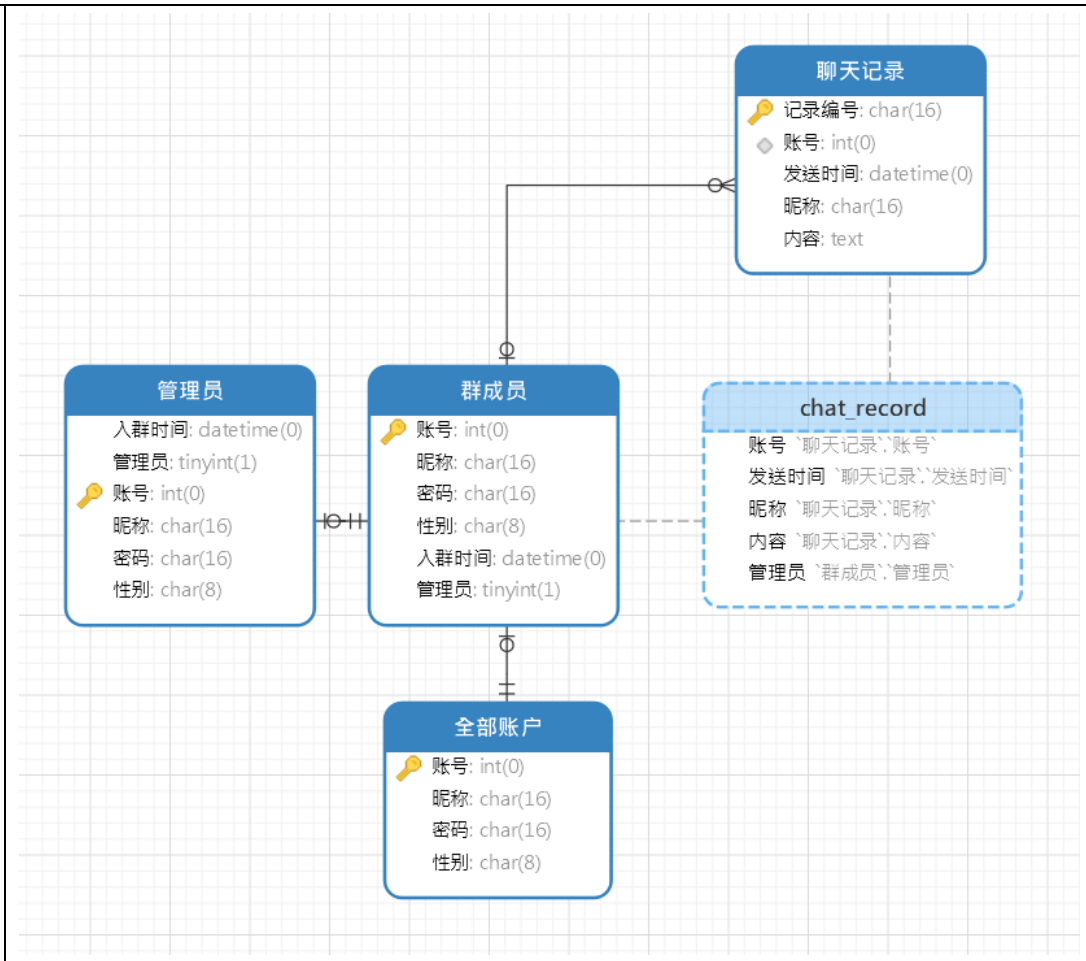

聊天记录
用户2 2024-05-27 12:15:00
好的，见面地点在公园入口处。
用户3 2024-05-27 12:30:00
明白了。
用户4 2024-05-27 12:45:00
我会提前到达。
用户6 2024-05-27 13:15:00
那我们就定下来了。
用户7 2024-05-27 13:30:00
我期待这次聚会。
用户8 2024-05-27 13:45:00
我也是。
发送
返回

