
哈尔滨工业大学

<<数据库系统>>

实验报告二

(2023 年度春季学期)

姓名:	
学号:	
学院:	
教师:	

实验二

一、实验目的

在熟练掌握 MySQL 基本命令、SQL 语言以及用 C 语言编写 MySQL 操作程序的基础上，学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计。

二、实验环境

Windows 11 操作系统，MySQL Workbench 8.0 CE。

三、实验过程及结果

1. 数据库设计

设计一个关于音乐的数据库，涉及如下实体：用户、专辑、歌曲、艺术家、MV、唱片公司、实体商品、演出活动。语义如下：

用户可以收藏多个专辑，一个专辑可以被多个用户收藏；

一张专辑可收录多首歌曲，一首歌曲只能属于一张专辑；

歌曲和 MV 是一一对应的；

一张专辑可以发行多种实体商品，一种实体商品只能附属于一张专辑；

一个唱片公司名下可发行多张专辑，一张专辑只能在一个唱片公司名下发行；

一个艺术家可发行多张专辑，一张专辑只能由一位艺术家发行；

一个艺术家可参与多场演出活动，一个演出活动也可包含多位艺术家。

由以上实体及语义，绘制出的 E-R 图如图 1。设计关系模式如下：

USERS(uid, uname, pwd)

ALBUM(alid, atitle, arid, genre, lname, adate)

COLLECTION(uid, alid)

ARTIST(arid, aname, atype)

SONG(sid, stitle, duration, arid, alid, tno)

PRODUCT(alid, pname, pprice)

LABEL(lname, year)

MV(sid, mtitle, mdate)

EVENT(eid, etitle, eprice, eaddr)

PERFORMANCE(arid, datetime, eid)

