数据库工程作业

要求：

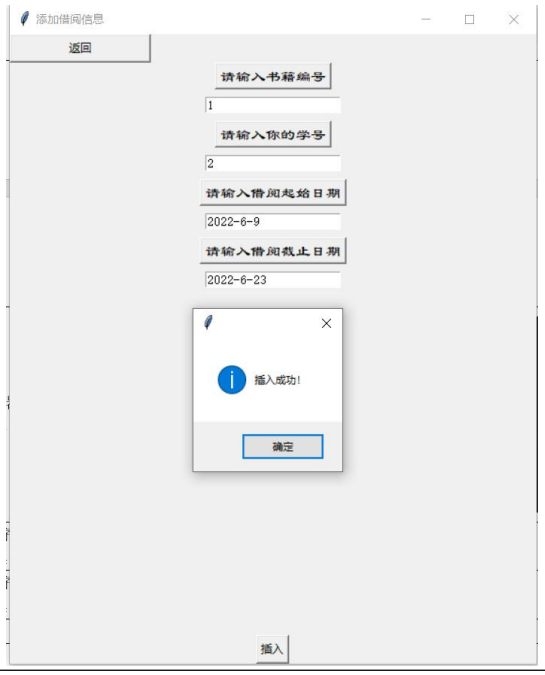
1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
3. 报告中所列举的四种操作，每种操作举一个例子即可。
4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

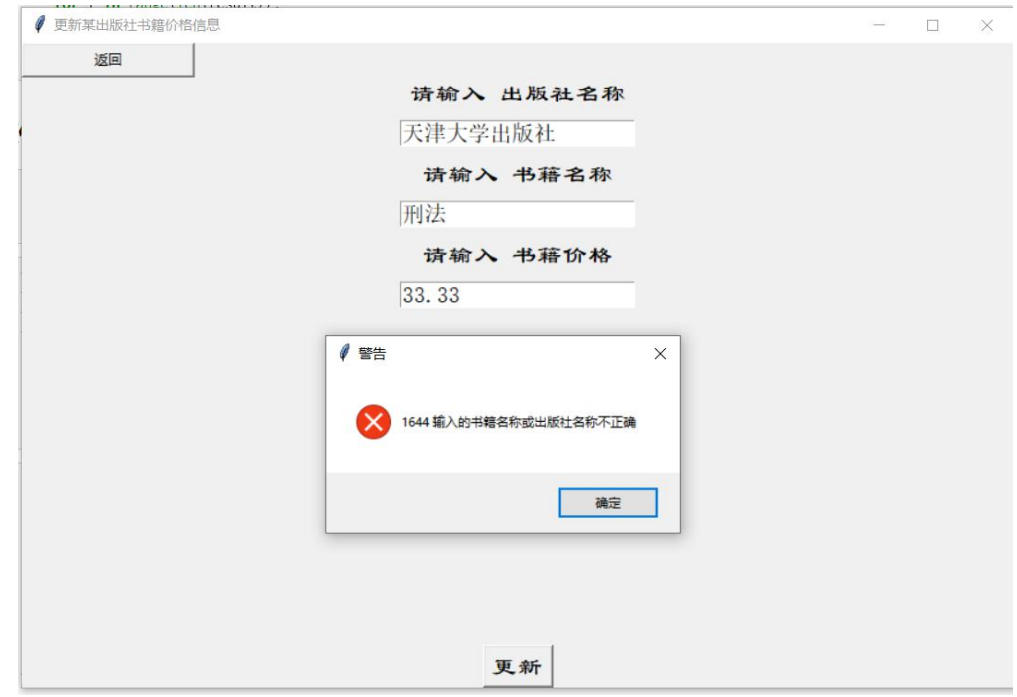
工程作业报告

1. 项目信息（10 分）

学号	2011440	姓名	杨馨仪	专业	计算机科学与技术
项目名称	图书馆书籍管理系统				
必备环境	Ver 8.0.29 for Win64 on x86_64 (MySQL Community Server - GPL) Python 3.10.4 Navicat Premium 15 Jupyter notebook 6.4.8				
系统主要功能简介（4 分）	1. 实现对信息（书籍借阅信息）的增添 2. 实现对信息（某楼层书籍所属信息）的删除 3. 实现对信息（某出版社书籍价格）的更改 4. 实现对信息（作者著作数量）的查询				
系统主要页面截图（6 分）					