2. 系统配置（10 分）

<table>
<tr><td colspan="2">说明</td><td colspan="4">（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。</td></tr>
<tr><td rowspan="4">配置步骤 2 分</td><td rowspan="2">DBMS</td><td colspan="4">1.mysql</td></tr>
<tr><td colspan="4">2.……</td></tr>
<tr><td rowspan="2">高级语言</td><td colspan="4">1.python</td></tr>
<tr><td colspan="4">2.……</td></tr>
<tr><td colspan="2" rowspan="6">连接串分析（6 分）</td><td>序号</td><td>名称</td><td>功能说明</td><td>取值</td></tr>
<tr><td>1</td><td>part</td><td>接受端口名</td><td>3306</td></tr>
<tr><td>2</td><td>password</td><td>接受密码</td><td>zhyf040216</td></tr>
<tr><td>3</td><td>host</td><td>接收主机名</td><td>localhost</td></tr>
<tr><td>4</td><td>user</td><td>接受用户名</td><td>root</td></tr>
<tr><td>5</td><td>db</td><td>接受数据库名</td><td>群聊系统</td></tr>
<tr><td colspan="2">连接串代码（截屏）（2 分）</td><td colspan="4"> 
</td></tr>
<tr><td colspan="2">备注</td><td colspan="4">无</td></tr>
</table>

3. 数据库设计（14 分）

说明	（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； （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。				
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	全部账户	账号	无	
	2	群成员	账号	全部账户.账号	全部账户(账号,昵称,密码,性别)
	3	聊天记录	记录编码	群成员.账号	群成员(账号,昵称,密码,性别,入群时间,管理员)
	4	管理员	账号	群成员.账号	群成员(账号,昵称,密码,性别,入群时间,管理员)

关系图（4）	
备注	

4. **含有事务应用的删除操作（13 分）**

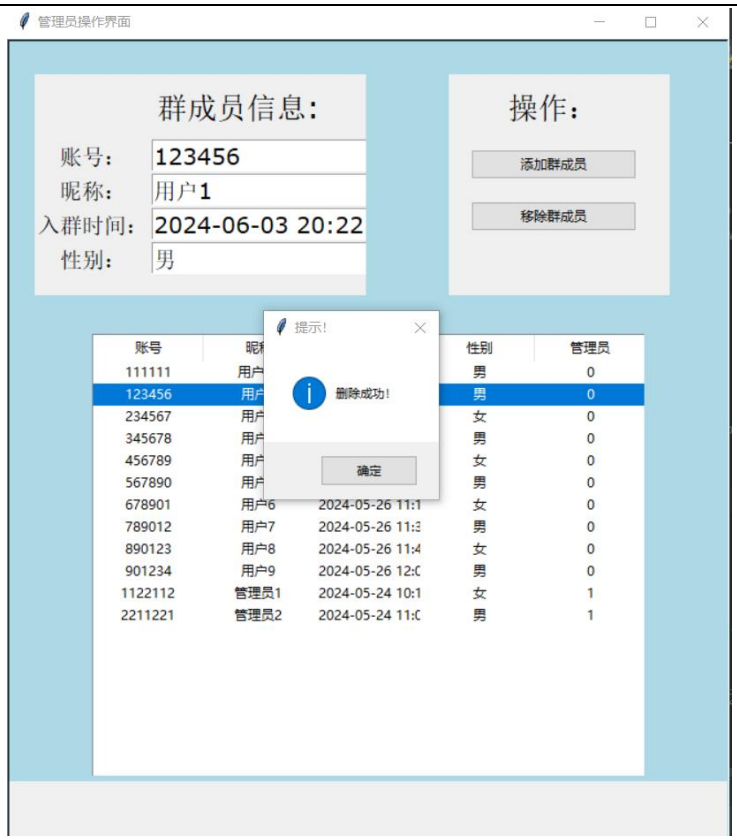
说明	（1 分）简要说明该操作所要完成的功能； （2 分）该操作会涉及的表（必须含有两张或两张以上的关系表，同时以“表名”的形式给出） （1 分）表连接涉及字段描述（描述方式为“表 1.属性=表 2.属性”） （1 分）删除条件涉及的字段描述(以“表名.属性=？”形式给出) （4 分）实现该操作的关键代码（高级语言、SQL），截图即可；（其中如果删除语句中不包含任何形式的事务应用将扣除 3 分） （4 分）如何执行该操作，按所述方法能够正常演示程序则给分。
功能描述（1 分）	管理员从群中移除选中群成员，顺带着把其聊天记录全部移除
涉及的表	群成员，聊天记录

