**计算机科学与技术学院课程设计成绩单**

**课程名称：数据库系统课程设计**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 |  | 性别 |  | 学号 | |  | | 班级 |  |
| 电话 |  | | 综合成绩 | |  | | 成绩等级 | |  |
| 程序运行情况  （占总成绩20%） | | | |  | | | | | |
| 程序功能完善程度  （占总成绩10%） | | | |  | | | | | |
| 数据库逻辑结构的合理性  （占总成绩10%） | | | |  | | | | | |
| 对问题的答辩情况  （占总成绩40%） | | | |  | | | | | |
| 学生的工作态度与独立工作能力  （占总成绩10%） | | | |  | | | | | |
| 设计报告的规范性  （占总成绩10%） | | | |  | | | | | |

A：90~100分 A-：85~89分 B+：82~84分 B：78~81分 B-：75~77分

C+：72~74分 C：68~71分 C-：64~67分 D：60~63分 F：<60分

武汉科技大学计算机科学与技术学院制表

会议室预约管理系统

一、需求分析

在会议室预约管理系统中，角色分普通用户和管理员两个角色。普通用户拥有账号，密码，姓名，电话，部门，邮箱等信息；管理员拥有账号，邮箱，密码，姓名等信息。用户在登录系统时，先输入账号密码以及身份类别，如果身份验证失败应该有错误反馈。两个角色在进入系统后都可以根据实际修改自己的基本信息。普通用户进入系统后，可以查询会议室信息，发起预约会议室的申请，或者取消预约申请。会议室有名称，地点，位置，状态等信息。管理员进入系统后，可以看到数据统计，也可以一览预约申请的信息，同时可对申请进行审批，即给予申请通过与否。系统的操作有相应的操作反馈，界面友好，交互性强。

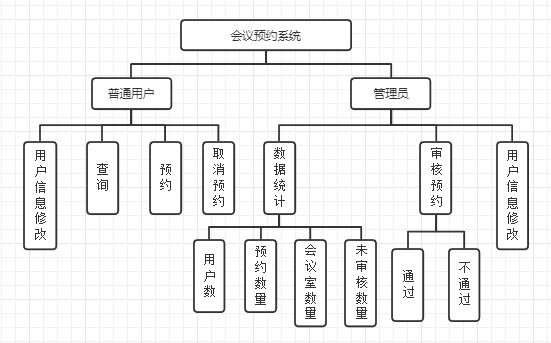


图1 总体功能图

二、概念结构设计

会议室预约系统涉及三个实体，分别为：普通用户、管理员、会议室。

1. 普通用户：id、所在部门、姓名、账号、电话、密码、邮箱、职务；
2. 管理员：id、账号、密码、电话、邮箱、姓名；
3. 会议室：id、会议室名称、地点、位置、状态。

实体间的关系包括：

1、普通用户与会议室之间存在三个关系：普通用户查询会议室为多对多关系，普通用户预约与取消预约会议室为多对多关系；

2、会议室与管理员之间存在两个关系：管理员对会议室进行数据统计为多对多关系，管理员审核预约会议室为多对多关系。

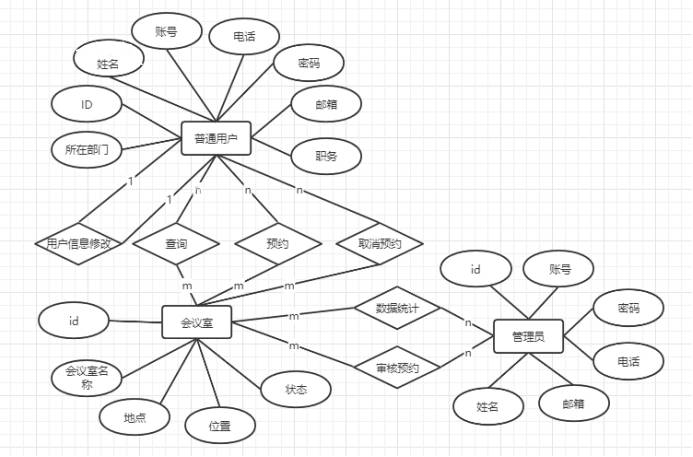


图2 E-R图

关系模式设计：

1. 管理员(id、name、phone、account、password、email)

2、普通用户(id、account、real\_name、apartment、phone、email、password、duty)

3、会议室(id、name、address、seat\_num、status)

4、会议室预约(id、user\_id、room\_id、status)

(注：单线：主键； 波浪线：外键)

1. 逻辑结构设计

下列一共四张表是由E-R图得到具体的逻辑结构表，详细的表结构见管理员表(表1)、用户表(表2)、会议室表(表3)、会议室预约表(表4)。

表1管理员表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | integer | 主键，非空 |
| name | varchar | 姓名 |
| phone | varchar | 电话 |
| account | varchar | 账号ID |
| password | varchar | 密码 |
| email | varchar | 邮箱 |