删除某楼层书籍所属信息
返回
请输入 图书馆楼层
4
类别编号 书籍编号 类别名称 所在楼层
1 1 计算机 4
1 3 计算机 4
1 4 计算机 4
5 10 体育 4
确定
取消
删除


添加借阅信息
返回
请输入书籍编号
1
请输入你的学号
2
请输入借阅起始日期
2022-6-9
请输入借阅截止日期
2022-6-23
插入成功!
确定
插入



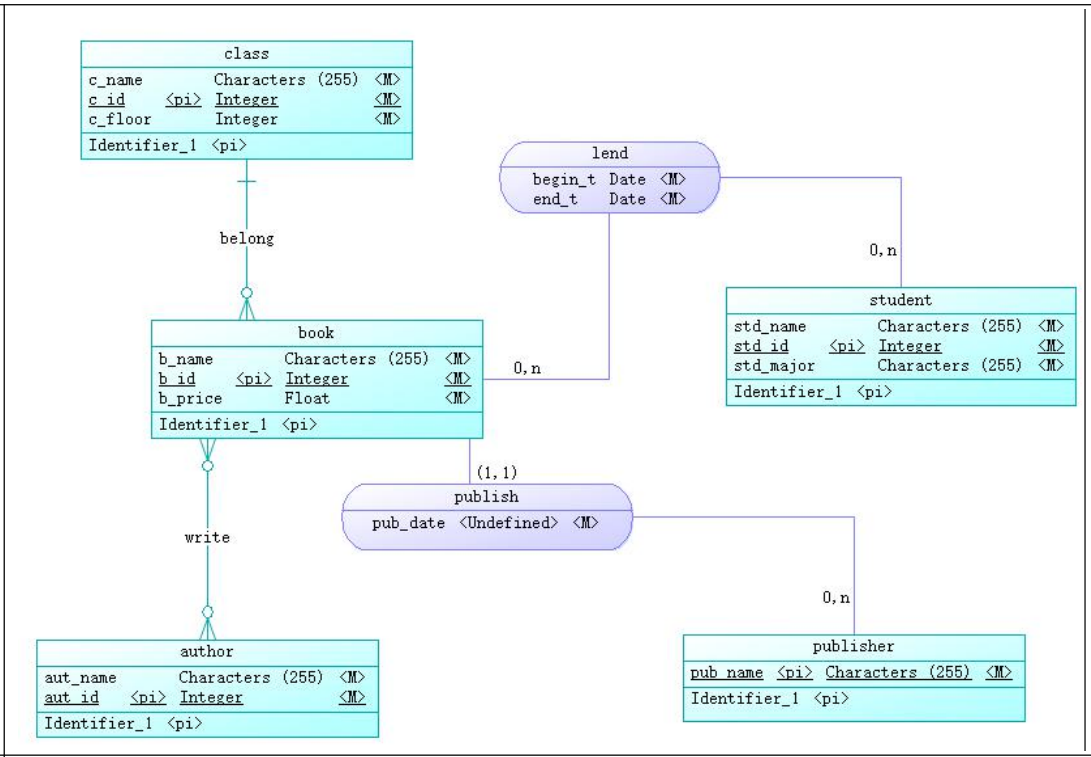
2. 系统配置（10 分）

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. 数据库设计：Ver 8.0.29 for Win64 on x86_64 (MySQL Community Server - GPL)			
		2. 数据库可视化(辅助操作)：Navicat Premium 15			
		3. ER 图绘制：PowerDesigner			
	高级语言	1. 高级开发语言：Python 3.10.4			
		2. 可视化编译工具：Jupyter notebook 6.4.8			
连接串		序号	名称	功能说明	取值

分析 （6 分）	1	host	要连接的主机地址	“localhost”
	2	user	用于登录的数据库用户	“root”
	3	password	密码	“yxy020819”
	4	database	要连接的数据库	“library”
	5	port	端口，一般为 3306	3306
	6	Autocommit	是否自动提交事务	“True”
连接串代码 （截屏） （2 分）				
备注				

