



专业	班级	学号	姓名
物联网工程	物联网2202	223428010210	陈梓欣
物联网工程	物联网2202	223428010201	方艺安
物联网工程	物联网2202	223428010202	刘书男
物联网工程	物联网2101	213428010126	陆浩然

2024 年 5 月

## 摘 要

本系统旨在设计一个高效的电影院信息管理系统，以便管理电影院的管理人员、顾客、电影、放映信息和影票信息。该系统的设计对象涵盖了与电影院运营相关的所有关键角色和数据，目的是实现数据的有序管理和高效查询。

在研究方法上，系统采用了关系型数据库模型，使用 SQLite 数据库实现数据存储与管理。数据库设计包含五个主要表格：管理人员表、顾客表、电影表、放映信息表和影票信息表。每个表格均定义了必要的字段和约束条件，以确保数据的一致性和完整性。具体方法包括创建表结构、设置主键与外键关系以及定义数据类型和约束。

系统的研究结果展示了一个功能完备的数据库结构，能够存储和管理管理人员信息（包括用户名和密码哈希）、顾客信息（包括姓名、性别、年龄、联系电话和电子邮件）、电影信息（包括电影标题、导演、主要演员、类型、上映日期和时长）、放映信息（包括放映时间、影厅和状态）以及影票信息（包括影票价格、座位号和顾客标识）。

通过本研究，得出的重要结论是：该数据库设计能够有效支持电影院日常运营中的信息管理需求，具有较高的实用价值。系统通过清晰的表结构和字段定义，确保了数据的完整性和一致性，能够支持复杂查询和数据分析，为电影院管理提供了强有力的技术支持。

**关键词：**数据库设计；电影院管理；SQLite；SQL

## 目录

摘 要 .....	2
1 功能实现 .....	5
2 数据需求 .....	6
2.1 管理人员数据需求: .....	6
2.2 顾客数据需求: .....	6
2.3 电影数据需求: .....	7
2.4 放映信息表数据需求: .....	8
2.5 影票信息数据需求: .....	9
3 代码实现 .....	10
3.1 具体代码及注释 .....	10
3.2 具体实现 .....	41
4 总 结 .....	42
参考文献 .....	47

项目	陈梓欣	方艺安	刘书男	陆浩然
整体框架设计	√			
技术实现	√			
ER 图与数据流图绘制		√		
功能设计			√	
测试与分析				√
资料收集			√	
报告撰写		√		
主要工作	√			

# 1 功能实现

1. 电影院系统的管理和存储：电影院管理系统可以帮助管理人员对电影放映情况，电影票信息进行管理，也可以帮助电影院对电影票的售票情况进行汇总。

2. 用户权限管理：系统可以区分不同类型的用户，如管理人员和顾客，为不同用户人员提供相应的操作权限。管理员可以对电影票价，放映时间进行管理，顾客可以通过查询电影信息进行电影票的购买。

3. 排片和座位管理：影院可以有效安排电影的放映计划，根据电影类型，热度等因素进行排片安排，有助于最大化利用影厅资源，提高收益。跟踪和管理电影院座位的状态，包括座位的分配，预留，售卖等，系统存储座位信息可以避免座位冲突和重复售卖。

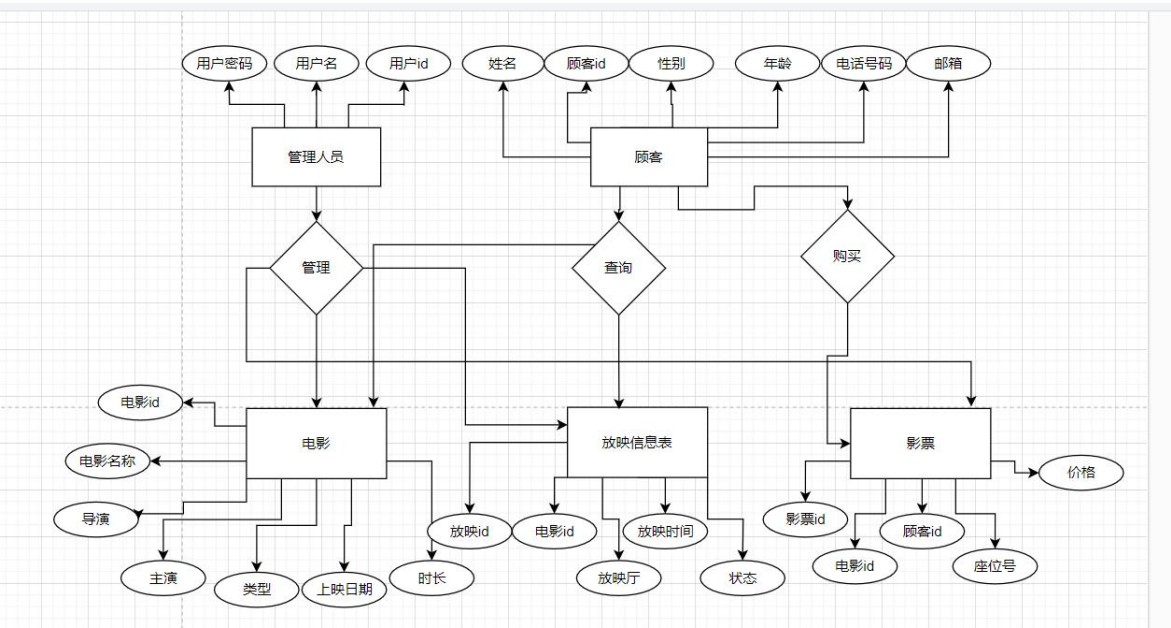
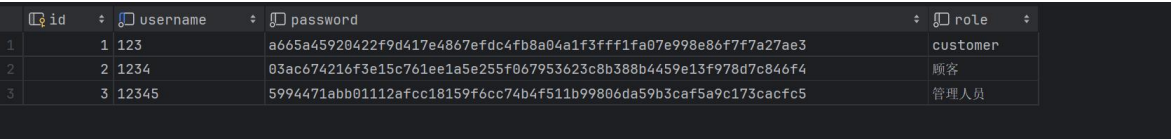