<table>
<tr><td>（2分）</td><td colspan="2"></td></tr>
<tr><td>表连接涉及字段（1分）</td><td colspan="2">群成员.账号=聊天记录.账号</td></tr>
<tr><td rowspan="2">删除条件字段描述（1分）</td><td>字段</td><td>规则</td></tr>
<tr><td>群成员.账号=</td><td>点击移除群成员后，系统会删除该账号的聊天记录和群成员信息</td></tr>
<tr><td>代码（4分）</td><td colspan="2">

```
def del_row(self):
    res = messagebox.askyesnocancel( title: '警告！ ',  message: '是否删除所选数据？ ')
    if res == True and self.row_info[4] == '0':
        # 检查 self.row_info[0] 是否确实存在于 self.id 列表中
        print(self.id)
        print(self.row_info[0])
        intid = int(self.row_info[0])#id列表中是int型
        if intid in self.id:
            # 打开数据库连接
            db = pymysql.connect(host='localhost', port=3306, db='群聊系统', user='root', password='zhyf040216')
            cursor = db.cursor()  # 使用cursor()方法获取操作游标
            sql = "set foreign_key_checks = 0"

            sql_delete = "DELETE 群成员,聊天记录 FROM 群成员,聊天记录 WHERE 群成员.账号=聊天记录.账号 AND 群成员.账号 = %s"

            sql_delete0 = "delete from 群成员 where 群成员.账号 = %s"
            sql1 = "set foreign_key_checks = 1"
```

</td></tr>
</table>

```
    # 执行SQL语句
    try:
        cursor.execute(sql)
        cursor.execute(sql_delete, args: (intid,))
        cursor.execute(sql_delete0, args: (intid,))
        cursor.execute(sql1)
        db.commit()  # 提交到数据库执行
        messagebox.showinfo( title: '提示！', message: '删除成功！')
    except pymysql.MySQLError as e:
        db.rollback()  # 发生错误时回滚
        messagebox.showinfo( title: '警告！', message: f'删除失败：{e}')
     # 关闭数据库连接

    # 删除本地列表中的记录
    id_index = self.id.index(intid)
    del self.id[id_index]
    del self.name[id_index]
    del self.time[id_index]
    del self.gender[id_index]
    del self.admin[id_index]

    # 从树形视图中删除对应的行
    self.tree.delete(self.tree.selection()[0])
else:
    messagebox.showinfo( title: '警告！', message: '要删除的记录不存在！')
```

程序演示（4分）	 管理员操作界面 群成员信息： 账号：123456 昵称：用户1 入群时间：2024-06-03 20:22 性别：男 操作： 添加群成员 移除群成员 提示! 删除成功! 确定 账号 昵称 入群时间 性别 管理员 111111 用户 男 0 123456 用户 男 0 234567 用户 女 0 345678 用户 男 0 456789 用户 女 0 567890 用户 男 0 678901 用户6 2024-05-26 11:1 女 0 789012 用户7 2024-05-26 11:3 男 0 890123 用户8 2024-05-26 11:4 女 0 901234 用户9 2024-05-26 12:0 男 0 1122112 管理员1 2024-05-24 10:1 女 1 2211221 管理员2 2024-05-24 11:0 男 1  管理员操作界面 群成员信息： 账号：111111 昵称：用户11 入群时间：2024-06-02 23:13 性别：男 操作： 添加群成员 移除群成员 账号 昵称 入群时间 性别 管理员 111111 用户11 2024-06-02 23:1 男 0 234567 用户2 2024-05-25 10:1 女 0 345678 用户3 2024-05-26 10:3 男 0 456789 用户4 2024-05-26 10:4 女 0 567890 用户5 2024-06-02 23:1 男 0 678901 用户6 2024-05-26 11:1 女 0 789012 用户7 2024-05-26 11:3 男 0 890123 用户8 2024-05-26 11:4 女 0 901234 用户9 2024-05-26 12:0 男 0 1122112 管理员1 2024-05-24 10:1 女 1 2211221 管理员2 2024-05-24 11:0 男 1
备注	

