

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	2210204	姓名	夏雨锳	专业	计算机科学与技术
项目名称	教务管理系统				
必备环境	Python3.8,MySQL,tkinter8.3				
系统主要功能简介（4 分）	实现了教务管理系统，分为老师和学生两部分，老师具有学生管理，课程管理，密码修改，信息修改，课程学生查询的功能。学生具有查询成绩，选课，修改密码，修改信息的功能。				
系统主要页面截图（6 分）	<div><div>教务管理系统</div><div>教务系统登录</div><div>账号：<input type="text"/></div><div>密码：<input type="password"/></div><div>登录退出</div></div> <div><div>教务管理系统</div><div>学号/工号：2210210</div><div>姓名：花泽</div><div>性别：女</div><div>出生日期：1970-01-01</div><div>联系方式：123-456-8002</div><div>功能选择</div><div>学生管理课程管理修改密码修改信息</div><div>课程学生查询</div></div>				



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	1. 安装 MySQL 数据库。			
		2. 在 navicat 创建数据库 school_db 。			
		3 创建所需的表和视图。			
	高级语言	1. 安装 Python 及其相关库（pymysql,tkinter 等） 2. 编写和调试 Python 程序以连接和操作数据库。			
连接串 分析 (6 分)		序号	名称	功能说明	取值
		1	主机名	指定数据库服务器的主机名或 IP 地址	localhost
		2	端口号	指定数据库服务器监听的端口	3306
		3	用户名	用于连接数据库的用户名	root
		4	密码	连接数据库的密码	Xyn20040516!
		5	数据库名	指定要连接的数据库	school_db
连接串代码 (截屏) (2 分)		<pre>class Register: 新* def __init__(self, master): self.root = master # 窗口传入 # 数据库登录 self.ip = 'localhost' self.port = 3306 self.id = 'root' self.pd = 'Xyn20040516!' self.db = 'school_db' # 个人信息</pre>			
备注		无			

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	student	sno	无	无
	2	course	cno	无	无
	3	student_course	(sno, tcid)	sno, tcid	student (sno), course (cno)

	4	Student_pwd	user	user	student (sno)
	5	teacher	tno	无	无
	6	Teacher_pwd	user	user	teacher (tno)
关系图 (4)					
备注					

4. 含有事务应用的删除操作（13 分）

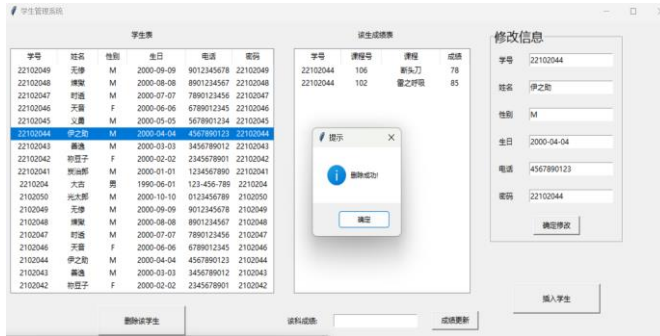
说明	(1 分) 简要说明该操作所要完成的功能; (2 分) 该操作会涉及的表（必须含有两张或两张以上的关系表，同时以“表名”的形式给出） (1 分) 表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”） (1 分) 删除条件涉及的字段描述（以“表名. 属性=? ”形式给出） (4 分) 实现该操作的关键代码（高级语言、SQL），截图即可；（其中如果删除语句中不包含任何形式的事务应用将扣除 3 分） (4 分) 如何执行该操作，按所述方法能够正常演示程序则给分。	
功能描述 (1 分)	在学生管理界面中，选中需要删除的学生后，点击“删除该学生”按钮。程序将会执行代码，从 student 中删除该学生，同时从数据库 student_pwd 表中删除选中的学生及其密码信息。	
涉及的表 (2 分)	student, student_pwd, student_course	
表连接涉及字段 (1 分)	student.sno = student_pwd.user = student_course.sno	
删除条件 字段描述 (1 分)	字段	规则
	student.sno = ?	student.sno 等于删除学生的 sno
	student_pwd.user = ?	student_pwd.user 等于删除学生的 sno
	student_course.sno = ?	student_course.sno 等于删除学生的 sno