图 1

## 2 数据需求

### 2.1 管理人员数据需求:

- (1)用户密码: 用于管理人员的登录。密码应该以安全的方式存储, 例如加密或哈希处理。
- (2)用户名: 用于管理人员的身份识别和显示。
- (3)用户 id: 用于唯一标识每个管理人员的身份。

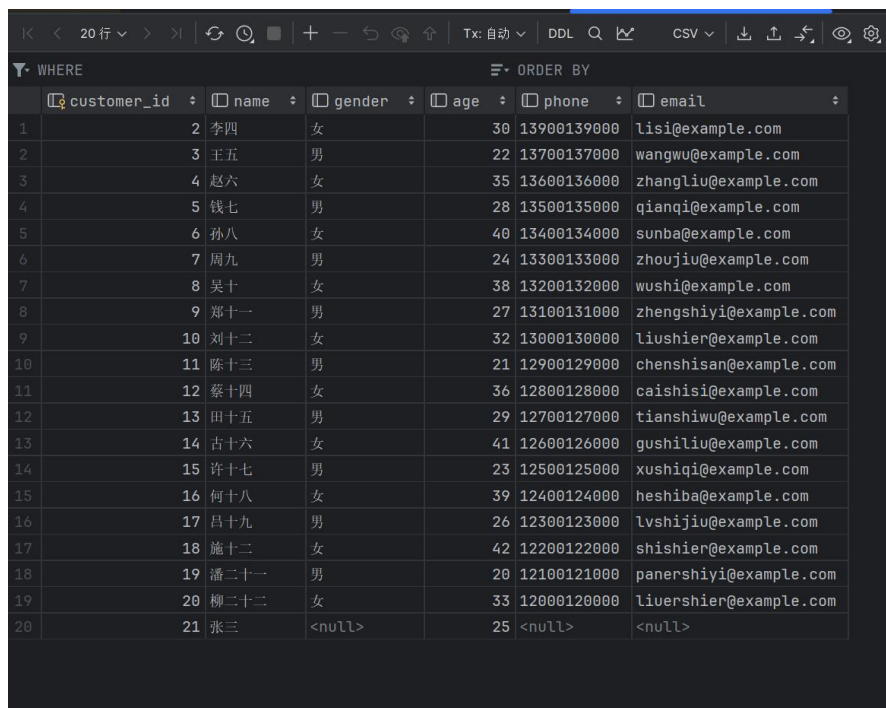


id	username	password	role
1	123	a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3	customer
2	1234	03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4	顾客
3	12345	5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5	管理人员

图 2 users 表

### 2.2 顾客数据需求:

- (1)姓名: 顾客的姓名或全名。
- (2)顾客 id: 用于唯一标识每个顾客的身份。
- (3)性别: 顾客的性别, 可以是男性、女性或其他。
- (4)年龄: 顾客的年龄信息。
- (5)电话号码: 顾客的联系电话号码。
- (6)邮箱: 顾客的电子邮件地址, 用于发送确认信息或促销活动。



	customer_id	name	gender	age	phone	email
1	2	李四	女	30	13900139000	lisi@example.com
2	3	王五	男	22	13700137000	wangwu@example.com
3	4	赵六	女	35	13600136000	zhangliu@example.com
4	5	钱七	男	28	13500135000	qianqi@example.com
5	6	孙八	女	40	13400134000	sunba@example.com
6	7	周九	男	24	13300133000	zhoujiu@example.com
7	8	吴十	女	38	13200132000	wushi@example.com
8	9	郑十一	男	27	13100131000	zhengshiyi@example.com
9	10	刘十二	女	32	13000130000	liushier@example.com
10	11	陈十三	男	21	12900129000	chenshisan@example.com
11	12	蔡十四	女	36	12800128000	caishisi@example.com
12	13	田十五	男	29	12700127000	tianshiwu@example.com
13	14	古十六	女	41	12600126000	gushiliu@example.com
14	15	许十七	男	23	12500125000	xushiqi@example.com
15	16	何十八	女	39	12400124000	heshiba@example.com
16	17	吕十九	男	26	12300123000	lvshijiu@example.com
17	18	施二十	女	42	12200122000	shishier@example.com
18	19	潘二十一	男	20	12100121000	panershiyi@example.com
19	20	柳二十二	女	33	12000120000	liuershier@example.com
20	21	张三	<null>	25	<null>	<null>

