1. 项目信息（10分）

学号：2013287

姓名：王浩

专业：计算机科学与技术

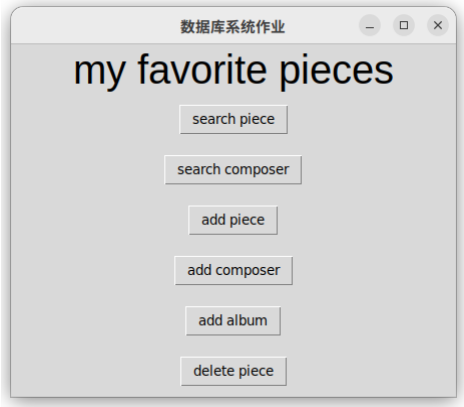
项目名称：my favorite pieces

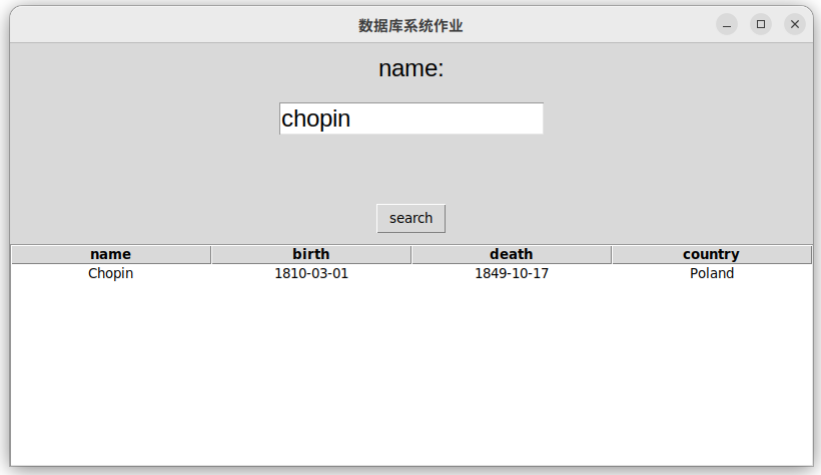
必备环境：Ubuntu22.04, python3 (使用tkinter, pymysql库), datagrip, MySQL Community Server

系统主要功能简介（4分）

一个简单的个人音乐收藏数据库，有查询音乐、查询作曲家、添加音乐、删除音乐等功能。

系统主要页面截图（6分）





2. 系统配置（10分）

配置步骤（2分）

DBMS后台数据库使用Mysql，高级语言使用python 3.10

连接串分析（6分）

参数	功能	取值
host	Mysql连接的主机	localhost
user	Mysql用户名	root
password	Mysql密码	password
database	Mysql中的数据库	music
port	连接端口	3306
autocommit	是否自动提交	True

连接串代码（2分）

```
db = pymysql.connect(host = "localhost", user = "root", password = "password", database =
"music", port = 3306, autocommit = True)
```

3. 数据库设计（14分）

数据表（10分）

1. piece(Opus, name, composer, album)

   主键：Opus, composer

   外键：composer(composer.name), album(album.name)

2. composer(name, birth_time, death_time, country)

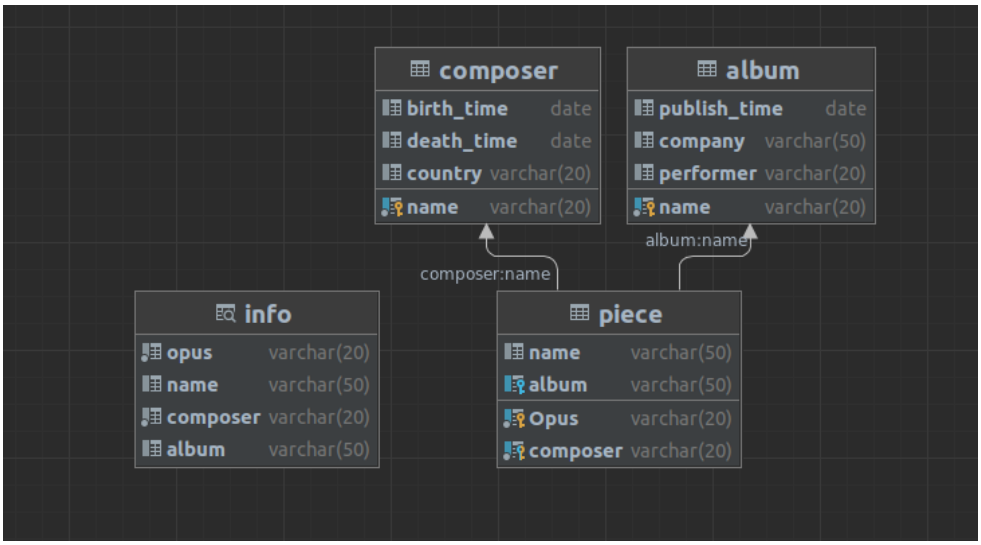
   主键：name

3. album(name, publish_time, company, performer)

   主键：name

关系图（4分）

由datagrip创建：



4. 含有事物应用的删除操作（13分）

功能描述（1分）

查找数据库中的某一首曲子，然后选择删除或取消

涉及的表（2分）

piece, composer

表连接涉及字段（1分）

where opus like '%" + str1 + "%' and composer like '%" + str2 + "%'