代码
(4分)

```
def shanage_delete(self):
    if self.tempary_sno != '':
        # 连接数据库
        self.connect = pymysql.connect(host=self.ip, port=self.port, user=self.user, password=self.passed, db=self.db)
        try:
            if self.connect:
                print('连接成功')
                print(self.no) # 用户名, 密码等
                # 开始事务
                self.connect.begin()
                # 删除语句
                delete_student_course_sql = "DELETE FROM student_course WHERE sno=%s"
                delete_student_sql = "DELETE FROM student WHERE sno=%s"
                delete_student_pwd_sql = "DELETE FROM student_pwd WHERE user=%s"
                # 创建游标
                self.cursor2 = self.connect.cursor()
                # 删除学生课程记录
                self.cursor2.execute(delete_student_course_sql, [self.tempary_sno])
                # 删除学生密码记录
                self.cursor2.execute(delete_student_pwd_sql, [self.tempary_sno])
                # 最后删除学生记录
                self.cursor2.execute(delete_student_sql, [self.tempary_sno])
                self.connect.commit()
                messagebox.showinfo(title='提示', message='删除成功!')
                # 重置treeview3
                self.del_button(self.treeview3)
                search_sql = """
                SELECT sno, name, sex, birthday, tel, pwd
                FROM student
                INNER JOIN student_pwd ON student.sno = student_pwd.user
                """
                self.cursor2.execute(search_sql)
                self.row = self.cursor2.fetchone() # 获取查询结果
                while self.row:
                    self.treeview3.insert('', END, B, values=(
                        self.row[0], self.row[1], self.row[2], self.row[3], self.row[4], self.row[5]))
                    self.row = self.cursor2.fetchone()
            except pymysql.Error as e:
                # 出现异常时回滚
                self.connect.rollback()
                messagebox.showinfo(title='数据库错误', message=str(e))
            finally:
                self.cursor2.close()
                self.connect.close()
        else:
            messagebox.showinfo(title='提示', message='未选中, 请选中学生!')
```

程序演示
(4分)

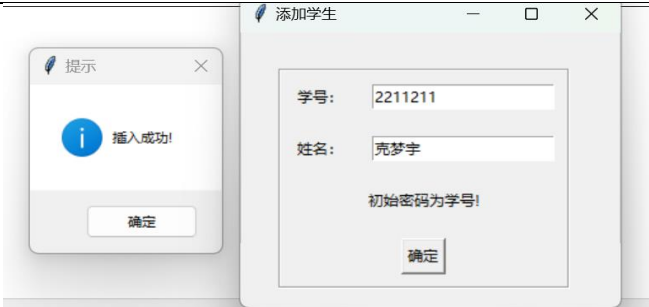
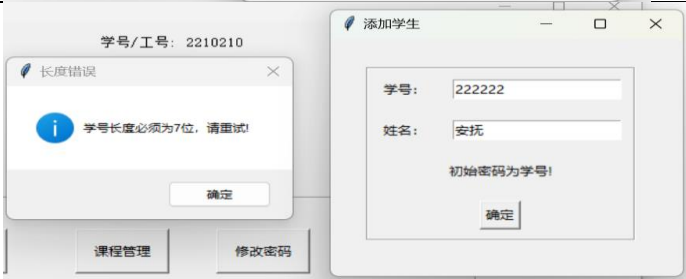
首先是选中学生进行删除，然后查看学生表和学生密码表，发现都被删除。