图 3 customers 表

## 2.3 电影数据需求:

- (1) 电影 id: 用于唯一标识每部电影的身份。
- (2) 电影名称: 电影的标题或名称。
- (3) 导演: 执导电影的导演姓名。
- (4) 主演: 主要演员或明星的姓名列表。
- (5) 类型: 电影的类型或类别, 如动作、喜剧、科幻等。
- (6) 上映日期: 电影的上映日期或发行日期。
- (7) 时长: 电影的播放时长, 通常以分钟为单位。

	WHERE		ORDER BY				
	movie_id	movie_name	director	cast	gengre	release_date	duration
1	1	哪吒之魔童降世	饺子	吕艳婷, 囧森瑟夫, 瀚墨	动画, 奇幻	2019/7/26	116
2	2	复仇者联盟4: 终局之战	安东尼·罗素	小罗伯特·唐尼, 克里斯·埃文斯, 马克·鲁法洛	动作, 科幻	2019/4/26	181
3	3	疯狂的外星人	宁浩	黄渤, 沈腾, 徐峥	喜剧, 科幻	2019-02-05 00:00:00	116
4	4	少年的你	曾国祥	周冬雨, 易烊千玺, 张耀	剧情, 爱情	2019-10-25 00:00:00	136
5	5	攀登者	李仁港	吴京, 章子怡, 张译	剧情, 冒险	2019-09-30 00:00:00	121
6	6	我和我的祖国	陈凯歌	黄渤, 张译, 葛优	剧情, 历史	2019-09-30 00:00:00	155
7	7	中国机长	刘伟强	张涵予, 欧豪, 杜江	剧情, 灾难	2019-09-30 00:00:00	111
8	8	烈火英雄	陈国辉	黄晓明, 杜江, 谭卓	剧情, 灾难	2019-08-01 00:00:00	120
9	9	误杀	柯汶利	肖央, 谭卓, 陈冲	剧情, 犯罪	2019-12-13 00:00:00	112
10	10	叶问4: 完结篇	叶伟信	甄子丹, 吴樾, 吴建豪	动作, 传记	2019-12-20 00:00:00	102
11	11	受益人	申奥	大鹏, 柳岩, 张子贤	剧情, 犯罪	2019-11-08 00:00:00	112
12	12	八佰	管虎	王千源, 张译, 姜武	剧情, 战争	2020-08-21 00:00:00	147
13	13	The Dark Knight	Christopher Nolan	Christian Bale, Heath Ledger, Mic...	Action, Crime, Drama	2008-07-18 00:00:00	152
14	14	Inception	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-...	Action, Adventure, Sci-Fi	2010-07-16 00:00:00	148
15	15	The Matrix Reloaded	Lana Wachowski, Lilly Wachowski	Keanu Reeves, Laurence Fishburne, ...	Action, Sci-Fi	2003-05-15 00:00:00	138
16	16	The Godfather: Part II	Francis Ford Coppola	Al Pacino, Robert De Miro, Talia ...	Crime, Drama	1974-12-20 00:00:00	155
17	17	Interstellar	Christopher Nolan	Matthew McConaughey, Anne Hathawa...	Adventure, Drama, Sci-Fi	2014-11-07 00:00:00	169
18	18	The Avengers	Joss Whedon	Robert Downey Jr., Chris Evans, M...	Action, Adventure, Sci-Fi	2012-05-04 00:00:00	143
19	19	Pulp Fiction	Quentin Tarantino	John Travolta, Samuel L. Jackson, ...	Crime, Drama, Thriller	1994-10-14 00:00:00	154
20	20	Iron Man	Jon Favreau	Robert Downey Jr., Gwyneth Paltro...	Action, Adventure, Sci-Fi	2008-05-02 00:00:00	126

图 4 movies 表

2.4 放映信息表数据需求:

- (1)放映 id: 用于唯一标识每次放映的身份。
- (2)电影 id: 放映的电影的唯一标识。
- (3)放映厅: 放映电影的影厅编号或名称。
- (4)放映时间: 放映电影的日期和时间。
- (5)状态: 放映的状态, 例如正在进行、已结束等。