表2 用户表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | integer | 主键，非空 |
| account | varchar | 账号 |
| real\_name | varchar | 真实姓名 |
| apartment | varchar | 所在部门 |
| phone | varchar | 电话 |
| email | varchar | 邮箱 |
| password | varchar | 密码 |
| duty | varchar | 植物 |

表3 会议室表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | integer | 主键，非空 |
| name | varchar | 会议室名称 |
| address | varchar | 会议室地点 |
| seat\_num | integer | 座位数 |
| status | varchar | 状态 |

表4 会议室预约表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | integer | 非空 |
| user\_id | integer | 用户id，外键 |
| room\_id | integer | 会议室id，外键 |
| status | integer | 审核状态 |

四、开发工具

* + 开发工具：Eclipse2021
  + 数据库：MySQL 8.0+
  + 编程语言：Java
  + 编码格式：UTF-8

1. 具体实现

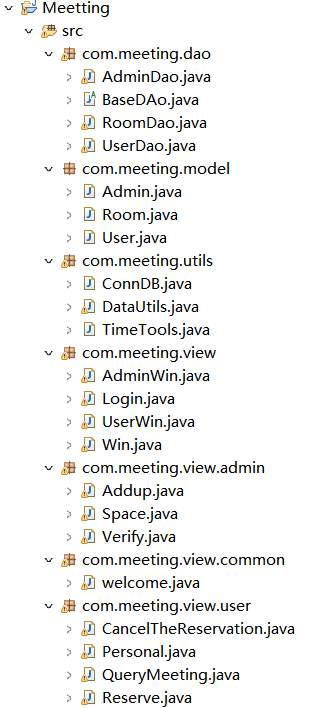


图7 总体结构

核心代码

1.Java连接数据库

**package** com.meeting.utils;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** ConnDB {

**public** **static** Connection getConnection() {

Connection conn = **null**;

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

String url ="jdbc:mysql://localhost:3306/meeting?useSSL=false&serverTimezone=Asia/Shanghai";

String user = "root";

String password = "xulu20011202";

conn = DriverManager.*getConnection*(url,user,password);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** conn;

}

**public** **static** **void** closeDB(ResultSet rs,Statement pst,Connection conn) {

**try** {

**if**(rs!=**null**) {

rs.close();

}

**if**(pst!=**null**) {

pst.close();

}

**if**(conn!=**null**) {

conn.close();

}

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

System.***out***.println(ConnDB.*getConnection*());

}

}

2.用户的查找与修改（用户、会议室、管理员的查找与修改实现方法类似）

**package** com.meeting.dao;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** com.meeting.model.Admin;

**public** **class** AdminDao **extends** BaseDAo{

// 根据用户名查找一个用户

**public** Admin selectOne(String account) {

String sql="select \* from admin where account=?";

**return** **super**.select(sql, **new** Object[] {account});

}

// 通过id查询用户信息

**public** Admin selectOneById(**int** id) {

String sql="select \* from admin where id=?";

**return** **super**.select(sql, **new** Object[] {id});

}

// 修改信息

**public** **int** updateInfo(Admin admin) {

String sql="UPDATE admin a set a.name=?,a.phone=?,a.email=?,a.`password`=? WHERE id=?";

**return** **super**.update(sql, **new** Object[] {

admin.getName(),

admin.getPhone(),

admin.getEmail(),

admin.getPassword(),

admin.getId()

});

}

@Override

**public** Admin rowMapper(ResultSet rs) {

Admin admin=**new** Admin();

**try** {

admin.setId(rs.getInt(1));

admin.setName(rs.getString(2));

admin.setPhone(rs.getString(3));

admin.setAccount(rs.getString(4));

admin.setPassword(rs.getString(5));

admin.setEmail(rs.getString(6));

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** admin;

}

}

3.基本sql封装

**package** com.meeting.dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** com.meeting.utils.ConnDB;

**public** **abstract** **class** BaseDAo {

// 进行增删改操作

**public** **int** update(String sql,Object[] arr) {

Connection conn = ConnDB.*getConnection*();

PreparedStatement pst = **null**;

**try** {

pst = conn.prepareStatement(sql);

**for** (**int** i = 0; i < arr.length; i++) {

pst.setObject(i+1, arr[i]);

}

**int** row = pst.executeUpdate();

**return** row;

} **catch** (SQLException e) {

e.printStackTrace();

}**finally** {

ConnDB.*closeDB*(**null**, pst, conn);

}

**return** 0;

}

**public** <T>T select(String sql,Object[] arr){

Connection conn = ConnDB.*getConnection*();

PreparedStatement pst = **null**;

ResultSet rs = **null**;

T t = **null**;