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	book	b_id	None	None
	2	student	std_id	None	None
	3	lend	(b_id,std_id,begin_t)	b_id	book(b_id)
				std_id	student(std_id)
	4	class	c_id	None	None
	5	belong	(c_id,b_id)	c_id	class(c_id)
				b_id	book(b_id)
	6	author	aut_id	None	None
	7	write	(aut_id,b_id)	aut_id	author(aut_id)
				b_id	book(b_id)
	8	publisher	pub_name	None	None
	9	publish	(pub_name,b_id)	pub_name	publisher(pub_name)
				b_id	book(b_id)

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）	通过用户输入楼层，可以删除该楼层所有书籍在 belong 表中的所属信息 （现实中可以用于图书馆书籍位置调整，图书馆楼层装修等问题） 同时该删除操作增加了提交与回滚功能，用于避免实际应用过程中错输，误触等行为的发生。	
涉及的表（2 分）	Belong,class	
表连接涉及字段（1 分）	Belong.c_id=class.c_id;(书籍所属类别编号与类别编号相同)	
删除条	字段	规则

件字段描述（1分）	Class.c_floor=?	由图书馆书籍借阅管理系统的用户输入获取，根据 belong 表与 class 表的连接，删除属于该楼层的书籍的全部所属信息。
代码（4分）	（截屏） **4.0删除界面** **主界面** In [7]: ``` # 删除信息 def delete(): otherFrame1 = tk.Toplevel() otherFrame1.geometry('700x450') otherFrame1.title("删除信息") handle = lambda: onCloseOtherFrame(window, otherFrame1) b = lambda: delete_book(otherFrame1) fb = lambda: delete_floor_book(otherFrame1) tk.Button(otherFrame1, text='返回', width=20, command=handle).pack(side='bottom', anchor='center') tk.Button(otherFrame1, text='删除书籍信息', font=('黑体', 15), width=25, command=b).pack(padx=5, pady=20) tk.Button(otherFrame1, text='删除某楼层书籍所属信息', font=('黑体', 15), width=25, command=fb).pack(padx=5, pady=20) ``` **4.2删除图书馆某层书的所属信息** In [11]: ``` # 删除图书馆某层书的所属信息 def delete_floor_book(oF): otherFrame = tk.Toplevel() otherFrame.geometry('700x450') otherFrame.title("删除某楼层书籍所属信息") handle = lambda: onCloseOtherFrame(oF, otherFrame) tk.Button(otherFrame, text='返回', width=20, command=handle).pack(side='top', anchor='nw') tk.Label(otherFrame, text="请输入 图书馆楼层", font=('黑体', 16)).pack() e1 = tk.StringVar() tk.Entry(otherFrame, show='', textvariable=e1, font=('宋体', 14)).pack(padx=5, pady=10) click = lambda: delete_floor_book_sql(otherFrame, oF, e1) tk.Button(otherFrame, text='删除', command=click, font=('黑体', 16)).pack(side='bottom') ``` In [12]: ``` def delete_floor_book_sql(oF, oF2, e1): floor = e1.get() tree = ttk.Treeview(oF) title = ["类别编号","书籍编号","类别名称","所在楼层"] tree["columns"] = ("类别编号","书籍编号","类别名称","所在楼层") for i in title: tree.column(i, anchor="center") tree.heading(i, text=i) temp = "SELECT * FROM belong natural join class WHERE c_floor = '" + floor + "';" cursor.execute(temp) result = cursor.fetchall() for i in range(len(result)): tree.insert("", i, values=result[i]) tree['show'] = 'headings' tree.pack(anchor="center", padx=5, pady=10) sql = "DELETE belong FROM belong natural join class WHERE c_floor = '" + floor + "';" click = lambda: cancel(oF, oF2, sql) tk.Button(oF, text='取消', command=click, font=('黑体', 12)).pack(side='bottom') click = lambda: insist(oF, oF2, sql) tk.Button(oF, text='确定', command=click, font=('黑体', 12)).pack(side='bottom') ``` **确认还是取消** In [8]: ``` def cancel(otherFrame, oF2, sql): cursor.execute("START TRANSACTION") cursor.execute(sql) cursor.execute("ROLLBACK") messagebox.showinfo(message='已取消') onCloseOtherFrame(oF2, otherFrame) def insist(otherFrame, oF2, sql): cursor.execute("START TRANSACTION") cursor.execute(sql) cursor.execute("COMMIT") messagebox.showinfo(message='已删除') onCloseOtherFrame(oF2, otherFrame) ```
程序演示（4分）	输出界面截图：	