删除条件字段描述（1分）

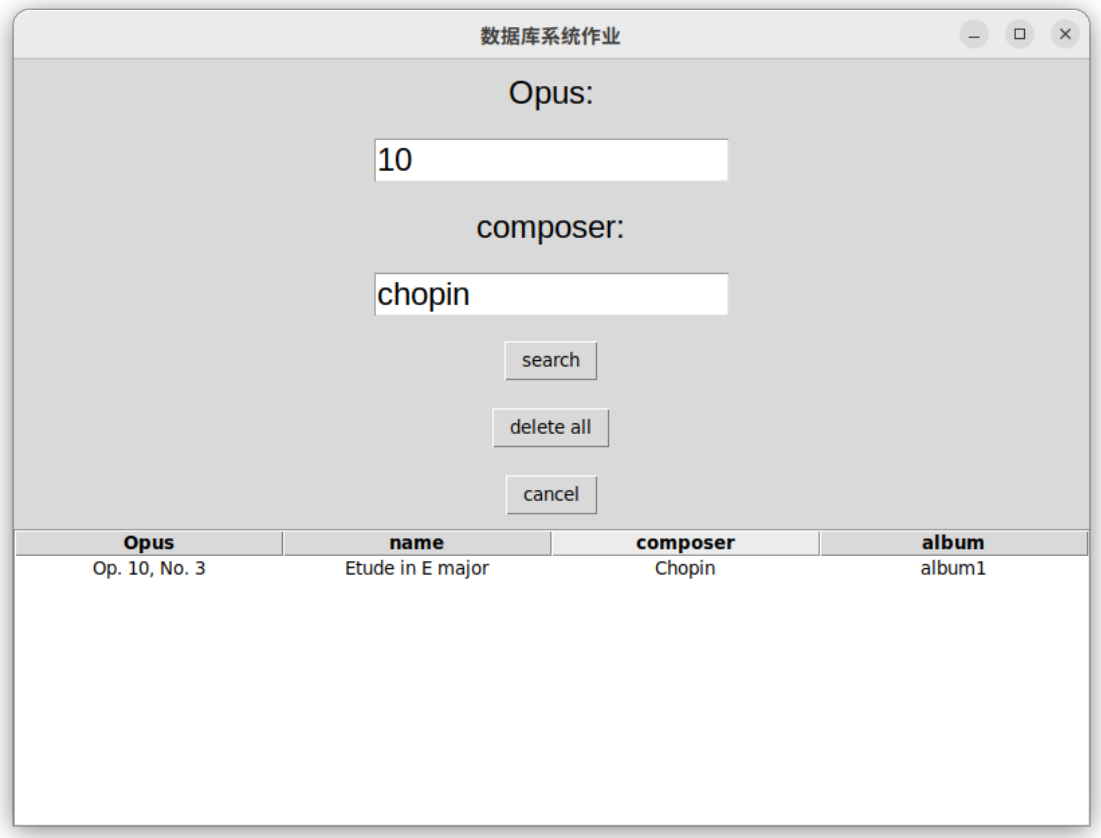
给两个参数str1和str2（其中一个可以为空），如果数据库中有该参数的模糊匹配，则找到对应曲子

代码（4分）

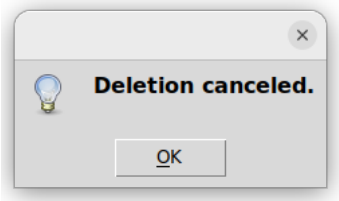
```
def delete_piece():
    frm = Toplevel()
    frm.rowconfigure([0, 1, 2, 3, 4, 5, 6, 7], minsize=50)
    frm.columnconfigure(0, minsize=50)

    Label(master=frm, text='Opus:', font=('Arial', 18)).grid(row=0, column=0)
    e1 = Entry(master=frm, font=('Arial', 18), width=20)
    e1.grid(row=1, column=0)
    Label(master=frm, text='composer:', font=('Arial', 18)).grid(row=2, column=0)
    e2 = Entry(master=frm, font=('Arial', 18), width=20)
    e2.grid(row=3, column=0)

    click = lambda:exec_delete_piece(frm, e1.get(), e2.get())
    Button(master=frm, text='search', command=click).grid(row=4, column=0)

def exec_delete_piece(frm, opus, composer):
    tree = ttk.Treeview(master=frm)
    ls = ['Opus', 'name', 'composer', 'album']
    tree['columns'] = ('Opus', 'name', 'composer', 'album')
    for i in ls:
```

```
        tree.column(i, anchor='center')
        tree.heading(i, text=i)

    show_sql = "select * from piece where opus like '%" + opus + "%' and composer like '%"
+ composer + "%';"
    cursor.execute(show_sql)
    result = cursor.fetchall()
    for i in range(len(result)):
        tree.insert("", i, values=result[i])
    tree['show'] = 'headings'
    tree.grid(row=7, column=0)

    del_sql = "delete from piece where opus like '%" + opus + "%' and composer like '%" +
composer + "%';"
    click = lambda:confirm(frm, del_sql)
    Button(frm, text='delete all', command=click).grid(row=5, column=0)
    click = lambda:cancel(frm, del_sql)
    Button(frm, text='cancel', command=click).grid(row=6, column=0)

def confirm(frm, del_sql):
    cursor.execute("START TRANSACTION")
    cursor.execute(del_sql)
    cursor.execute("COMMIT")
    messagebox.showinfo(message='Delete successfully.')

def cancel(frm, del_sql):
    cursor.execute("START TRANSACTION")
    cursor.execute(del_sql)
    cursor.execute("ROLLBACK")
    messagebox.showinfo(message='Deletion canceled.')
```