备注 三个删除字段都满足才开始删除

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）	在添加学生时，当学生学号不符合要求时会触发触发器报错。	
触发器描述 （2 分）	触发器会检查学号的长度，如果长度不为 7，就会触发，从而报错。	
涉及的表 （1 分）	Student	
输入数据 （2 分）	字段	规则
	学号	长度必须为 7 位
插入操作 源码 （3 分）	<pre>def smanage_insert(self): root_window = Toplevel(self.root) # 初始化的声明 root_window.geometry('300x250+100+100') root_window.title('添加学生') # 界面 self.frame_smanage_insert = LabelFrame(root_window) self.frame_smanage_insert.grid(padx=30, pady=30) # Label self.label_smanage_insert_sno = Label(self.frame_smanage_insert, text='学号: ') self.label_smanage_insert_sno.grid(row=0, column=0, padx=10, pady=10) self.label_smanage_insert_sname = Label(self.frame_smanage_insert, text='姓名: ') self.label_smanage_insert_sname.grid(row=1, column=0, padx=10, pady=10) self.label_smanage_insert_pwd = Label(self.frame_smanage_insert, text='初始密码为学号!') self.label_smanage_insert_pwd.grid(row=2, column=0, columnspan=2, padx=10, pady=10) # 输入框 self.var_smanage_insert_sno = StringVar self.var_smanage_insert_sname = StringVar self.entry_smanage_insert_sno = Entry(self.frame_smanage_insert, textvariable=self.var_smanage_insert_sno) self.entry_smanage_insert_sno.grid(row=0, column=1, padx=10, pady=10) self.entry_smanage_insert_sname = Entry(self.frame_smanage_insert, textvariable=self.var_smanage_insert_sname) self.entry_smanage_insert_sname.grid(row=1, column=1, padx=10, pady=10) # 按钮 self.button_ok = Button(self.frame_smanage_insert, text='确定', command=self.smanage_insert_ok) self.button_ok.grid(row=3, column=0, columnspan=2, padx=10, pady=10)</pre>	
触发器源码 （3 分）	<pre>except pymysql.Error as e: # 检查错误信息是否包含触发器的错误信息 if '学号长度必须为7位' in str(e): messagebox.showinfo(title='长度错误', message='学号长度必须为7位。请重试!') else: messagebox.showinfo(title='数据库错误', message=str(e)) finally: self.cursor2.close() self.connect.close() DROP TRIGGER IF EXISTS `check_sno_length`; delimiter ;; CREATE TRIGGER `check_sno_length` BEFORE INSERT ON `student` FOR EACH ROW BEGIN IF CHAR_LENGTH(NEW.sno) != 7 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '学号长度必须为7位'; END IF; END ;; delimiter ; SET FOREIGN_KEY_CHECKS = 1;</pre>	

程序演示 (4 分)	
程序演示 (4 分)	
备注	

6. 存储过程控制下的更新操作（18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。	
功能描述 （1分）	实现课程名称的更新，同时更新所有选修该课程的学生课程信息	
存储过程 功能描述 （1分）	该存储过程更新课程名称，并确保相关学生课程信息也同步更新。	
涉及的关系表 （2分）	course, tudent_course	
表连接涉及 及字段 （1分）	course.cno = student_course.tcid	
更改字段 （2分）	字段	规则
	course.cno	当修改条件触发，且 course.cno 等于被修改的 course.cno
	student_course.tcid	当修改条件触发，且 student_course.tcid 等于被修改的 student_course.tcid
更新代码 （3分）	<pre> try: if self.connect: print('连接成功') print(self.no) # 开始事务 self.connect.begin() # 调用存储过程更新课程信息和学生课程信息 self.cursor2.execute(sql_call_proc, args=(self.temporary_cno, self.entry_cname.get())) # 提交事务 self.connect.commit() messagebox.showinfo(title='提示', message='信息修改成功!') # 再次treeview5中读 sql_up3 = "SELECT cno, cname, credit FROM course" self.del_button(self.treeview5) # 插入查询结果 self.cursor2.execute(sql_up3) self.row = self.cursor2.fetchone() # 读取查询结果 while self.row: self.treeview5.insert(parent='', index=0, values=(self.row[0], self.row[1], self.row[2])) self.row = self.cursor2.fetchone() except pymysql.Error as e: # 出现异常时回滚 self.connect.rollback() messagebox.showinfo(title='数据库错误', message=str(e)) finally: self.cursor2.close() self.connect.close() else: messagebox.showinfo(title='提示', message='请勿修改课程号!') </pre>	