删除某楼层书籍所属信息
返回
请输入 图书馆楼层
5
类别编号
书籍编号
类别名称
所在楼层
3
8
心理学
5
3
19
心理学
5
确定
取消
删除

取消删除：


删除某楼层书籍所属信息
返回
请输入 图书馆楼层
5
类别编号
书籍编号
类别名称
所
3
8
心理学
3
19
心理学
已取消
确定
确定
取消
删除

确认删除：

	 删除某楼层书籍所属信息 返回 请输入 图书馆楼层 5 类别编号 书籍编号 类别名称 所在楼层 3 心理学 5 3 心理学 5 已删除 确定 确定 取消 删除
备注	

5. 触发器控制下的添加操作（20 分）

<table>
<tr><td>说明</td><td>（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>通过给出借阅者学号，借阅书籍编号，借阅开始时间与借阅截止时间添加书籍借阅信息到 lend 表。</td></tr>
<tr><td>触发器描述
（2 分）</td><td>触发器用于判断输入的日期是否有效。如果开始日期大于当前时间 CURTIME()或者截止日期早于开始日期，则判定当前借阅信息无效，并且提示“借阅时间输入不合法”。</td></tr>
<tr><td>涉及的表
（1 分）</td><td>Lend</td></tr>
<tr><td rowspan="5">输入数据
（2 分）</td><td>字段</td><td>规则</td></tr>
<tr><td>B_id</td><td>书籍编号参照 book 表的 b_id，必须满足书籍编号为 book 表中已经存在的书籍编号</td></tr>
<tr><td>Std_id</td><td>学号参照 student 表中的 std_id，必须满足学号为 student 表中已经存在的学号</td></tr>
<tr><td>Begin_t</td><td>开始日期必须在当前时间以前，借阅开始时间不应当在未来</td></tr>
<tr><td>End_t</td><td>结束日期必须晚于开始时间，不得提早于开始时间</td></tr>
<tr><td>插入操作
源码
（3 分）</td><td colspan="2">

5. 触发器控制下的添加操作

5.0主界面

In [53]:

```
# 添加信息
def append():
    otherFrame1 = tk.Toplevel()
    otherFrame1.geometry('960x600')
    otherFrame1.title("添加信息")
    handle = lambda: onCloseOtherFrame(window, otherFrame1)
    p = lambda: append_lend(otherFrame1)
    t = lambda: append_book(otherFrame1)
    tk.Button(otherFrame1, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    tk.Button(otherFrame1, text='添加借阅信息', font=('隶书', 14), width=20, command=p).pack(padx=5, pady=5)
    tk.Button(otherFrame1, text='添加书籍信息', font=('隶书', 14), width=20, command=t).pack(padx=5, pady=5)
```

5.1添加借阅信息

In [54]:

```
# 添加借阅信息
def append_lend(oF):
    otherFrame = tk.Toplevel()
    otherFrame.geometry('560x660')
    otherFrame.title("添加借阅信息")
    handle = lambda: onCloseOtherFrame(oF, otherFrame)
    tk.Button(otherFrame, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    tk.Button(otherFrame, text="请输入书籍编号", font=('隶书', 12)).pack()
    e3 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e3, font=('宋体', 10)).pack(padx=4, pady=8)
    tk.Button(otherFrame, text="请输入你的学号", font=('隶书', 12)).pack()
    e1 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e1, font=('宋体', 10)).pack(padx=4, pady=8)
    tk.Button(otherFrame, text="请输入借阅起始日期", font=('隶书', 12)).pack()
    e2 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e2, font=('宋体', 10)).pack(padx=4, pady=8)
    tk.Button(otherFrame, text="请输入借阅截止日期", font=('隶书', 12)).pack()
    e8 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e8, font=('宋体', 10)).pack(padx=4, pady=8)
    click = lambda: insert_lend(otherFrame, e3, e1, e2, e8)
    tk.Button(otherFrame, text='插入', command=click).pack(side='bottom')
```