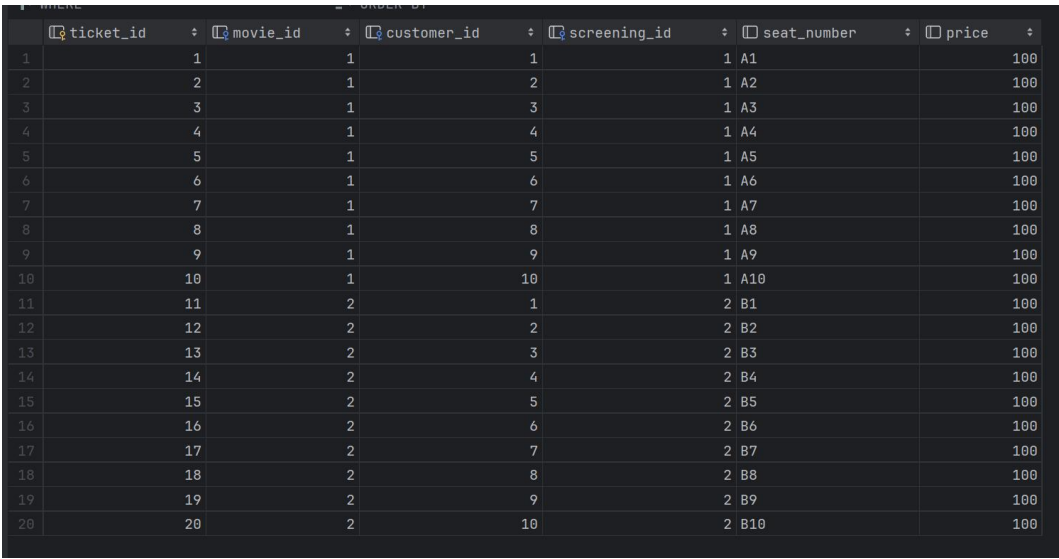
	WHERE		ORDER BY				
	screening_id	movie_id	screening_room	screening_time	status		
1	1	1	1号放映厅	2019-04-26 19:00:00	已放映		
2	2	2	2号放映厅	2019-07-26 19:00:00	已放映		
3	3	3	3号放映厅	2019-02-05 19:00:00	已放映		
4	4	4	4号放映厅	2019-10-25 19:00:00	已放映		
5	5	5	5号放映厅	2019-09-30 19:00:00	已放映		
6	6	6	6号放映厅	2019-09-30 19:00:00	已放映		
7	7	7	7号放映厅	2019-09-30 19:00:00	已放映		
8	8	8	8号放映厅	2019-08-01 19:00:00	已放映		
9	9	9	9号放映厅	2019-12-13 19:00:00	已放映		
10	10	10	10号放映厅	2019-12-20 19:00:00	已放映		
11	11	11	11号放映厅	2019-11-08 19:00:00	已放映		
12	12	12	12号放映厅	2019-10-25 19:00:00	已放映		
13	13	13	13号放映厅	2019-09-30 19:00:00	已放映		
14	14	14	14号放映厅	2019-09-30 19:00:00	已放映		
15	15	15	15号放映厅	2019-09-30 19:00:00	已放映		
16	16	16	16号放映厅	2019-08-01 19:00:00	已放映		
17	17	17	17号放映厅	2019-12-13 19:00:00	已放映		
18	18	18	18号放映厅	2019-12-20 19:00:00	已放映		
19	19	19	19号放映厅	2019-11-08 19:00:00	已放映		
20	20	20	20号放映厅	2019-10-25 19:00:00	已放映		

图 5 screenings 表



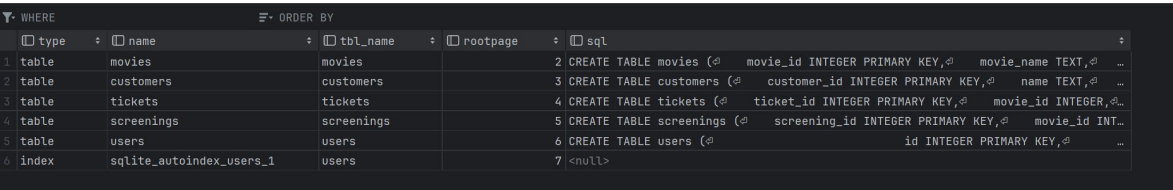
## 2.5 影票信息数据需求:

- (1)影票 id: 用于唯一标识每张影票的身份。
- (2)电影 id: 影票所属的电影的唯一标识。
- (3)顾客 id: 购买影票的顾客的唯一标识。
- (4)座位号: 影票对应的座位号或座位编号。
- (5)价格: 影票的售价或价格信息。



	ticket_id	movie_id	customer_id	screening_id	seat_number	price
1	1	1	1	1	A1	100
2	2	1	1	2	A2	100
3	3	1	1	3	A3	100
4	4	1	1	4	A4	100
5	5	1	1	5	A5	100
6	6	1	1	6	A6	100
7	7	1	1	7	A7	100
8	8	1	1	8	A8	100
9	9	1	1	9	A9	100
10	10	1	1	10	A10	100
11	11	2	1	1	B1	100
12	12	2	2	2	B2	100
13	13	2	2	3	B3	100
14	14	2	2	4	B4	100
15	15	2	2	5	B5	100
16	16	2	2	6	B6	100
17	17	2	2	7	B7	100
18	18	2	2	8	B8	100
19	19	2	2	9	B9	100
20	20	2	2	10	B10	100