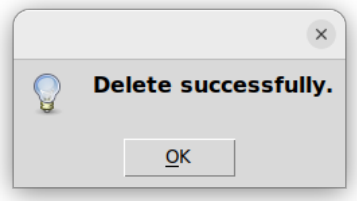
程序演示（4分）



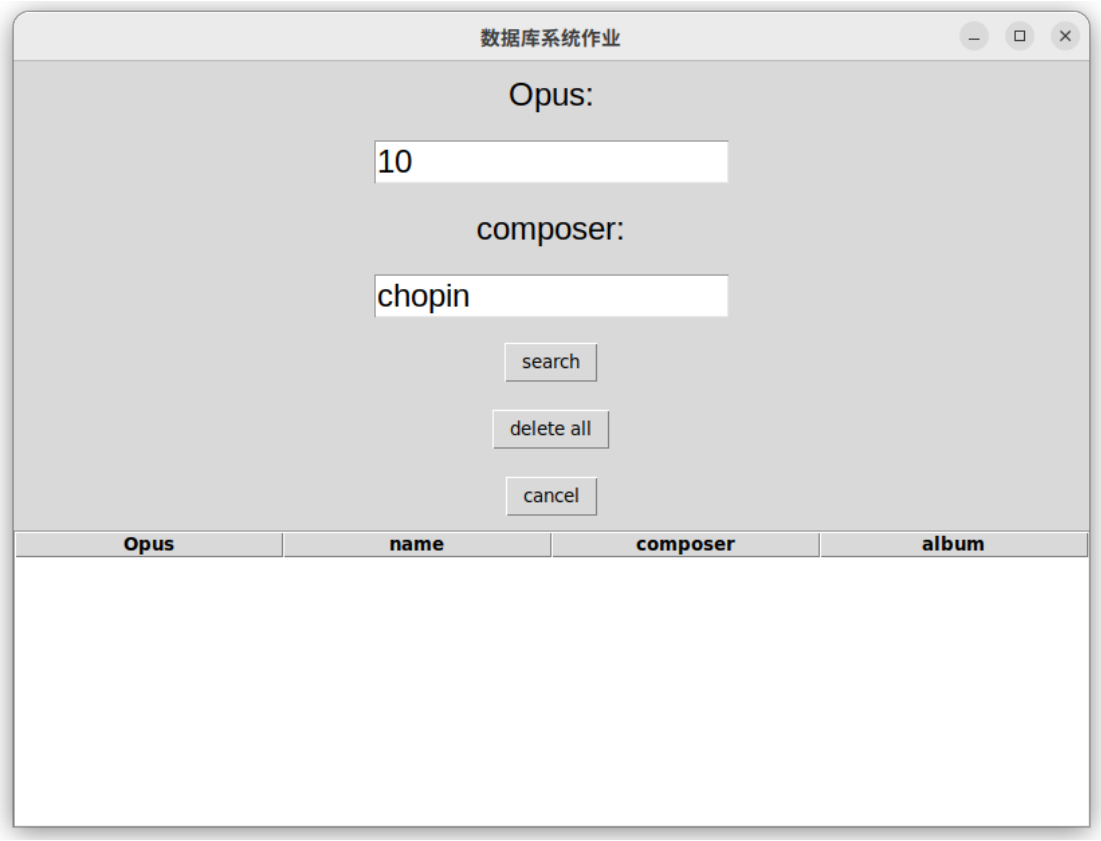
点search后找到对应曲子，如果点cancel，就会在mysql中执行ROLLBACK：