**try** {

pst= conn.prepareStatement(sql);

**for** (**int** i = 0; i < arr.length; i++) {

pst.setObject(i+1, arr[i]);

}

rs = pst.executeQuery();

**if**(rs.next()) {

t = **this**.rowMapper(rs);

}

} **catch** (SQLException e) {

e.printStackTrace();

}**finally** {

ConnDB.*closeDB*(**null**, pst, conn);

}

**return** t;

}

// 查询结果集合

**public** <T> List<T> selectAll(String sql,Object[] arr){

List<T> list = **new** ArrayList<>();

Connection conn = ConnDB.*getConnection*();

PreparedStatement pst = **null**;

ResultSet rs = **null**;

**try** {

pst= conn.prepareStatement(sql);

**for** (**int** i = 0; i < arr.length; i++) {

pst.setObject(i+1, arr[i]);

}

rs = pst.executeQuery();

T t = **null**;

// 对结果集的收集

**while**(rs.next()) {

// 收集对象的属性

t = **this**.rowMapper(rs);

// 添加到集合

list.add(t);

}

} **catch** (SQLException e) {

e.printStackTrace();

}**finally** {

ConnDB.*closeDB*(**null**, pst, conn);

}

**return** list;

}

// 定义方法 每个查询的结果集获取对应的属性

**public** **abstract** <T> T rowMapper(ResultSet rs) ;

}

运行结果

1、登录验证。首先用户输入账号和密码，并选择所属身份。如果身份验证通过，则可以进入系统，否则不能进入系统，并有错误反馈。