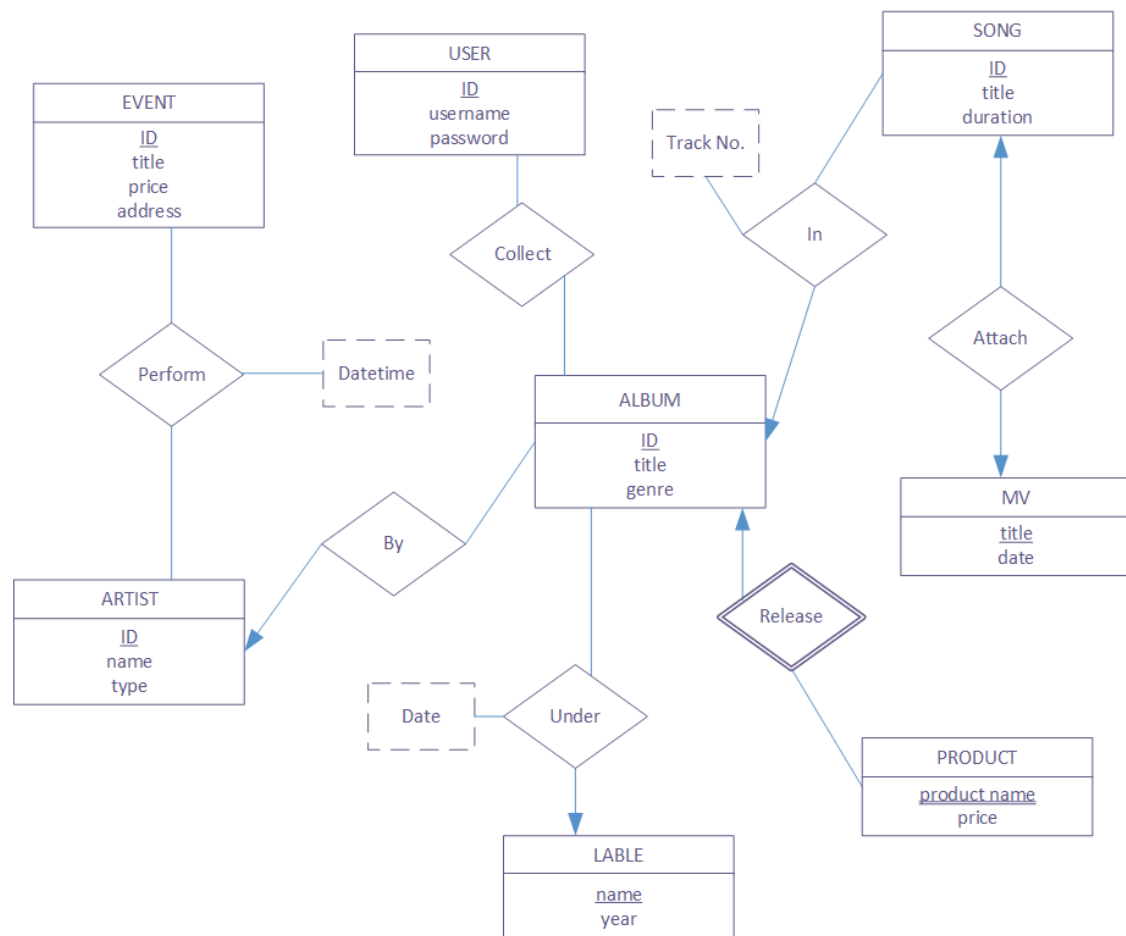


图 1 E-R 图

2.数据库实现

创建并使用数据库 music:

```
CREATE DATABASE music;  
USE music;
```

创建对应关系表:

```
CREATE TABLE users (  
    uid CHAR(3),  
    uname VARCHAR(20),  
    pwd VARCHAR(50)  
);  
  
CREATE TABLE album (  
    alid CHAR(3),  
    atitle VARCHAR(50),  
    arid CHAR(3),  
    genre VARCHAR(20),  
    lname VARCHAR(50),  
    adate DATE  
);  
  
CREATE TABLE collection (  
    uid CHAR(3),  
    alid CHAR(3),  
    date DATE  
);
```

```
        uid CHAR(3),
        alid CHAR(3)
    );
CREATE TABLE artist (
    arid CHAR(3),
    aname VARCHAR(20),
    atype CHAR(1)
);
CREATE TABLE song (
    sid CHAR(3),
    stitle VARCHAR(50),
    duration INT,
    alid CHAR(3),
    tno INT
);
CREATE TABLE product (
    alid CHAR(3),
    pname VARCHAR(50),
    pprice INT
);
CREATE TABLE label (
    lname VARCHAR(50),
    year INT
);
CREATE TABLE mv (
    sid CHAR(3),
    mtitle VARCHAR(50),
    mdate DATE
);
CREATE TABLE event (
    eid CHAR(3),
    etitle VARCHAR(50),
    eprice INT,
    eaddr VARCHAR(50)
);
CREATE TABLE performance (
    arid CHAR(3),
    dt DATETIME,
    eid CHAR(3)
);
```

添加主键、非空、重复约束，以表 album 为例：

```
ALTER TABLE `music`.`album`
CHANGE COLUMN `alid` `alid` CHAR(3) NOT NULL ,
```

```
ADD PRIMARY KEY (`alid`);
```

```
ALTER TABLE `music`.`album`  
CHANGE COLUMN `atitle` `atitle` VARCHAR(50) NOT NULL ;
```

```
ALTER TABLE `music`.`album`  
ADD UNIQUE INDEX `alid_UNIQUE` (`alid` ASC) VISIBLE;
```

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
alid	CHAR(3)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
atitle	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
arid	CHAR(3)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
genre	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
lname	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
adate	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

图 2 添加约束后效果

添加外键：

```
ALTER TABLE `music`.`album`  
ADD INDEX `album-artist_idx` (`arid` ASC) VISIBLE;  
;  
ALTER TABLE `music`.`album`  
ADD CONSTRAINT `album-artist`  
FOREIGN KEY (`arid`)  
REFERENCES `music`.`artist` (`arid`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION;
```

Foreign Key Name	Referenced Table	Column	Referenced Column	Foreign Key Options
album-artist	`music`.`artist`	<input checked="" type="checkbox"/> arid	arid	On Update: RESTRICT On Delete: RESTRICT <input type="checkbox"/> Skip in SQL generation