如果点delete all，就会删除所有选中的曲子，并COMMIT：



再搜索就搜索不到了：



5. 触发器下的添加操作（20分）

功能描述（1分）

添加曲子，如果piece.composer不在composer表中，则执行触发器

触发器描述（2分）

如果piece.composer不在composer表中，则执行触发器

涉及的表（1分）

piece, composer

输入数据（2分）

- opus：作品号
- name：作品名
- composer：作曲家，需要在数据库composer表中有
- album：作品所属专辑

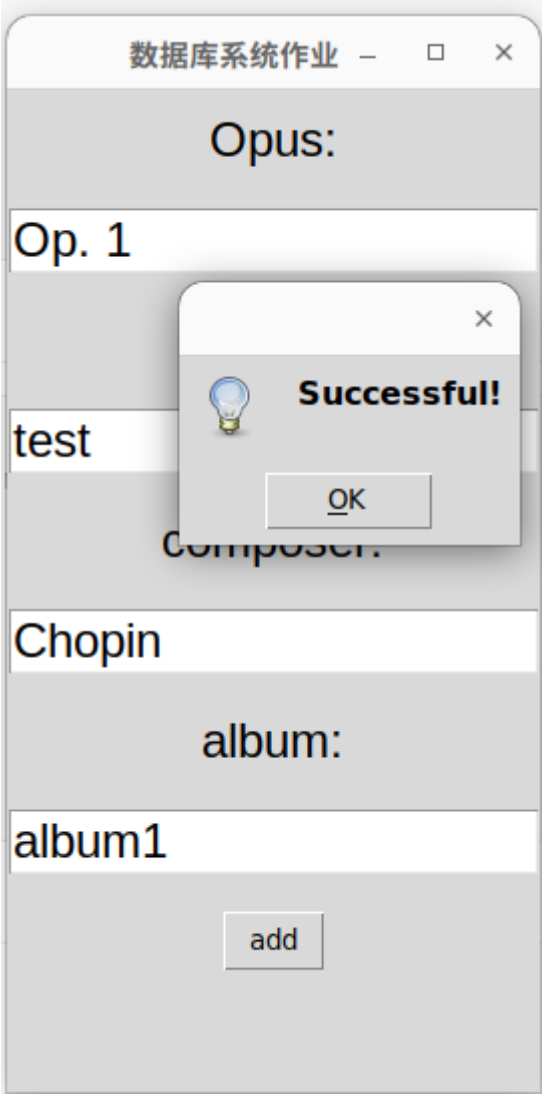
插入操作源码（3分）

```
def add_piece():
    frm = Toplevel()
    frm.rowconfigure([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], minsize=50)
    frm.columnconfigure(0, minsize=50)

    Label(master=frm, text='Opus:', font=('Arial', 18)).grid(row=0, column=0)
    e1 = Entry(master=frm, font=('Arial', 18), width=20)
    e1.grid(row=1, column=0)
    Label(master=frm, text='name:', font=('Arial', 18)).grid(row=2, column=0)
    e2 = Entry(master=frm, font=('Arial', 18), width=20)
    e2.grid(row=3, column=0)
    Label(master=frm, text='composer:', font=('Arial', 18)).grid(row=4, column=0)
    e3 = Entry(master=frm, font=('Arial', 18), width=20)
    e3.grid(row=5, column=0)
    Label(master=frm, text='album:', font=('Arial', 18)).grid(row=6, column=0)
    e4 = Entry(master=frm, font=('Arial', 18), width=20)
    e4.grid(row=7, column=0)

    click = lambda:exec_add_piece(frm, e1.get(), e2.get(), e3.get(), e4.get())
    Button(master=frm, text='add', command=click).grid(row=8, column=0)

def exec_add_piece(frm, opus, name, composer, album):
    sql = "insert into piece values('" + opus + "', '" + name + "', '" + composer + "', '"
+ album + "');"
    try:
        cursor.execute(sql)
        messagebox.showinfo(message="Successful!")
    except Exception as m:
        messagebox.showerror("error", m.args)
```

触发器源码（3分）

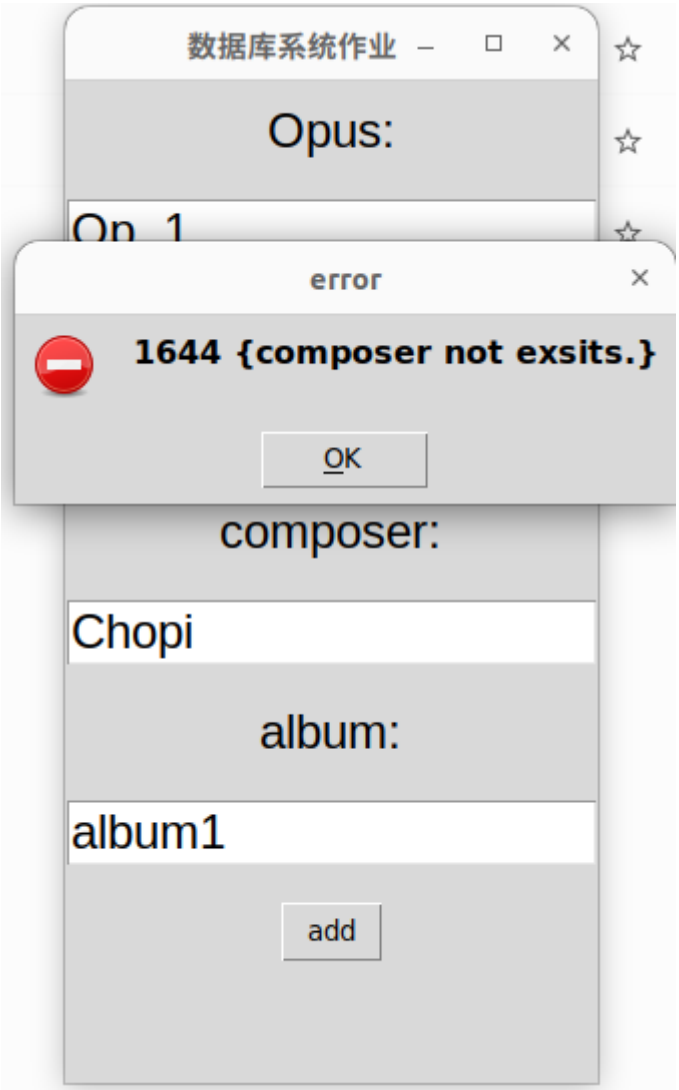
```
DELIMITER ;;
!50003 CREATE DEFINER=`root`@`localhost`RIGGER `check_insert_piece` BEFORE INSERT ON
`piece` FOR EACH ROW begin
        if new.composer not in (select name from composer)  then
            signal SQLSTATE '45000'
            set message_text = "composer not exsits.";
        end if;
    end */;;
DELIMITER ;
```

程序演示，不违背触发器（4分）



数据库中有Chopin这位作曲家，所以成功添加。

程序演示，违背触发器（4分）



数据库中没有Chopi这位作曲家，所以触发器抛出了`composer not exsits.`信息。

6. 存储过程控制下的更新操作（18分）

功能描述（1分）

更改piece中某首曲子的所属专辑

存储过程功能描述（1分）

用procedure更改piece.album，要求新的piece.album在album表中

涉及的关系表（2分）

piece, album

表连接涉及字段（1分）

album_name not in (select name from album)

更改字段（2分）

piece.album：要求必须在表album中存在

更新代码（3分）

```python
def exec_search_piece(frm, str1, str2):
    # 得到了查询结果result...

    # update album
    if (len(result) == 1 and str1 != ''):
        Label(master=frm, text='new album:', font=('Arial', 18)).grid(row=6, column=0)
        e3 = Entry(master=frm, font=('Arial', 18), width=20)
        e3.grid(row=7, column=0)
        click = lambda:exec_update(frm, str1, e3.get())
        Button(master=frm, text="update", command=click).grid(row=8, column=0)
```