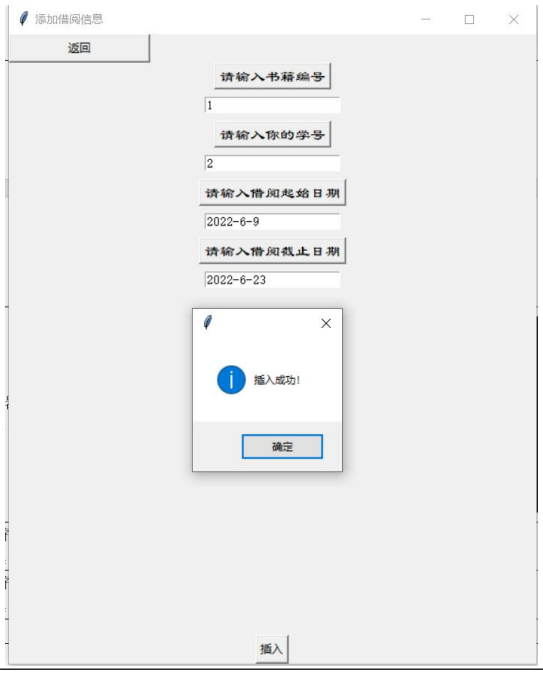
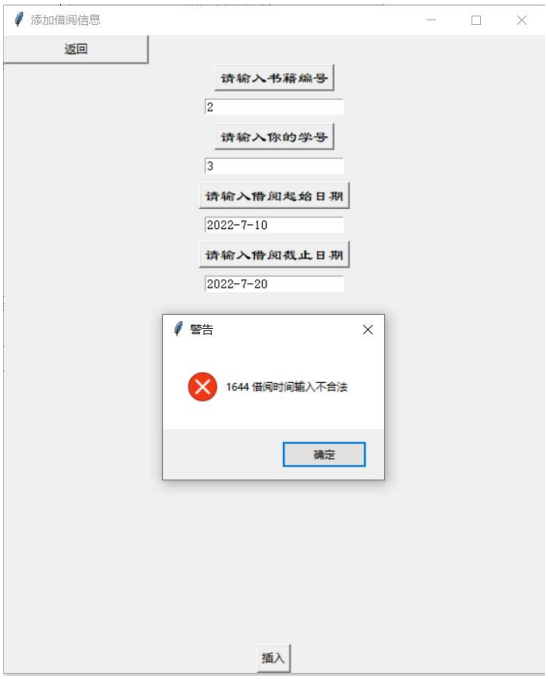
In [55]:

```
def insert_lend(otherFrame, e3, e1, e2, e8):
    b_id= e3.get()
    std_id = e1.get()
    begin_t = e2.get()
    end_t = e8.get()
    sql = "INSERT INTO lend values('" + b_id + "', '" + std_id + "', '" + begin_t + "', '" + end_t +  "');"
    print( "INSERT INTO lend values('" + b_id + "', '" + std_id + "', '" + begin_t + "', '" + end_t +  "');")
    try:
        cursor.execute(sql)
        messagebox.showinfo(message='插入成功！')
    except Exception as m:
        messagebox.showerror('警告', m.args)
```

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

<table>
<tr><td>触发器源码
（3 分）</td><td>

```
DROP trigger IF EXISTS `check_date`;
delimiter //
CREATE trigger `check_date` BEFORE INSERT ON lend for each ROW
BEGIN
IF new.begin_t>curtime() OR new.begin_t>new.end_t
THEN
signal sqlstate '22003' SET message_text='借阅时间输入不合法';
END IF;
END
//
DELIMITER ;
```