图 3 添加外键后效果

添加索引，以在 album 中查询专辑标题为例：

```
ALTER TABLE `music`.`album`  
ADD INDEX `album-title` (`atitle` ASC) VISIBLE;
```

添加视图，以 songinfo（歌曲、艺术家、所属专辑）为例：

```
CREATE VIEW songinfo (Song , Artist , Album) AS  
SELECT  
    stitle, aname, atitle  
FROM  
    song,  
    artist,  
    album
```

WHERE

song.arid = artist.arid

AND song.alid = album.alid;

用命令 SELECT * FROM music.songinfo;查看视图，结果如下。

	Song	Artist	Album
▶	Hunter	Bjork	Homogenic
	Joga	Bjork	Homogenic
	Unravel	Bjork	Homogenic
	Bachelorette	Bjork	Homogenic
	All Neon Like	Bjork	Homogenic
	5 Years	Bjork	Homogenic
	Immature	Bjork	Homogenic
	Alarm Call	Bjork	Homogenic
	Pluto	Bjork	Homogenic

图 4 查看视图结果

3.数据库使用

3.1 正常单表查询

使用命令

SELECT

stitle

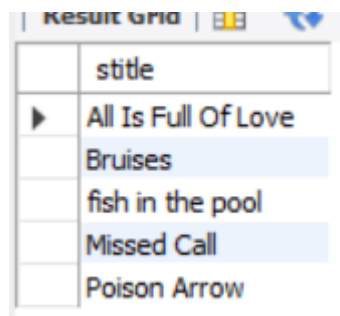
FROM

song

WHERE

stitle LIKE '%is%';

进行对歌名的字符串匹配查找，类似于搜索引擎的功能，得到结果条目如图 5。



	stitle
▶	All Is Full Of Love
	Bruises
	fish in the pool
	Missed Call
	Poison Arrow

图 5 单表查询示例结果

若在指令前加 explain，显示结果如图 6，可见在表 song 中对非主键属性 stitle 建立的索引可以正常工作。

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	song	NULL	index	NULL	song-title	203	NULL	58	11.11	Using wher

图 6 explain 结果

3.2 正常插入条目

使用命令 insert into artist value ('005', 'A', 'M');向表 artist 中插入一条新的条目，可以正常插入，结果如图 7。

	arid	aname	atype
▶	001	Bjork	F
	002	FKA twigs	F
	003	Kelela	F
	004	yeule	G
	005	A	M

图 7 正常插入示例结果

3.3 插入重复条目

执行代码 `insert into artist value ('001', 'A', 'M');`，因 `artist` 中已有 `aid` 为 001 的条目，会出现错误 “Error Code: 1062. Duplicate entry '001' for key 'artist.PRIMARY'”，如图 8。

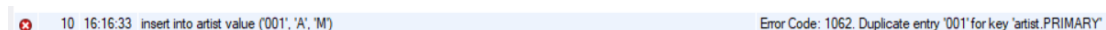


图 8 重复插入示例报错

3.4 插入空值条目

执行代码 `insert into users value ('004', NULL, 2022-01-01);` 新增一条用户信息，但其用户名为空，而因用户名有非空约束，会出现错误 “Error Code: 1048. Column 'uname' cannot be null”，如图 9。



图 9 空值插入示例报错

3.5 插入条目外键约束

执行代码 `insert into collection value ('001', '010');`，在 `collection` 表中新增一个条目，表示 `uid` 为 001 的用户欲收藏 `alid` 为 010 的专辑，但因外键 `alid` 在表 `album` 中不存在，会报错 “Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (‘music’.‘collection’, CONSTRAINT ‘collection-akbum’ FOREIGN KEY (‘alid’) REFERENCES ‘album’ (‘alid’))”，如图 10。