图 6 tickets 表



	type	name	tbl_name	rootpage	sql
1	table	movies	movies		2 CREATE TABLE movies (movie_id INTEGER PRIMARY KEY, movie_name TEXT, ...
2	table	customers	customers		3 CREATE TABLE customers (customer_id INTEGER PRIMARY KEY, name TEXT, ...
3	table	tickets	tickets		4 CREATE TABLE tickets (ticket_id INTEGER PRIMARY KEY, movie_id INTEGER, ...
4	table	screenings	screenings		5 CREATE TABLE screenings (screening_id INTEGER PRIMARY KEY, movie_id INT...
5	table	users	users		6 CREATE TABLE users (id INTEGER PRIMARY KEY, ...
6	index	sqlite_autoindex_users_1	users		7 <null>

图 7 ssqlite\_master 表

## 3 代码实现

### 3.1 具体代码及注释

```
import sqlite3

import pandas as pd

import shutil

import hashlib

# 连接到数据库

conn = sqlite3.connect('database.db')

c = conn.cursor()

# 创建用户表

def create_user_table():

    c.execute("""CREATE TABLE IF NOT EXISTS users (

                id INTEGER PRIMARY KEY,

                username TEXT NOT NULL UNIQUE,

                password TEXT NOT NULL

            """)

    conn.commit()

# 用户注册

def register_user():
```

```

username = input("请输入用户名: ")

password = input("请输入密码: ")

# 对密码进行哈希处理

hashed_password = hashlib.sha256(password.encode()).hexdigest()

# 尝试插入新用户信息到数据库

try:

    c.execute("INSERT INTO users (username, password) VALUES (?, ?)",
(username, hashed_password))

    conn.commit()

    print("注册成功! ")

except sqlite3.IntegrityError:

    print("用户名已存在, 请选择其他用户名。")


# 用户登录

def login_user():

    username = input("请输入用户名: ")

    password = input("请输入密码: ")

# 对密码进行哈希处理

hashed_password = hashlib.sha256(password.encode()).hexdigest()

# 查询数据库中是否存在匹配的用户名和密码

c.execute("SELECT * FROM users WHERE username=? AND password=?",

```

```

(username, hashed_password))

    user = c.fetchone()

    if user:

        print("登录成功！ ")

    else:

        print("用户名或密码错误。 ")


def operation_user():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请输入要进行的操作： ")

    print("1. 查询整个用户表")

    print("2. 查询特定用户")

    print("3. 插入用户")

    print("4. 更新用户密码")

    print("5. 删除用户")

    table_num = int(input())

    if table_num == 1:

        query_all_users()

    elif table_num == 2:

        query_user_by_id()

    elif table_num == 3:

        insert_user()

    elif table_num == 4:

```

```

        update_user_password()

    elif table_num == 5:
        delete_user_by_id()

    conn.close()

# 查询整个用户表
def query_all_users():
    c.execute("SELECT * FROM users")

    users = c.fetchall()

    for user in users:
        print(user)

# 查询特定用户
def query_user_by_id():
    user_id = input("请输入要查询的用户 ID: ")

    c.execute("SELECT * FROM users WHERE id=?", (user_id,))

    user = c.fetchone()

    if user:
        print(user)

    else:
        print("未找到该用户")

# 插入用户
def insert_user():

```

```

username = input("请输入用户名: ")

password = input("请输入密码: ")

hashed_password = hashlib.sha256(password.encode()).hexdigest()

try:
    c.execute("INSERT INTO users (username, password) VALUES (?, ?)",
(username, hashed_password))

    conn.commit()

    print("用户添加成功! ")

except sqlite3.IntegrityError:

    print("用户名已存在, 请选择其他用户名。")


# 更新用户密码

def update_user_password():

    user_id = input("请输入要更新密码的用户 ID: ")

    new_password = input("请输入新密码: ")

    hashed_password = hashlib.sha256(new_password.encode()).hexdigest()

    c.execute("UPDATE users SET password=? WHERE id=?", (hashed_password,
user_id))

    conn.commit()

    print("密码更新成功! ")


# 删除用户

def delete_user_by_id():

    user_id = input("请输入要删除的用户 ID: ")

    c.execute("DELETE FROM users WHERE id=?", (user_id,))

```

```
conn.commit()

print("用户删除成功！")
```

```
def create_tables():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    create_user_table()

    # 创建电影信息表

    cursor.execute("""

    CREATE TABLE IF NOT EXISTS movies (

        movie_id INTEGER PRIMARY KEY,

        movie_name TEXT,

        director TEXT,

        cast TEXT,

        genre TEXT,

        release_date TEXT,

        duration INTEGER

    )

    """)

    # 创建顾客信息表

    cursor.execute("""

    CREATE TABLE IF NOT EXISTS customers (

        customer_id INTEGER PRIMARY KEY,
```

```
        name TEXT,  
        gender TEXT,  
        age INTEGER,  
        phone TEXT,  
        email TEXT  
    )  
    ")
```