</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>
<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>根据提供的书籍名称与出版社名称实现对书籍价格的更新。
（现实中用于出版社对书籍价格做出更改的情况）</td></tr>
<tr><td>存储过程功能描述（1 分）</td><td>用户输入出版社名称与书籍名称，以及想要更改的价格，如果输入的出版社名称或书籍名称不存在则提示“输入的书籍名称或出版社名称不正确”，如果出版社没有发行过该书籍（由用户输入）则不进行改价操作，如果存在该出版社发行的该书籍，则把相应的价格改为用户输入的价格。</td></tr>
<tr><td>涉及的关系表(2 分)</td><td>Book,publish</td></tr>
<tr><td>表连接涉及字段(1)</td><td>Book.b_id=publish.b_id(书籍编号，与出版表中的书籍编号相同)</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>Book.price</td><td>如果输入的出版社名称或书籍名称不存在则提示“输入的书籍名称或出版社名称不正确”，如果出版社没有发行过该书籍（由用户输入）则不进行改价操作，如果存在该出版社发行的该书籍，则把相应的价格改为用户输入的价格。</td></tr></table></td></tr>
<tr><td>更新代码（3 分）</td><td>

6. 存储过程控制下的更新操作

In [76]:
```
# 更新信息
def update():
    otherFrame1 = tk.Toplevel()
    otherFrame1.geometry('960x600')
    otherFrame1.title("更新信息")
    handle = lambda: onCloseOtherFrame(window, otherFrame1)
    p = lambda: update_price(otherFrame1)
    tk.Button(otherFrame1, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    tk.Button(otherFrame1, text='更新 某出版社书籍价格信息', font=('隶书', 16), width=30, command=p).pack(padx=5, pady=5)
```

In [77]:
```
# 更新某出版社书籍价格信息
def update_price(oF):
    otherFrame = tk.Toplevel()
    otherFrame.geometry('1440x800')
    otherFrame.title("更新某出版社书籍价格信息")
    handle = lambda: onCloseOtherFrame(oF, otherFrame)
    tk.Button(otherFrame, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    tk.Label(otherFrame, text="请输入 出版社名称", font=('隶书', 16)).pack()
    e1 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e1, font=('宋体', 14)).pack(padx=5, pady=10)
    tk.Label(otherFrame, text="请输入 书籍名称", font=('隶书', 16)).pack()
    e2 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e2, font=('宋体', 14)).pack(padx=5, pady=10)
    tk.Label(otherFrame, text="请输入 书籍价格", font=('隶书', 16)).pack()
    e3 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e3, font=('宋体', 14)).pack(padx=5, pady=10)
    click = lambda: update_price_sql(otherFrame, e1,e2,e3)
    tk.Button(otherFrame, text='更新', command=click, font=('隶书', 16)).pack(side='bottom')
```