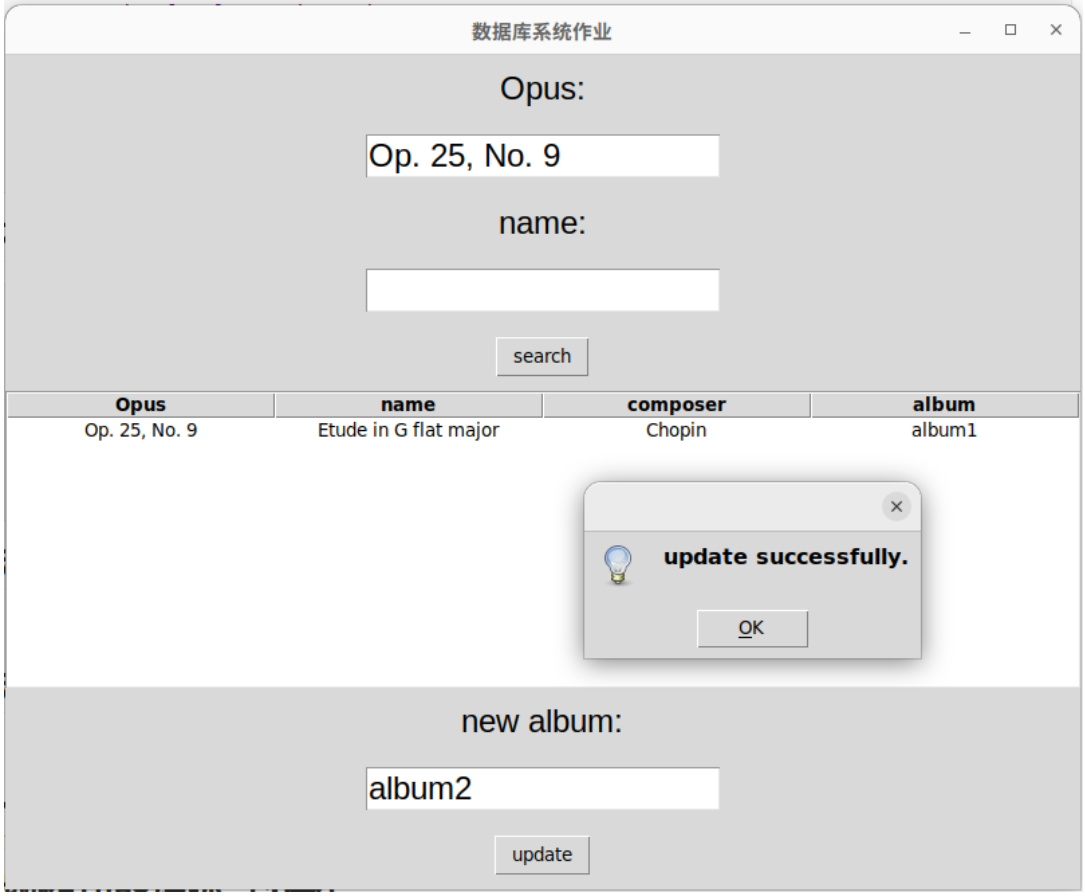
创建存储过程源码（3分）

```python
create procedure update_album(in opus_name varchar(20), in album_name varchar(50))
begin
    if album_name not in (select name from album) then
        signal sqlstate '45000'
        set message_text = 'album not exists.';
    else
        update piece set album = album_name where Opus like CONCAT('%', opus_name, '%');
    end if;
end;
```

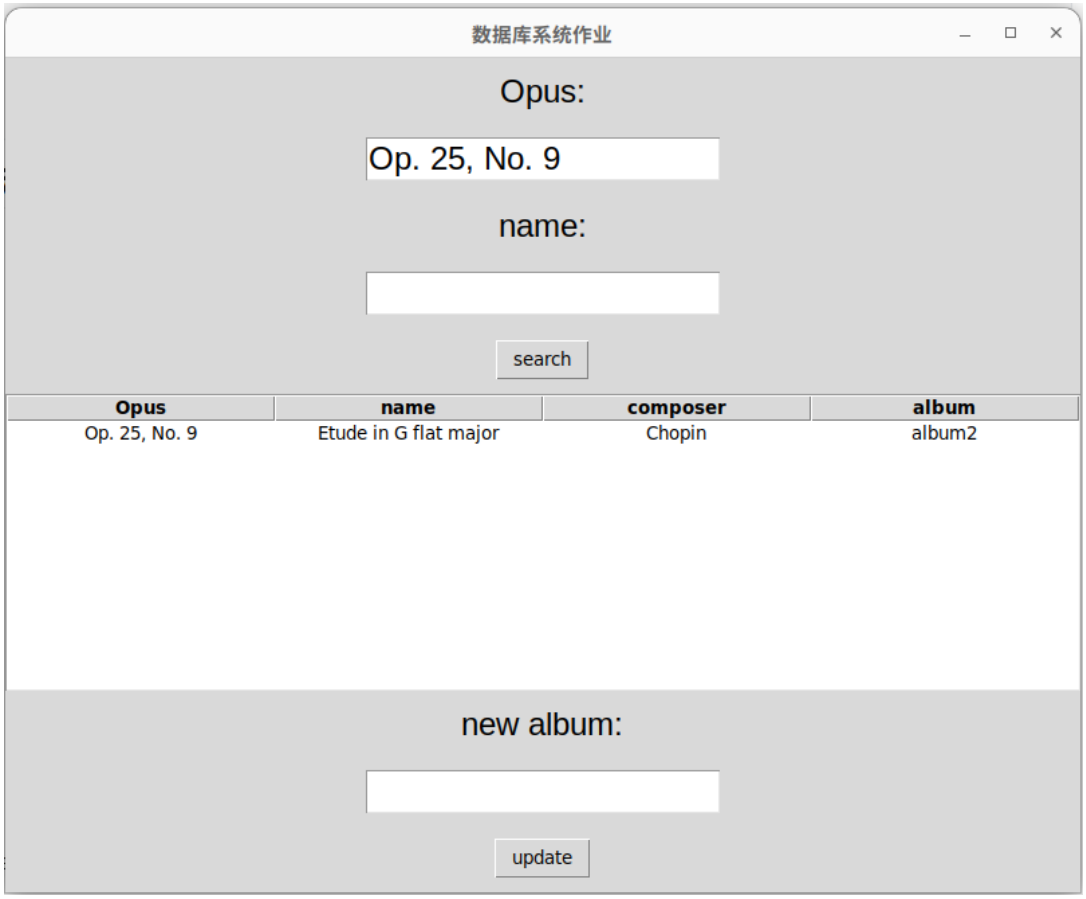
存储过程执行源码（1分）

```python
def exec_update(frm, opus, new_album):
    sql = "call update_album('" + opus + "', '" + new_album + "');"
    try:
        cursor.execute(sql)
        messagebox.showinfo(message='update successfully.')
    except Exception as m:
        messagebox.showerror('error', m.args)
```