5. **触发器控制下的添加操作（20 分）**

<table>
<tr><td>说明</td><td colspan="2">（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。</td></tr>
<tr><td>功能描述（1 分）</td><td colspan="2">管理员输入账号，向群成员中添加该用户，添加存在限制条件，其中部分使用触发器实现。</td></tr>
<tr><td>触发器描述（2 分）</td><td colspan="2">在添加群成员时，使用一个 before 触发器(before_insert_check)，在插入之前，检查要添加的群成员是否已经在群成员中，如果存在就不会进行插入操作，并返回一个错误“不能重复添加该用户”。</td></tr>
<tr><td>涉及的表（1 分）</td><td colspan="2">群成员</td></tr>
<tr><td rowspan="2">输入数据（2 分）</td><td>字段</td><td>规则</td></tr>
<tr><td>账号</td><td>不能为群成员中已经存在的账号</td></tr>
</table>

<table>
<tr>
<td>插入操作源码（3分）</td>
<td>

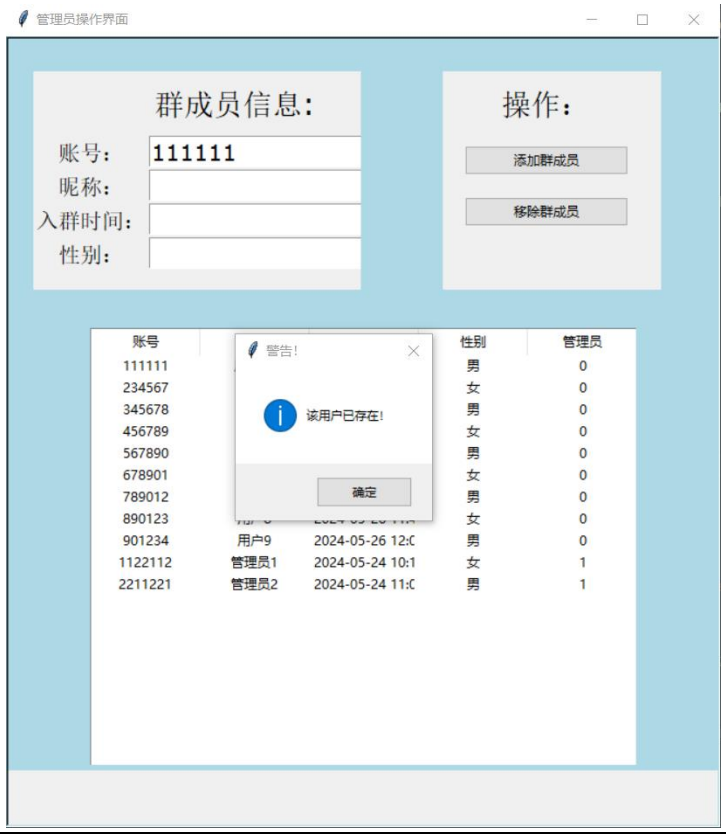
```python
#触发器控制下的添加操作
1 个用法
def new_row(self):
    ...
    current_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(current_date)

    if self.var_id.get() == '':
        messagebox.showinfo( title: '警告！', message: '请填写学生数据')
        return

    if int(self.var_id.get()) in self.id:...

    db = pymysql.connect(host='localhost', port=3306, db='群聊系统', user='root', password='zhyf040216')
    cursor = db.cursor()  # 使用cursor()方法获取操作游标
    sql0 = "SELECT 账号, 昵称, 密码, 性别 FROM 全部账户 WHERE 账号 = %s"
    cursor.execute(sql0, args: (self.var_id.get(),))
    results = cursor.fetchall()

    if len(results) == 0:
        messagebox.showinfo( title: '警告！', message: '没有找到匹配的用户！')
        db.close()
        return
```

```python
sql = "INSERT INTO 群成员(账号, 昵称, 密码, 性别,入群时间,管理员) \
       VALUES ('%s', '%s', '%s', '%s','%s', '%s')" % \
      (self.var_id.get(), results[0][1], results[0][2], results[0][3], current_date, '0')  # SQL 插入语句

try:
    cursor.execute(sql)  # 执行sql语句
    db.commit()  # 提交到数据库执行
except pymysql.MySQLError as e:
    db.rollback()  # 发生错误时回滚
    messagebox.showinfo( title: '警告！', message: f'触发器：{e}')
    db.close()
    return

db.close()  # 关闭数据库连接
```