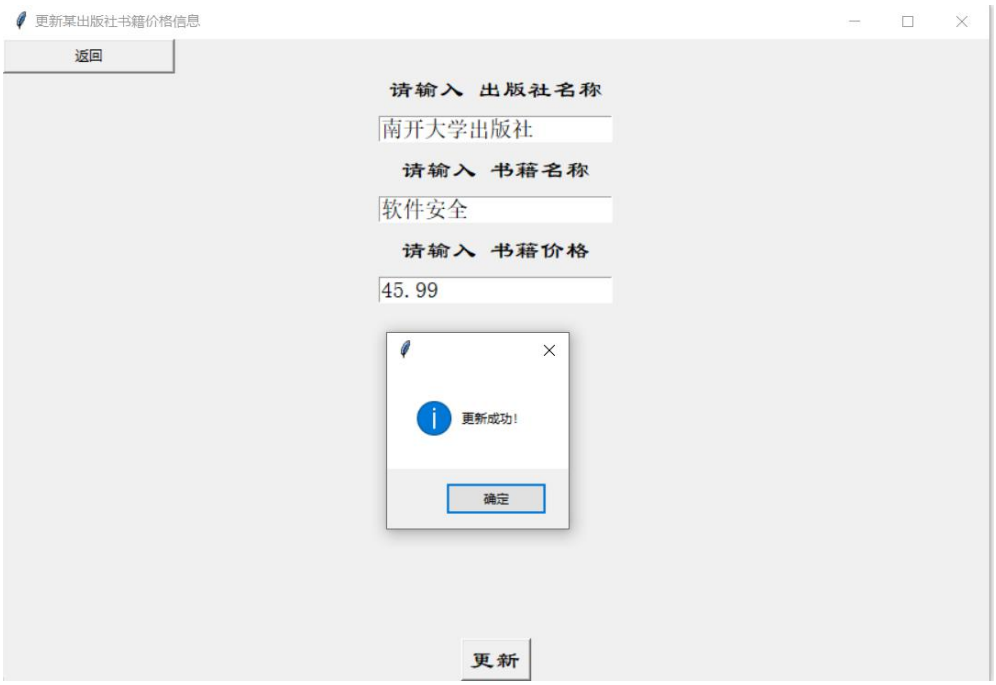
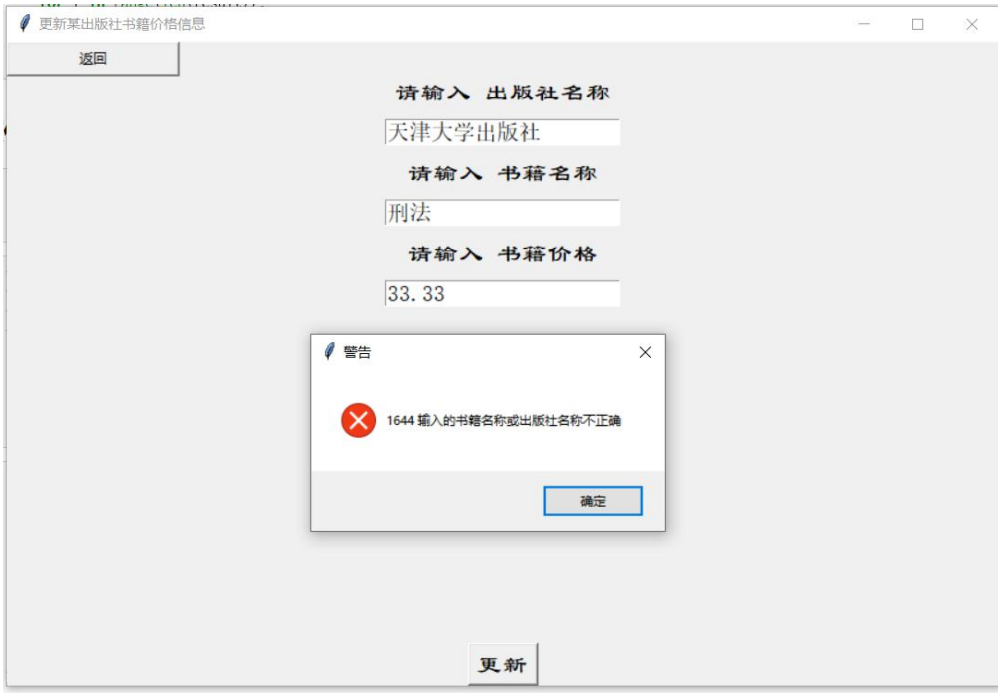
</td></tr>
<tr><td>创建存储过程源码（3 分）</td><td>

```
DROP PROCEDURE IF EXISTS `book_update`;
delimiter //
CREATE PROCEDURE `book_update`(in `publish_name` varchar(255),in `book_name` varchar(255),in `book_price` float(10,2))
IF book_name NOT IN (SELECT b_name FROM book) OR publish_name NOT IN (SELECT pub_name FROM publish)
THEN signal sqlstate '22003' set message_text='输入的书籍名称或出版社名称不正确';
ELSE
UPDATE `book` SET `b_price`=book_price WHERE `b_name`=book_name AND `b_id` IN (SELECT b_id FROM publish WHERE pub_name=publish_name);
END IF;
END
//
delimiter ;
```

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

<table>
<tr><td>存储过程
执行源码
（1 分）</td><td>

```
In [78]:  def update_price_sql(oF, e1, e2, e3):
              publish_name = e1.get()
              book_name = e2.get()
              book_price = e3.get()
              try:
                  sql = "call book_update('" + publish_name + "', '" + book_name + "', '" + book_price + "');"
                  cursor.execute(sql)
                  messagebox.showinfo(message='更新成功!')
              except Exception as m:
                  messagebox.showerror('警告', m.args)
```