图 10 插入条目外键约束示例报错

3.6 删除条目外键约束

执行代码 `delete from artist where arid=001;`，企图将 `arid` 为 001 的艺术家删除，但因外键约束的存在，会报错 “Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (‘music’.‘album’, CONSTRAINT ‘album-artist’ FOREIGN KEY (‘arid’) REFERENCES ‘artist’ (‘arid’))”，如图 11。

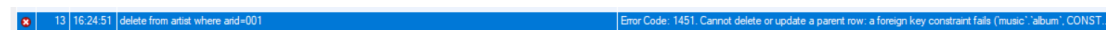


图 11 删除条目外键约束示例报错

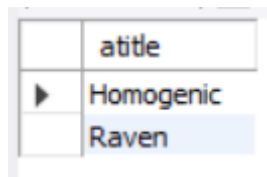
3.7 连接查询

查询用户名为 “WYX” 的所有用户收藏的所有专辑的标题，SQL 命令为：

```
SELECT
    atitle
FROM
    users,
    album,
    collection
WHERE
```

```
users.uid = collection.uid
AND collection.alid = album.alid
AND users.uname = 'WYX';
```

所得结果如图 12。



	atitle
▶	Homogenic
	Raven

图 12 连接查询示例结果

3.8 嵌套查询

查询 arid 为 004 的艺术家所发行的所有专辑的所有实体商品信息：

```
SELECT
    *
FROM
    product
WHERE
    alid IN (SELECT
        alid
        FROM
            album
        WHERE
            arid = 004);
```

结果如图 13。



	alid	pname	pprice
▶	004	cd	15
	004	ink-vinyl	30
	005	red-vinyl	35

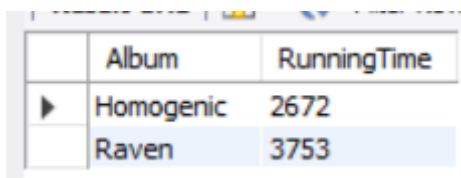
图 13 嵌套查询示例结果

3.9 分组查询

查询数据库中所有时长大于 43 分钟（数据库中单位为秒）的专辑及其时长：

```
SELECT
    atitle AS Album, SUM(duration) AS RunningTime
FROM
    album,
    song
WHERE
    album.alid = song.alid
GROUP BY song.alid
HAVING SUM(duration) > 2580;
```


所得结果如图 14。



	Album	RunningTime
▶	Homogenic	2672
	Raven	3753

图 14 分组查询示例结果

4. 前端程序

源码见“程序”文件夹下，main.py 是主要运行的程序，login.py 是登录界面的图形界面源码，因为时间原因，我只编写了这一个界面，其余功能是通过命令行交互实现的。程序运行的视频也已附上。