</td>
</tr>
<tr>
<td>触发器源码（3分）</td>
<td>

```
mysql> show create trigger before_insert_check;
+---------------------+---------------------------------------------------------------------------------------------------------------------------+-------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------+---------------------+----------------------+--------------------+----
--------------------+
| Trigger             | sql_mode                                                                                                                  | SQL Original S
tatement
                                                                                            | character_set_client | collation_connection | Database Collation | Crea
ted                 |
+---------------------+---------------------------------------------------------------------------------------------------------------------------+-------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------+---------------------+----------------------+--------------------+----
--------------------+
| before_insert_check | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | CREATE DEFINER
=`root`@`localhost` TRIGGER `before_insert_check` BEFORE INSERT ON `群成员` FOR EACH ROW BEGIN
    IF EXISTS (SELECT 1 FROM 群成员 WHERE 账号 = NEW.账号) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '不能重复添加该用户';
    END IF;
END | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci | 2024-06-02 19:34:52.17 |
+---------------------+---------------------------------------------------------------------------------------------------------------------------+-------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------+---------------------+----------------------+--------------------+----
--------------------+
1 row in set (0.03 sec)
```

</td>
</tr>
<tr>
<td>程序演示（4分）</td>
<td>说明：不违背触发器能够执行插入操作。</td>
</tr>
</table>

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>程序演示（4分）</td><td>说明：违背触发器要求，不能够执行插入操作，系统报错。 
</td></tr>
<tr><td>备注</td><td></td></tr>
</table>