</td></tr>
<tr><td>程序演示
（2 分）</td><td>说明：不违背存储过程，能够执行更新操作
 
</td></tr>
<tr><td>程序演示
（2 分）</td><td>说明：违背存储过程，系统报错；
 
</td></tr>
<tr><td>备注</td><td></td></tr>
</table>

7. 含有视图的查询操作（15 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。</td></tr>
<tr><td>操作功能描述（1 分）</td><td>实现对图书馆中 A 作者有多少作品进行查询。</td></tr>
<tr><td>视图功能描述（1 分）</td><td>建立视图，连接 author 表与 write 表，存储每个作者的姓名以及图书馆中有几本该作者的著作。</td></tr>
<tr><td>涉及的关系表（2 分）</td><td>Author,write</td></tr>
<tr><td>表连接字段（1 分）</td><td>Author.aut_id=write.aut_id</td></tr>
<tr><td>创建视图代码（3 分）</td><td>

```
-- 创建视图
DROP view IF EXISTS `author_find`;
CREATE view `author_find`(author_name,countworks) AS
SELECT author.aut_name,count(*)
FROM `write` natural join `author`
GROUP BY author.aut_id;
```

</td></tr>
<tr><td>查询代码（3 分）</td><td>

7. 含有视图的查询操作

In [103]:

```
# 查询信息
def search():
    otherFrame1 = tk.Toplevel()
    otherFrame1.geometry('960x600')
    otherFrame1.title("查询信息")
    handle = lambda: onCloseOtherFrame(window, otherFrame1)
    tk.Button(otherFrame1, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    p = lambda: search_book(otherFrame1)
    tk.Button(otherFrame1, text='查询某作者著作数量', font=('隶书', 14), width=20, command=p).pack(padx=5, pady=5)
```

In [109]:

```
# 查询书籍信息
def search_book(oF):
    otherFrame = tk.Toplevel()
    otherFrame.geometry('960x600')
    otherFrame.title("查询某作者著作数量")
    handle = lambda: onCloseOtherFrame(oF, otherFrame)
    tk.Button(otherFrame, text='返回', width=20, command=handle).pack(side='top', anchor='nw')
    tk.Label(otherFrame, text="请输入 作者姓名", font=('隶书', 16)).pack()
    e2 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e2, font=('宋体', 14)).pack(padx=5, pady=10)
    tk.Label(otherFrame, text="请输入 至少著作数量", font=('隶书', 16)).pack()
    e3 = tk.StringVar()
    tk.Entry(otherFrame, show='', textvariable=e3, font=('宋体', 14)).pack(padx=5, pady=10)
    click = lambda: find_book(otherFrame, e2, e3)
    tk.Button(otherFrame, text='查询', command=click, font=('隶书', 16)).pack(side='bottom')
```

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

```
In [110]:
def find_book(oF, e2, e3):
    aut_name = e2.get()
    works_num = e3.get()
    tree = ttk.Treeview(oF)
    title = ["作者姓名", "著作数量"]
    tree["columns"] = ("作者姓名", "著作数量")
    for i in title:
        tree.column(i, anchor="center")
        tree.heading(i, text=i)
    temp = 'select * from author_find '
    if aut_name != '' and works_num != '':
        temp = temp + "where author_name = '" + str(aut_name) + "' and countworks >= '" + str(works_num) + "';"
    elif aut_name != '':
        temp = temp + "where author_name = '" + str(aut_name) + "';"
    elif works_num != '':
        temp = temp + "where countworks >= '" + str(works_num) + "';"
    else:
        tk.Label(oF, text='查 询 失 败', font=('楷书', 22), width=30, height=2).pack()
        return
    cursor.execute(temp)
    result = cursor.fetchall()
    if len(result) == 0:
        tk.Label(oF, text='查 询 失 败', font=('楷书', 22), width=30, height=2).pack()
        return
    for i in range(len(result)):
        tree.insert("", i, values=result[i])
    tree['show'] = 'headings'
    tree.pack(anchor="center", padx=5, pady=10)
```

程序演示（4 分）

查询成功：

不存在满足要求的人：

	 查询某作者著作数量 返回 请输入 作者姓名 mike 请输入 至少著作数量 2 查 询 失 败 查询
备注	