创建存储
过程源码
(3 分)

```
DROP PROCEDURE IF EXISTS 'update_course_and_student_course';
delimiter ;;
CREATE PROCEDURE 'update_course_and_student_course'(IN course_no CHAR(10),
IN new_course_name CHAR(20))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '存储过程执行失败';
    END;

    START TRANSACTION;

    -- 更新课程表中的课程名称
    UPDATE course
    SET cname = new_course_name
    WHERE cno = course_no;

    -- 更新学生课程表中的课程名称
    UPDATE student_course sc
    JOIN course c ON sc.ctcid = c.cno
    SET sc.ctcid = c.cno
    WHERE c.cno = course_no;

    COMMIT;
END
;;
delimiter ;
```

```
DROP PROCEDURE IF EXISTS 'update_course_and_student_course';
delimiter ;;
CREATE PROCEDURE 'update_course_and_student_course'(IN course_no CHAR(10),
IN new_course_name CHAR(20))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '存储过程执行失败';
    END;

    START TRANSACTION;

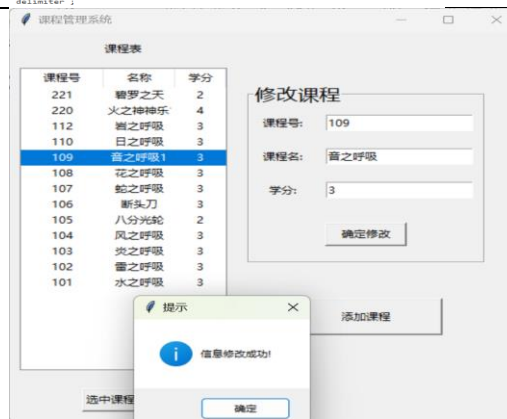
    -- 更新课程表中的课程名称
    UPDATE course
    SET cname = new_course_name
    WHERE cno = course_no;

    -- 更新学生课程表中的课程名称
    UPDATE student_course sc
    JOIN course c ON sc.ctcid = c.cno
    SET sc.ctcid = c.cno
    WHERE c.cno = course_no;

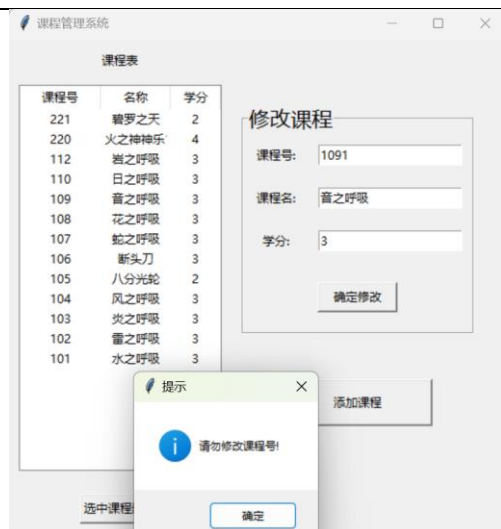
    COMMIT;
END
;;
delimiter ;
```

存储过程
执行源码
(1 分)

程序演示
(2 分)



程序演示
(2 分)