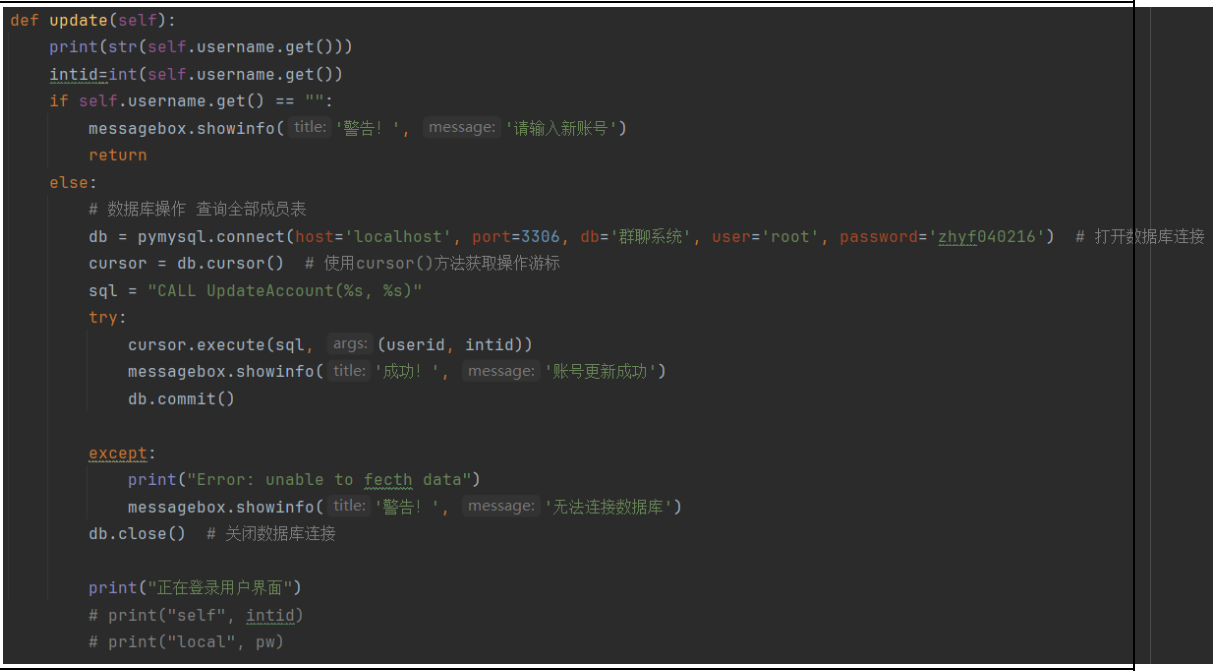
6. **存储过程控制下的更新操作（18 分）**

<table>
<tr><td>说明</td><td>（1 分）简要说明该操作所要完成的功能；
（1 分）简要说明该存储过程所要完成的功能；
（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）
（1 分）表连接涉及字段描述（描述方式为“表 1.属性=表 2.属性”）
（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；
（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；
（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。</td></tr>
<tr><td>功能描述（1 分）</td><td>用户可以修改自己的账号达到个性化目的，如果用户在群中，那么修改时会将全部账户,群成员,聊天记录中的账号信息一起更新。</td></tr>
<tr><td>存储过程功能描述（1 分）</td><td>存储过程有两个参数，一个是旧账号，另一个是新账号，过程实现修改账号操作，如果新账号已经存在全部账户中，会抛出错误“该账号已存在”，如果符合条件，则会将全部账户,群成员,聊天记录中的账号一起更新。</td></tr>
<tr><td>涉及的关系表（2 分）</td><td>全部账户,群成员,聊天记录</td></tr>
<tr><td>表连接涉及字段（1）</td><td>全部账户.账号 = 群成员.账号,群成员.账号=聊天记录.账号</td></tr>
<tr><td>更改字段（2 分）</td><td><table><tr><td>字段</td><td>规则</td></tr><tr><td>聊天记录.账号</td><td>如果新账号不存在于全部账户中，则更新聊天记录中的账号，否则报错</td></tr><tr><td>全部账户.账号</td><td>如果新账号不存在于全部账户中，则更新全部账户中的账号，否则报错</td></tr><tr><td>群成员.账号</td><td>如果新账号不存在于全部账户中，则更新群成员中的账号，否则报错</td></tr></table></td></tr>
<tr><td>更新代码（3 分）</td><td><pre>
UPDATE 全部账户
SET 账号 = new_account_id
WHERE 账号 = old_account_id;

UPDATE 聊天记录
SET 账号 = new_account_id
WHERE 账号 = old_account_id;

UPDATE 群成员
SET 账号 = new_account_id
WHERE 账号 = old_account_id;
</pre></td></tr>
</table>

创建存储过程源码（3分）		UpdateAccount	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION	CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateAccount`(
				IN old_account_id int,
				IN new_account_id int
				)
				BEGIN
				-- 声明变量来存储查询结果
				DECLARE exists_count INT;
				
				-- 检查新账号是否已存在
				SELECT COUNT(*) INTO exists_count
				FROM 全部账户
				WHERE 账号 = new_account_id;
				
				-- 如果新账号已存在，则抛出错误
				IF exists_count > 0 THEN
				SIGNAL SQLSTATE '45000'
				SET MESSAGE_TEXT = '该账号已存在';
				END IF;
				
				-- 如果新账号不存在，则更新账号
				set foreign_key_checks = 0;
				UPDATE 全部账户
				SET 账号 = new_account_id
				WHERE 账号 = old_account_id;
				
				UPDATE 聊天记录
				SET 账号 = new_account_id
				WHERE 账号 = old_account_id;
				
				UPDATE 群成员
				SET 账号 = new_account_id
				WHERE 账号 = old_account_id;
				set foreign_key_checks = 1;
				END
					utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	