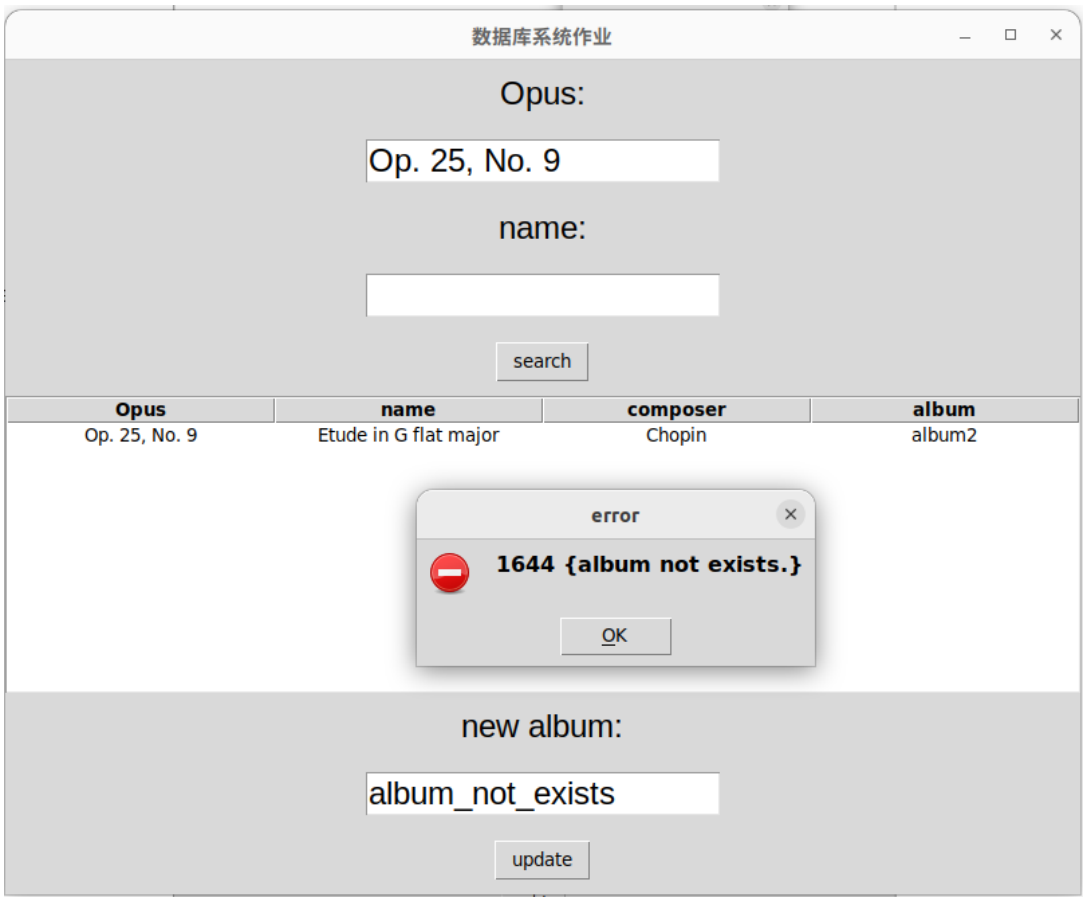
程序演示，不违背存储过程（2分）



album表中有album2这张专辑，因此更新成功，Op. 25, No. 9现在属于album2：



程序演示，违背存储过程（2分）



选择一张不存在的专辑就会得到error。

7. 含有视图的查询操作（15分）

操作功能描述（1分）

根据opus, name模糊匹配找到一首曲子

视图功能描述（1分）

综合piece和composer中的opus, piece.name, composer, album, country属性到一张视图里

涉及的关系表（2分）

piece, composer

表连接字段（1分）

where opus like '%" + str1 + "%' and name like '%" + str2 + "%'