# 创建影票信息表

```
cursor.execute("  
CREATE TABLE IF NOT EXISTS tickets (  
    ticket_id INTEGER PRIMARY KEY,  
    movie_id INTEGER,  
    customer_id INTEGER,  
    screening_id INTEGER,  
    seat_number TEXT,  
    price REAL,  
    FOREIGN KEY (movie_id) REFERENCES movies (movie_id),  
    FOREIGN KEY (customer_id) REFERENCES customers (customer_id),  
    FOREIGN KEY (screening_id) REFERENCES screenings (screening_id)  
)  
")
```

# 创建放映信息表

```
cursor.execute("  
CREATE TABLE IF NOT EXISTS screenings (  

```



```
        screening_id INTEGER PRIMARY KEY,  
        movie_id INTEGER,  
        screening_room TEXT,  
        screening_time TEXT,  
        status TEXT,  
        FOREIGN KEY (movie_id) REFERENCES movies (movie_id)  
    )  
    """)
```

```
conn.commit()  
conn.close()
```

# 查找数据的函数

```
def query_data():  
    conn = sqlite3.connect('电影院管理系统.db')  
    cursor = conn.cursor()  
  
    print("请输入要查询的表的序号：")  
    print("1. 电影信息表")  
    print("2. 顾客信息表")  
    print("3. 影票信息表")  
    print("4. 放映信息表")  
    table_num = int(input())  
    if table_num == 1:
```

```
print("请问进行整表操作 or 操作行数据")

print("1. 整表操作")

print("2. 行数据操作")

table_num_1 = int(input())

if table_num_1 == 1:

    query_movies()

elif table_num_1 == 2:

    movie_id = input("请输入电影 ID: ")

    query_movies_id(movie_id)

elif table_num == 2:

    print("请问进行整表操作 or 操作行数据")

    print("1. 整表操作")

    print("2. 行数据操作")

    table_num_2 = int(input())

    if table_num_2 == 1:

        query_customers()

    elif table_num_2 == 2:

        customer_id = input("请输入顾客 ID: ")

        query_customers_id(customer_id)

elif table_num == 3:

    print("请问进行整表操作 or 操作行数据")

    print("1. 整表操作")

    print("2. 行数据操作")

    table_num_3 = int(input())

    if table_num_3 == 1:
```

```

        query_tickets()

    elif table_num_3 == 2:

        ticket_id = input("请输入影票 ID: ")

        query_tickets_id(ticket_id)

elif table_num == 4:

    print("请问进行整表操作 or 操作行数据")

    print("1. 整表操作")

    print("2. 行数据操作")

    table_num_4 = int(input())

    if table_num_4 == 1:

        query_screenings()

    elif table_num_4 == 2:

        screening_id = input("请输入放映 ID: ")

        query_screenings_id(screening_id)

# elif action_num == 3:

#     update_data()

# elif action_num == 4:

#     delete_data()

# elif action_num == 5:

#     backup_data()

conn.close()

def query_movies():

    conn = sqlite3.connect('电影院管理系统.db')

```

```
cursor = conn.cursor()

cursor.execute('SELECT * FROM movies')
movies = cursor.fetchall()

print("\n 电影信息: ")

for movie in movies:
    print(movie)

conn.close()

def query_customers():
    conn = sqlite3.connect('电影院管理系统.db')
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM customers')
    customers = cursor.fetchall()
    print("\n 顾客信息: ")

    for customer in customers:
        print(customer)

    conn.close()

def query_tickets():
    conn = sqlite3.connect('电影院管理系统.db')
    cursor = conn.cursor()
```

```

        cursor.execute('SELECT * FROM tickets')

        tickets = cursor.fetchall()

        print("\n 影票信息: ")

        for ticket in tickets:

            print(ticket)


    conn.close()


def query_screenings():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()


    cursor.execute('SELECT * FROM screenings')

    screenings = cursor.fetchall()

    print("\n 放映信息: ")

    for screening in screenings:

        print(screening)


    conn.close()


def query_movies_id(movie_id):

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

```

```
cursor.execute('SELECT * FROM movies WHERE rowid=?', (movie_id,))
```

```
movie = cursor.fetchone()
```

```
if movie:
```

```
    print("\n 电影信息: ")
```

```
    print(movie)
```

```
else:
```

```
    print("未找到该电影")
```

```
conn.close()
```

```
def query_customers_id(customer_id):
```

```
    conn = sqlite3.connect('电影院管理系统.db')
```

```
    cursor = conn.cursor()
```

```
    cursor.execute('SELECT * FROM customers WHERE rowid=?', (customer_id,))
```

```
    customer = cursor.fetchone()
```

```
    if customer:
```

```
        print("\n 顾客信息: ")
```

```
        print(customer)
```

```
    else:
```

```
        print("未找到该顾客")
```

```
    conn.close()
```

```
def query_tickets_id(ticket_id):
```

```

conn = sqlite3.connect('电影院管理系统.db')

cursor = conn.cursor()

cursor.execute('SELECT * FROM tickets WHERE rowid=?', (ticket_id,))

ticket = cursor.fetchone()

if ticket:

    print("\n 影票信息: ")

    print(ticket)

else:

    print("未找到该影票")

conn.close()

def query_screenings_id(screening_id):

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    cursor.execute('SELECT      *      FROM      screenings      WHERE      rowid=?',
(screening_id,))

    screening = cursor.fetchone()

    if screening:

        print("\n 放映信息: ")

        print(screening)

    else:

        print("未找到该放映信息")