备注

7. 含有视图的查询操作（15 分）

说明	<p>（1 分）简要说明该操作所要完成的功能；</p> <p>（1 分）简要说明建立的该视图的功能；</p> <p>（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）</p> <p>（1 分）简要说明表连接涉及的字段（以“表 1. 属性=表 2. 属性”）</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>（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>
操作功能描述（1 分）	实现查询某门课程的所有选课学生信息及其成绩。
视图功能描述（1 分）	创建一个视图，将课程表、学生表和学生课程表连接起来，以便能够查询某门课程的所有选课学生信息及其成绩。
涉及的关系表（2 分）	<ul style="list-style-type: none"> course（课程表） student（学生表） student_course（学生课程表）
表连接字段（1 分）	<ul style="list-style-type: none"> course.cno = student_course.tcid student.sno = student_course.sno
创建视图代码（3 分）	<pre> DROP VIEW IF EXISTS `course_student_view`; CREATE ALGORITHM = UNDEFINED SQL SECURITY DEFINER VIEW `course_student_view` AS select `c`.`cno` AS `cno`, `c`.`cname` AS `cname`, `s`.`sno` AS `sno`, `s`.`sname` AS `sname`, `sc`.`score` AS `score` from ((`course` `c` join `student_course` `sc` on((`c`.`cno` = `sc`.`tcid`))) join `student` `s` on((`sc`.`sno` = `s`.`sno`))); </pre>
查询代码（3 分）	<pre> def query_course_students(self): """ 查询课程学生信息 """ def fetch_course_students(): course_no = entry_course_no.get() # 连接数据库 self.connect = pymysql.connect(host=self.ip, port=self.port, user=self.id, password=self.pd, db=self.db) if self.connect: print('连接成功') # 查询语句 search_sql = "SELECT * FROM course_student_view WHERE cno = %s" # 创建游标 self.cursor2 = self.connect.cursor() self.cursor2.execute(search_sql, [course_no]) self.row = self.cursor2.fetchone() # 获取查询结果 # 清空 treeview for item in self.treeview.get_children(): self.treeview.delete(item) # 插入查询结果 while self.row: self.treeview.insert('', index=0, values=(self.row[0], self.row[1], self.row[2], self.row[3], self.row[4])) self.row = self.cursor2.fetchone() self.cursor2.close() self.connect.close() # 创建查询窗口 query_window = Tk() query_window.geometry('400x400+100+100') query_window.title('课程学生查询') # 输入框和按钮 label_course_no = Label(query_window, text='课程号:', font=('黑体', 12)) label_course_no.grid(row=0, column=0, padx=10, pady=10) entry_course_no = Entry(query_window) entry_course_no.grid(row=0, column=1, padx=10, pady=10) button_fetch = Button(query_window, text='查询', command=fetch_course_students) button_fetch.grid(row=0, column=2, padx=10, pady=10) # 表格 columns = ("课程号", "课程名", "学生号", "学生名", "成绩") self.treeview = ttk.Treeview(query_window, height=18, show='headings', columns=columns) self.treeview.column("课程号", width=100, anchor='center') self.treeview.column("课程名", width=150, anchor='center') self.treeview.column("学生号", width=100, anchor='center') self.treeview.column("学生名", width=150, anchor='center') self.treeview.column("成绩", width=100, anchor='center') self.treeview.heading("课程号", text="课程号") self.treeview.heading("课程名", text="课程名") self.treeview.heading("学生号", text="学生号") self.treeview.heading("学生名", text="学生名") self.treeview.heading("成绩", text="成绩") self.treeview.grid(row=1, column=0, columnspan=3, padx=10, pady=10) query_window.mainloop() </pre>

程序演示 (4 分)	<div><div><div>教务管理系统</div><div>学号/工号: 2210210 姓名: 花泽 性别: 女 出生日期: 1970-01-01 联系方式: 123-456-8002</div><div>功能选择</div><div><div>学生管理</div><div>课程管理</div><div>修改密码</div><div>修改信息</div><div>课程学生查询</div></div></div><div><div>课程学生查询</div><div>课程号: 221 查询</div><table><tr><th>课程号</th><th>课程名</th><th>学生号</th><th>学生名</th><th>成绩</th></tr><tr><td>221</td><td>碧罗之天</td><td>2211111</td><td>夏雨诺</td><td>100</td></tr><tr><td>221</td><td>碧罗之天</td><td>1111111</td><td>冯言旭</td><td>100</td></tr></table></div></div>	课程号	课程名	学生号	学生名	成绩	221	碧罗之天	2211111	夏雨诺	100	221	碧罗之天	1111111	冯言旭	100
课程号	课程名	学生号	学生名	成绩												
221	碧罗之天	2211111	夏雨诺	100												
221	碧罗之天	1111111	冯言旭	100												
备注																