4.1 用户登录

可以通过用户 ID 和密码（在数据库 USERS 关系表中）进行登录。此处涉及到一次对数据库的查询，ID 和密码数据对应正确即可登录，否则报错，如图 15。

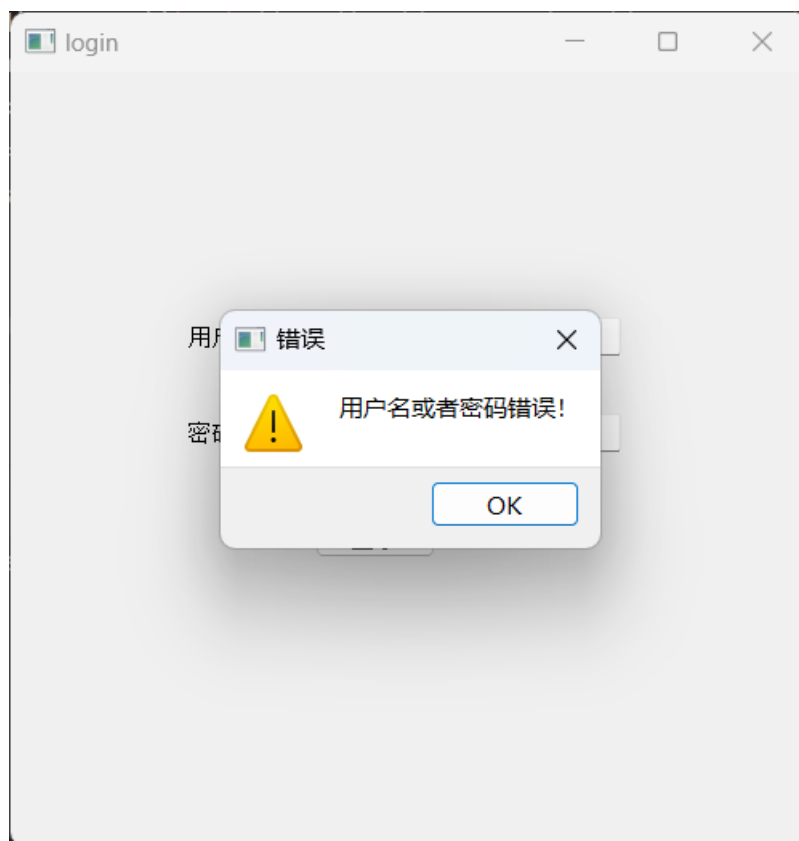


图 15 登陆失败效果

用户登录所涉及的最主要代码见图 16。登录成功后，其余功能可通过键盘输入指令来实现，接下来会详细介绍目前实现的若干功能。

```

def to_login(self):
    uid = self.ui.user_name.text()
    pwd = self.ui.password.text()
    sql = f"select pwd,uname from users where uid={uid};"
    cursor.execute(sql)
    result = cursor.fetchone()
    if result is None or result[0] != pwd:
        QMessageBox.warning(self, QWidget: '错误', p_str: '用户名或者密码错误! ')
    else:
        loginwindow.close()
        uname = result[1]
        QMessageBox.about(self, QWidget: '登录成功!', p_str: f'欢迎, {uname}! ')
        menu(uid, uname)

```

图 16 用户登录所涉及的最主要代码

4.2 查看已收藏的专辑列表

登录后，可通过“list”命令来查看当前所登录用户所收藏的专辑列表，如图 17。

```

欢迎, WYX。命令列表:
list: 查看您已收藏的专辑
search: 专辑名查找专辑
quit: 退出系统
help: 查看当前可操作指令
当前页面: 菜单。请选择操作: list
ID      专辑标题      艺术家
001 Homogenic Bjork
003      Raven Kelela
show + ID: 显示专辑曲目列表
remove + ID: 取消收藏专辑
return: 返回至菜单
help: 查看当前可操作指令
当前页面: 收藏架。请选择操作:

```

图 17 list 命令结果

List 命令涉及到的主要代码如图 18，其中涉及到了三个表的连接操作。

```

def printlist(uid):
    sql = f"select album.alid, atitle, aname from album, collection, \
    artist where artist.arid=album.arid and collection.alid=album.alid \
    and collection.uid={uid}"
    cursor.execute(sql)
    result = cursor.fetchall()
    df = pd.DataFrame(result, columns=["ID", "专辑标题", "艺术家"])
    print(df.to_string(index=False))

```

图 18 list 命令主要代码

4.3 按名字搜索专辑

可通过 search 命令进行专辑名字的关键字搜索,如图 19,搜索结果返回了所有专辑标题里包含“a”的专辑信息。

```
quit: 退出系统
help: 查看当前可操作指令
当前页面: 菜单。请选择操作: search
搜索关键词: a
  专辑ID      专辑标题      艺术家
0  002  MAGDALENE FKA twigs
1  003      Raven      Kelela
2  004  softscars      yeule
show + ID: 显示专辑曲目列表
collect + ID: 收藏专辑
return: 返回至菜单
help: 查看当前可操作指令
当前页面: 搜索结果。请选择操作: |
```

图 19 search 命令结果

Search 命令涉及的主要代码见图 20, 涉及到了两个表的连接和字符串运算。

```
def search(uid, kw):
    sql = f"select alid, atitle, aname from album, artist where \
    artist.arid=album.alid and album.atitle like '%{kw}%' "
    cursor.execute(sql)
    result = cursor.fetchall()
    df = pd.DataFrame(result, columns=["专辑ID", "专辑标题", "艺术家"])
    print(df)
    print('show + ID: 显示专辑曲目列表')
```