```

```
conn.close()

#插入数据的函数
def insert_data():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    # 从 xlsx 文件中读取数据并插入到表中

    print("请输入要插入的表的序号： ")

    print("1. 电影信息表")

    print("2. 顾客信息表")

    print("3. 影票信息表")

    print("4. 放映信息表")

    table_num = int(input())

    if table_num == 1:

        insert_movies()

    elif table_num == 2:

        insert_customers()

    elif table_num == 3:

        insert_tickets()

    elif table_num == 4:
```



```

        insert_screenings()

    conn.close()

def insert_movies():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请选择插入方式： ")

    print("1. 整表操作")

    print("2. 行数据操作")

    choice = int(input())

    if choice == 1:

        # 整表操作示例：从文件中读取电影数据并插入

        movies_df = pd.read_excel('movies.xlsm')

        movies_df.to_sql('movies', conn, if_exists='append', index=False)

    elif choice == 2:

        # 行数据操作示例：手动输入电影 ID 并插入

        movie_id = int(input("请输入电影 ID: "))

        movie_name = input("请输入电影名称: ")

        director = input("请输入导演姓名: ")

        cast = input("请输入演员阵容: ")

        gengre = input("请输入类型: ")

        release_date = input("请输入上映时间: ")

```

```

        duration = input("请输入电影时长: ")

        cursor.execute('INSERT INTO movies (movie_id,
movie_name,director,cast,gengre,release_date,duration) VALUES (?, ?, ?, ?, ?, ?, ?)',
(movie_id, movie_name,director,cast,gengre,release_date,duration))

    else:

        print("无效的选择，请重新选择。")

```

```

conn.commit()

print("电影信息已插入")

conn.close()

```

# 插入顾客信息的函数

```

def insert_customers():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请选择插入方式: ")

    print("1. 整表操作")

    print("2. 行数据操作")

    choice = int(input())

    if choice == 1:

        # 整表操作示例：从文件中读取顾客数据并插入

        customers_df = pd.read_excel('customers.xlsm')

```

```

        customers_df.to_sql('customers', conn, if_exists='append', index=False)
elif choice == 2:
    # 行数据操作示例：手动输入顾客 ID 并插入

    customer_id = int(input("请输入顾客 ID: "))

    name = input("请输入顾客姓名: ")

    cursor.execute('INSERT INTO customers (customer_id, name) VALUES
(?, ?)', (customer_id, name))

else:

    print("无效的选择，请重新选择。")


conn.commit()

print("顾客信息已插入")

conn.close()


# 插入影票信息的函数
def insert_tickets():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()


    print("请选择插入方式: ")

    print("1. 整表操作")

    print("2. 行数据操作")

    choice = int(input())

    if choice == 1:

```

```

# 整表操作示例：从文件中读取影票数据并插入

tickets_df = pd.read_excel('tickets.xlsx')

tickets_df.to_sql('tickets', conn, if_exists='append', index=False)

elif choice == 2:

    # 行数据操作示例：手动输入影票 ID 并插入

    ticket_id = int(input("请输入影票 ID: "))

    movie_id = int(input("请输入电影 ID: "))

    customer_id = int(input("请输入顾客 ID: "))

    cursor.execute('INSERT INTO tickets (ticket_id, movie_id, customer_id)
VALUES (?, ?, ?)', (ticket_id, movie_id, customer_id))

else:

    print("无效的选择，请重新选择。")


conn.commit()

print("影票信息已插入")

conn.close()


# 插入放映信息的函数

def insert_screenings():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请选择插入方式: ")

    print("1. 整表操作")

    print("2. 行数据操作")

```

```

choice = int(input())

if choice == 1:
    # 整表操作示例：从文件中读取放映数据并插入
    screenings_df = pd.read_excel('screenings.xlsm')
    screenings_df.to_sql('screenings', conn, if_exists='append', index=False)
elif choice == 2:
    # 行数据操作示例：手动输入放映 ID 并插入
    screening_id = int(input("请输入放映 ID: "))
    movie_id = int(input("请输入电影 ID: "))
    screening_room = input("请输入放映厅: ")
    screening_time = input("请输入放映时间: ")
    status = input("请输入状态: ")

    cursor.execute('INSERT INTO screenings (screening_id, movie_id,
screening_room, screening_time, status) VALUES (?, ?, ?, ?, ?)', (screening_id, movie_id,
screening_room, screening_time, status))
else:
    print("无效的选择，请重新选择。")

conn.commit()

print("放映信息已插入")

conn.close()

#更新数据的函数