创建视图代码（3分）

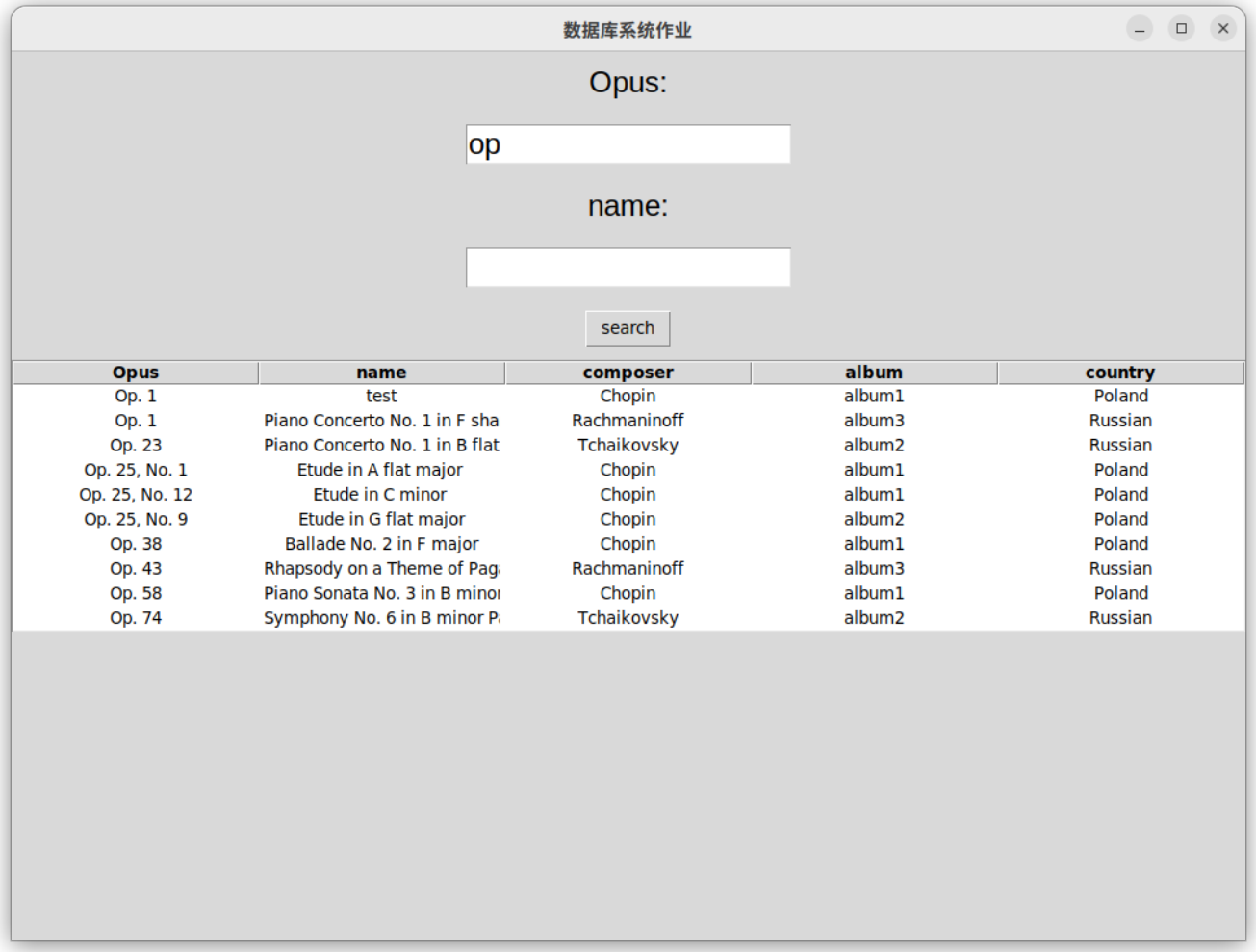
```
CREATE ALGORITHM=UNDEFINED
DEFINER=`root`@`localhost` SQL SECURITY DEFINER
VIEW `info` AS select `piece`.`Opus` AS `opus`,`piece`.`name` AS `name`,`piece`.`composer`
AS `composer`,`piece`.`album` AS `album`,`composer`.`country` AS `country` from (`piece`
join `composer`) where (`piece`.`composer` = `composer`.`name`);
```

查询代码（3分）

```python
def search_piece():
    frm = Toplevel()
    frm.rowconfigure([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], minsize=50)
    frm.columnconfigure(0, minsize=50)
    Label(master=frm, text='Opus:', font=('Arial', 18)).grid(row=0, column=0)
    e1 = Entry(master=frm, font=('Arial', 18), width=20)
    e1.grid(row=1, column=0)
    Label(master=frm, text='name:', font=('Arial', 18)).grid(row=2, column=0)
    e2 = Entry(master=frm, font=('Arial', 18), width=20)
    e2.grid(row=3, column=0)
    click = lambda:exec_search_piece(frm, e1.get(), e2.get())
    Button(master=frm, text='search', command=click).grid(row=4, column=0)

def exec_search_piece(frm, str1, str2):
    tree = ttk.Treeview(master=frm)
    ls = ['Opus', 'name', 'composer', 'album', 'country']
    tree['columns'] = ('Opus', 'name', 'composer', 'album', 'country')
    for i in ls:
        tree.column(i, anchor='center')
        tree.heading(i, text=i)
    sql = "select * from info "
    if str1 != '' and str2 != '':
        sql = sql + "where opus like '%" + str1 + "%' and name like '%" + str2 + "%';"
    elif str1 != '':
        sql = sql + "where opus like '%" + str1 + "%';"
    elif str2 != '':
        sql = sql + "where name like '%" + str2 + "%';"
    else:
        Label(master=frm, text='input valid', font=('Arial', 18)).grid(row=5, column=0)
        return
    cursor.execute(sql)
    result = cursor.fetchall()
    for i in range(len(result)):
        tree.insert('', i, values=result[i])
    tree['show'] = 'headings'
    tree.grid(row=5, column=0)
```