图 20 search 命令最主要代码

4.4 收藏专辑与取消收藏

在搜索结果页面,可以通过“collect+专辑 ID”命令来进行专辑的收藏,其实质是在 COLLECTION 中插入一条元组。成功收藏的结果如图 21。

```
help: 查看当前可操作指令
当前页面: 搜索结果。请选择操作: collect 002
收藏成功! 当前收藏列表:
  ID      专辑标题      艺术家
001 Homogenic      Bjork
002 MAGDALENE FKA twigs
003      Raven      Kelela
show + ID: 显示专辑曲目列表
remove + ID: 取消收藏专辑
return: 返回至菜单
help: 查看当前可操作指令
当前页面: 收藏架。请选择操作: |
```

图 21 成功收藏专辑结果

由于 COLLECTION 表中 uid、alid（用户 ID 和专辑 ID）有参照完整性约束，当用户试图收藏一张数据库中不存在的专辑时，会提示收藏失败，如图 22。

```
当前页面：搜索结果。请选择操作：collect 100
收藏失败！
当前页面：搜索结果。请选择操作：|
```

图 22 收藏失败效果

Collect 操作对应的主要代码见图 23。

```
def collect(uid, alid):
    sql = f"insert into collection values ('{uid}','{alid}')"
    try:
        cursor.execute(sql)
    except:
        print('收藏失败! ')
        return 0
    print('收藏成功! 当前收藏列表: ')
    printlist(uid)
    return 1
```

图 23 collect 命令主要代码

在收藏专辑列表界面，可通过“remove+专辑 ID”命令取消收藏某张专辑，如图 24。

```
当前页面：收藏架。请选择操作：remove 002
取消收藏成功！当前收藏列表：
ID      专辑标题      艺术家
001 Homogenic Bjork
003      Raven Kelela
```

图 24 remove 命令结果

Remove 命令的主要代码见图 25，实质是对 COLLECTION 表中元组的删除操作。

```
def remove(uid, alid):
    sql = f"delete from collection where uid={uid} and alid={alid}"
    try:
        cursor.execute(sql)
    except:
        print('取消收藏失败! ')
        return
    print('取消收藏成功! 当前收藏列表: ')
    printlist(uid)
```

图 25 remove 命令主要代码

4.5 显示专辑曲目列表

在搜索结果和收藏列表界面下，可通过“show+专辑 ID”进行专辑详细曲目列表的查看，如图 26。

```

当前页面：搜索结果。请选择操作： show 002
序号          歌曲标题  歌曲时长
1  thousand eyes 05:00
2  home with you 03:44
3      sad day 04:15
4  holy terrain 04:03
5 mary magdalene 05:21
6  fallen alien 03:58
7 mirrored heart 04:32
8      daybed 04:31
9  cellophane 03:24
-----
专辑总时长          38:48
当前页面：搜索结果。请选择操作：

```

图 26 show 命令结果

Show 命令涉及的主要代码见图 27，涉及了连接查询和聚集操作（用以计算专辑总时长）。

```

def show(alid):
    sql = f"select tno, stitle, duration from song where alid={alid} order by tno"
    cursor.execute(sql)
    result = cursor.fetchall()
    df = pd.DataFrame(result, columns=["序号", "歌曲标题", "秒"])
    df['歌曲时长'] = pd.to_datetime(df['秒'], unit='s').dt.strftime('%M:%S')
    df = df.drop(labels='秒', axis=1)
    print(df.to_string(index=False))
    print('-----')
    sql = f"select sum(duration) from song where alid={alid}"
    cursor.execute(sql)
    result = cursor.fetchone()
    total = result[0]
    total_M = total // 60
    total_S = total % 60
    print(f'专辑总时长          {total_M}:{total_S}')

```

图 27 show 命令主要代码

四、实验心得

实验初期对关系数据库的设计感到有些困难，以及初上手嵌入式 SQL 和图形界面库的使用非常陌生，感到有些难度。