存储过程执行源码（1分）	
程序演示（2分）	说明：不违背存储过程，能够执行更新操作 登录账号 1234567

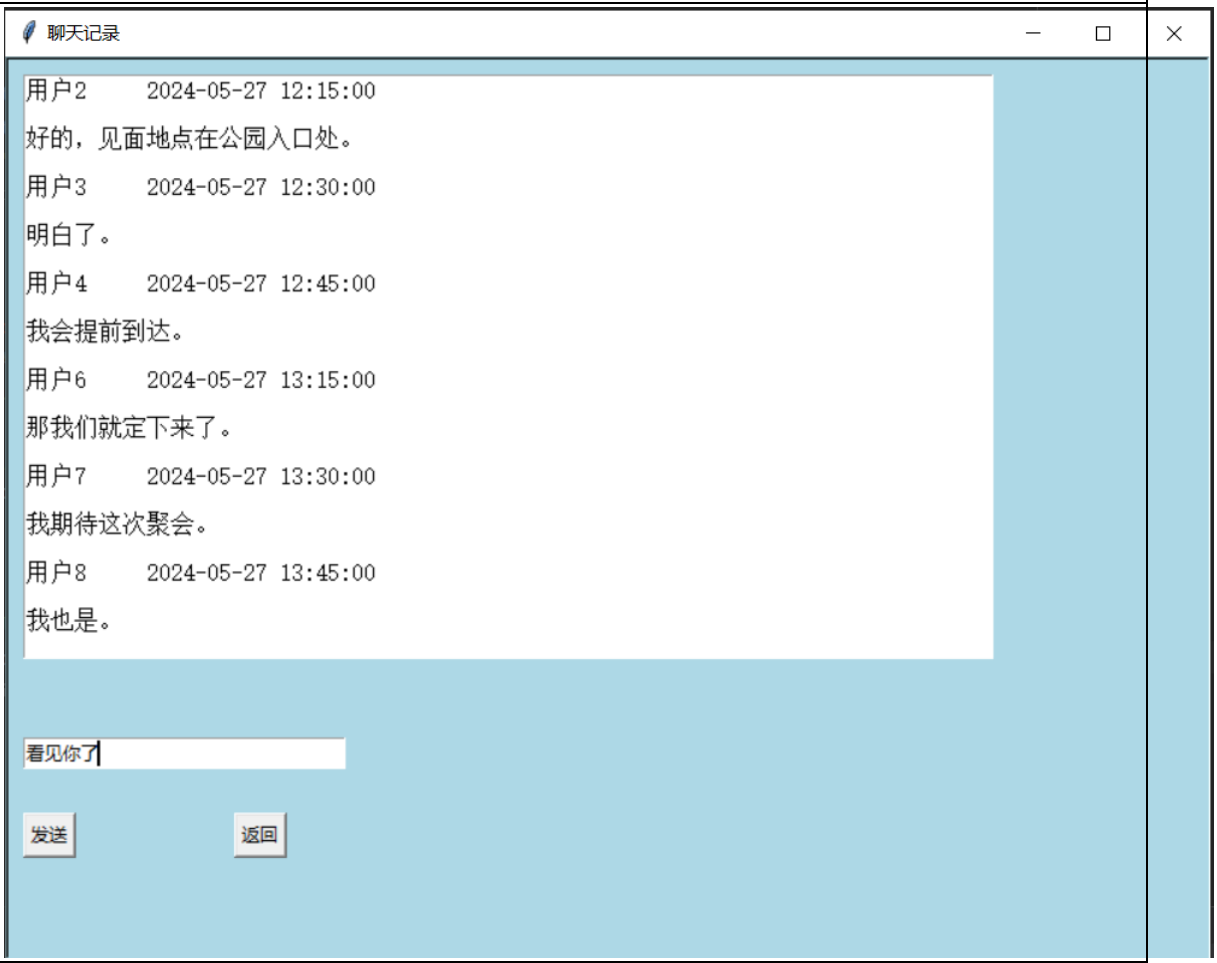
<table>
<tr><td></td><td> 
</td></tr>
<tr><td>程序演示（2分）</td><td>说明：违背存储过程，系统报错；
登录账号 1234567
 
</td></tr>
<tr><td>备注</td><td></td></tr>
</table>

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）	选择聊天记录中的属性与群成员中的属性构建一个视图，从中查询全部聊天记录并显示出来。
视图功能描述（1 分）	将群成员与聊天记录两张表通过账号连接成一张虚拟表，可以从中查询部分属性。属性为昵称，发送时间，管理员，内容
涉及的关系表（2 分）	群成员，聊天记录
表连接字段（1 分）	群成员.账号=聊天记录.账号
创建视图代	'CREATE VIEW chat_record AS SELECT 聊天记录.账号 as 账号,聊天记录.发送时间 as 发送时间,聊天记录.昵称 as 昵称,聊天记录.内容 as 内容,群成员.管理员 as 管理员

码（3分）	FROM 聊天记录 INNER JOIN 群成员 ON 聊天记录.账号 = 群成员.账号;"

查询代码（3分）	# 连接数据库
	db = pymysql.connect(host='localhost', port=3306, db='群聊系统', user='root', password='zhyf040216')
	cursor = db.cursor()
	sql_drop = "DROP VIEW IF EXISTS chat_record"
	sql = "CREATE VIEW chat_record AS SELECT 聊天记录.账号 as 账号,聊天记录.发送时间 as 发送时间,聊天记录.昵称 as 昵称,聊
	sql_view = "SELECT 昵称,发送时间,管理员,内容 FROM chat_record"
	
	try:
	# 执行SQL语句
	cursor.execute(sql_drop)
	cursor.execute(sql)
	cursor.execute(sql_view) # 使用参数化查询
	# 获取所有记录列表
	results = cursor.fetchall()

程序演示（4分）	
备注	