验证失败示例：



图8登录验证失败

验证通过示例：



图9登录验证通过

2、用户个人信息。用户可以在版面查看自己的基本信息，点击修改按钮，文本框切换到可编辑状态，此时可以修改个人基本信息，然后点保存即可更新完毕。



图10用户个人信息界面

3、会议室查询。可以一览全部可用会议室，也可以通过会议室名称搜索相应的会议室信息。



图11会议室查询界面

4、会议预约。输入会议室的id，然后点击“我要预约”按钮，如果操作成功，即可在我的预约记录里面新增一条。

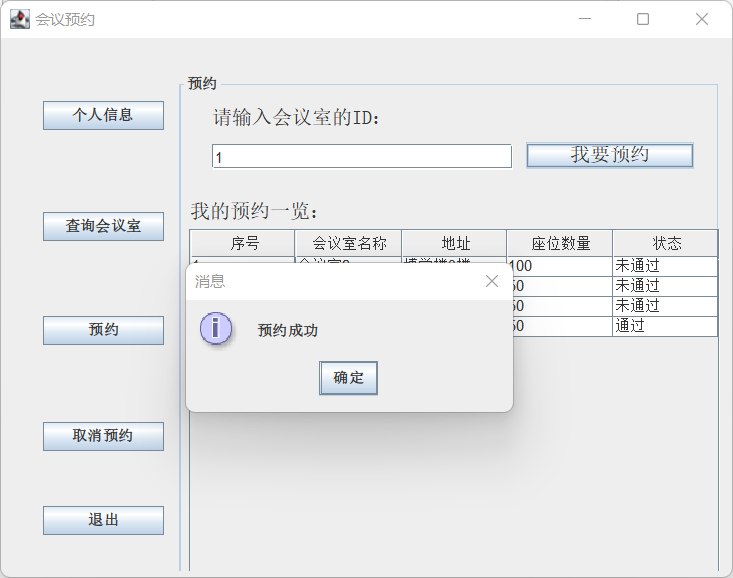


图12会议室预约成功界面

5、取消预约。通过输入已经预约的会议的ID，可取消预约。

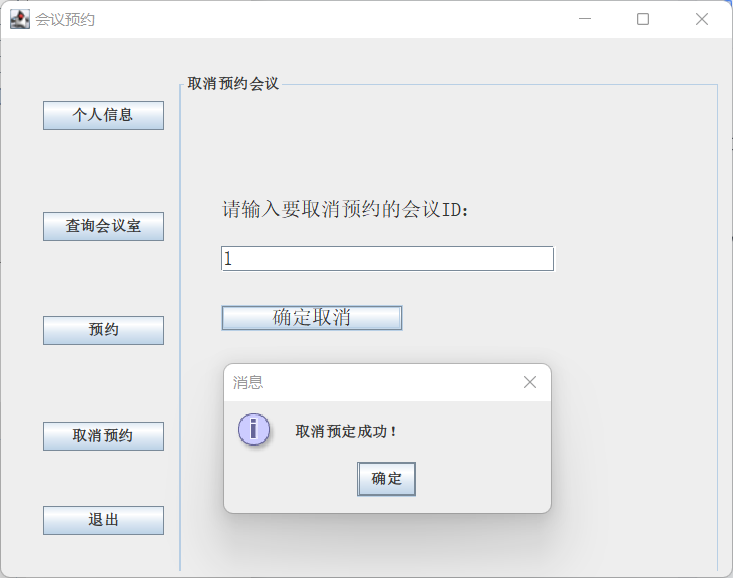


图13会议室取消预约界面

6、管理员数据统计。在此面板，可以看到系统的一些数据统计，如总用户数，可用会议数量，预约数量等。



图14管理员数据统计界面

7、审核预约。在此模块管理员可以看到已审核的预约申请记录，也可以对待审核的申请进行“通过”与“未通过”标记。



图15管理员审核界面

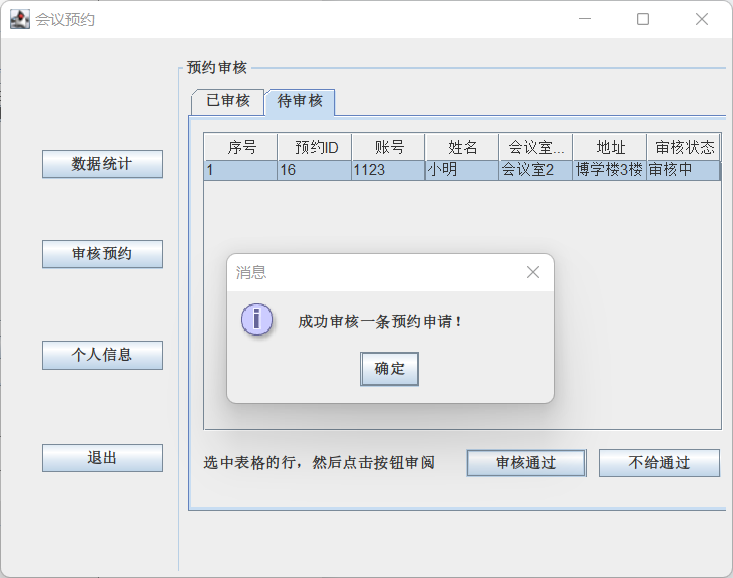


图16管理员审核通过界面

8、管理员个人信息更新与查看。

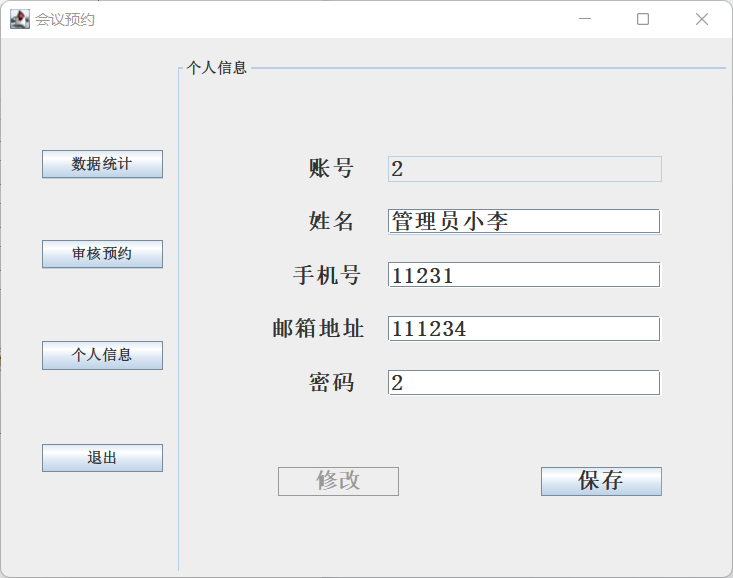


图17管理员个人信息更新与查看界面

六、总结

通过这次的课程设计，加深了自己对JDBC的理解，了解了更多Java编程的知识及应用，也熟悉了Java编程与MySQL数据库交互的过程，同时我也清楚地意识到自己对Java及数据库编程上的知识还是知之甚少，这其中需要的一些知识我还没有接触过，上网查资料的过程中更是发现我们之前学到的东西仅仅只是皮毛，还有我也发现了有很多学过的东西我并未理解到位，不能真正做到灵活运用于实际，不能很好的用以解决问题。后续我将继续对自己已接触到的知识查漏补缺，继续做一些相应的练习来巩固之前所学的知识，也会努力抽时间去了解自己未接触过的知识，让开发应用到实际的生活中，解决切实的生活问题。

在本次课程设计完成的过程中我遇到了不少困难，比如一些自己不太熟悉的功能操作部分会去网络上找寻相似功能的内容去借鉴他们的方法结构来帮助自己以达到效果，代码的完成过程也让我注意到了之前一直忽略的自己在编程过程中的一些不良习惯从而及时加以改正。综上，我意识到自己急需不断大量的实践，通过不断地学习，不断地发现问题，思考问题，进而解决问题。