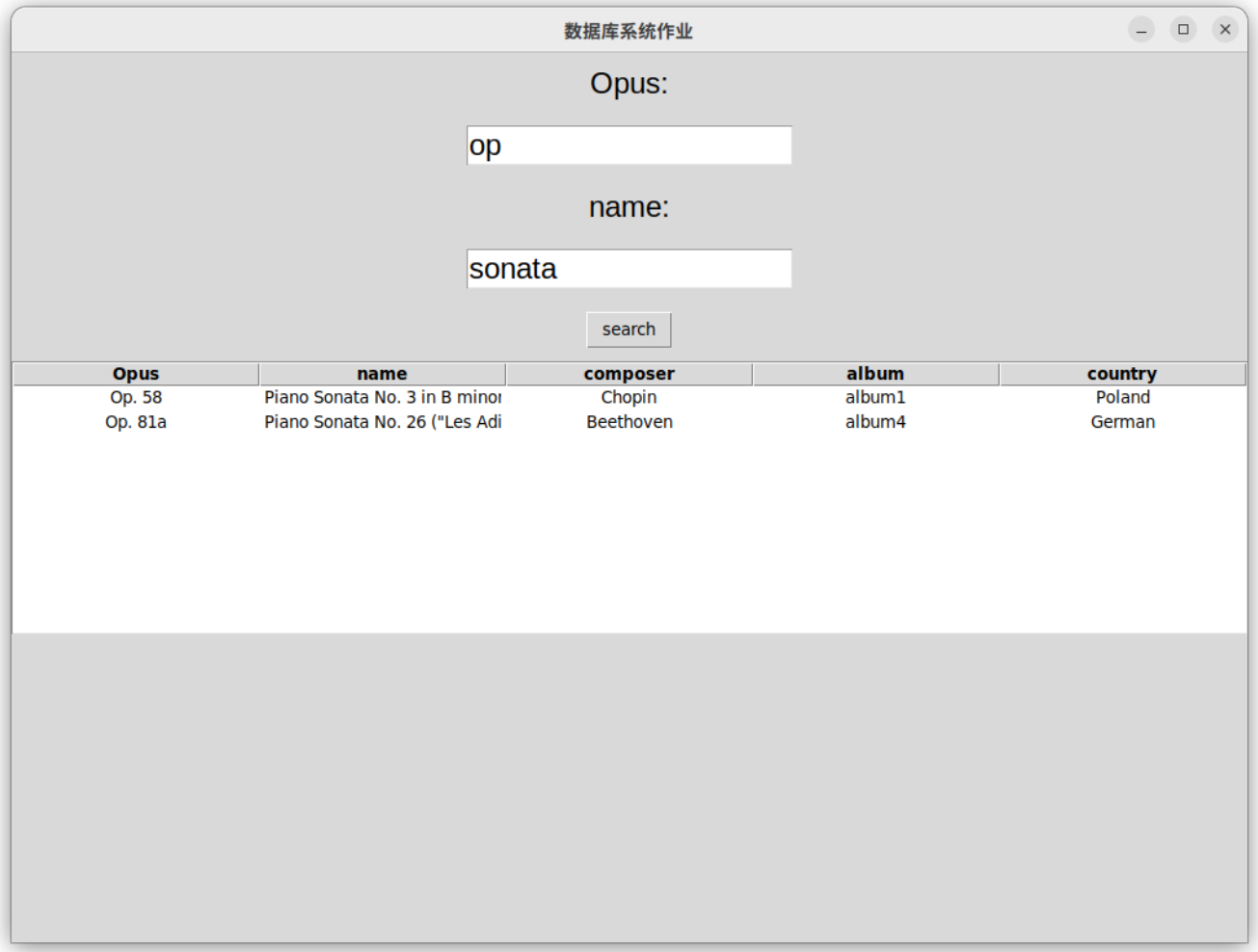
程序演示（4分）

查找所有曲子：



Opus	name	composer	album	country
Op. 1	test	Chopin	album1	Poland
Op. 1	Piano Concerto No. 1 in F sha	Rachmaninoff	album3	Russian
Op. 23	Piano Concerto No. 1 in B flat	Tchaikovsky	album2	Russian
Op. 25, No. 1	Etude in A flat major	Chopin	album1	Poland
Op. 25, No. 12	Etude in C minor	Chopin	album1	Poland
Op. 25, No. 9	Etude in G flat major	Chopin	album2	Poland
Op. 38	Ballade No. 2 in F major	Chopin	album1	Poland
Op. 43	Rhapsody on a Theme of Pag	Rachmaninoff	album3	Russian
Op. 58	Piano Sonata No. 3 in B minor	Chopin	album1	Poland
Op. 74	Symphony No. 6 in B minor P	Tchaikovsky	album2	Russian

查找奏鸣曲（Sonata）：



Opus	name	composer	album	country
Op. 58	Piano Sonata No. 3 in B minor	Chopin	album1	Poland
Op. 81a	Piano Sonata No. 26 ("Les Adi	Beethoven	album4	German

其中的country字段就是通过连接了表piece和表composer的视图info显示的。