```

```
def update_data():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请输入要更新的表的序号： ")

    print("1. 电影信息表")

    print("2. 顾客信息表")

    print("3. 影票信息表")

    print("4. 放映信息表")

    table_num = int(input())

    if table_num == 1:

        update_movies()

    elif table_num == 2:

        update_customers()

    elif table_num == 3:

        update_tickets()

    elif table_num == 4:

        update_screenings()

    conn.close()

def update_movies():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()
```

```

print("请输入要更新的电影 ID: ")

movie_id = int(input("电影 ID: "))

new_name = input("新的电影名称: ")

new_director = input("新的导演: ")

new_cast = input("新的演员阵容: ")

new_gengre = input("新的类型: ")

new_release_date = input("新的上映日期: ")

new_duration = int(input("新的时长: "))


cursor.execute("""
UPDATE movies
SET movie_name = ?,
    director = ?,
    cast = ?,
    gengre = ?,
    release_date = ?,
    duration = ?
WHERE movie_id = ?

", (new_name, new_director, new_cast, new_gengre, new_release_date,
new_duration, movie_id))

conn.commit()

print("电影信息已更新")

```

```

conn.close()

def update_customers():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请输入要更新的顾客 ID: ")

    customer_id = int(input("顾客 ID: "))

    new_name = input("新的顾客姓名: ")

    new_gender = input("新的顾客性别: ")

    new_age = int(input("新的顾客年龄: "))

    new_phone = input("新的顾客电话: ")

    new_email = input("新的顾客邮箱: ")

    cursor.execute("""

        UPDATE customers

        SET name = ?,

            gender = ?,

            age = ?,

            phone = ?,

            email = ?

        WHERE customer_id = ?

        """, (new_name, new_gender, new_age, new_phone, new_email,
customer_id))

    conn.commit()

```



```
print("顾客信息已更新")
```

```
conn.close()
```

```
def update_tickets():
```

```
    conn = sqlite3.connect('电影院管理系统.db')
```

```
    cursor = conn.cursor()
```

```
    print("请输入要更新的影票 ID: ")
```

```
    ticket_id = int(input("影票 ID: "))
```

```
    new_movie_id = int(input("新的电影 ID: "))
```

```
    new_customer_id = int(input("新的顾客 ID: "))
```

```
    new_screening_id = int(input("新的放映 ID: "))
```

```
    new_seat_number = input("新的座位号: ")
```

```
    new_price = float(input("新的票价: "))
```

```
    cursor.execute("""
```

```
        UPDATE tickets
```

```
        SET movie_id = ?,
```

```
            customer_id = ?,
```

```
            screening_id = ?,
```

```
            seat_number = ?,
```

```
            price = ?
```

```
        WHERE ticket_id = ?
```

```
        ", (new_movie_id, new_customer_id, new_screening_id, new_seat_number,  
new_price, ticket_id))
```

```
conn.commit()
```

```
print("影票信息已更新")
```

```
conn.close()
```

```
def update_screenings():
```

```
    conn = sqlite3.connect('电影院管理系统.db')
```

```
    cursor = conn.cursor()
```

```
    print("请输入要更新的放映 ID: ")
```

```
    screening_id = int(input("放映 ID: "))
```

```
    new_movie_id = int(input("新的电影 ID: "))
```

```
    new_screening_room = input("新的放映厅: ")
```

```
    new_screening_time = input("新的放映时间: ")
```

```
    new_status = input("新的状态: ")
```

```
    cursor.execute("""
```

```
        UPDATE screenings
```

```
        SET movie_id = ?,
```

```
            screening_room = ?,
```

```
            screening_time = ?,
```

```
        status = ?

        WHERE screening_id = ?

        ", (new_movie_id, new_screening_room, new_screening_time, new_status,
screening_id))
```

```
conn.commit()

print("放映信息已更新")
```

```
conn.close()
```

#删除数据的函数

```
def delete_data():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    print("请输入要删除的表的序号： ")

    print("1. 电影信息表")

    print("2. 顾客信息表")

    print("3. 影票信息表")

    print("4. 放映信息表")

    table_num = int(input())

    if table_num == 1:
```

```
        delete_movies()
elif table_num == 2:
    delete_customers()
elif table_num == 3:
    delete_tickets()
elif table_num == 4:
    delete_screenings()

conn.close()
```

```
def delete_movies():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    movie_id = int(input("请输入要删除的电影 ID: "))

    cursor.execute("""
        DELETE FROM movies
        WHERE movie_id = ?
        """, (movie_id,))

    conn.commit()

    print("电影信息已删除")

    conn.close()
```

```
def delete_customers():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    customer_id = int(input("请输入要删除的顾客 ID: "))

    cursor.execute("""
        DELETE FROM customers
        WHERE customer_id = ?
    """, (customer_id,))

    conn.commit()

    print("顾客信息已删除")

    conn.close()

def delete_tickets():

    conn = sqlite3.connect('电影院管理系统.db')

    cursor = conn.cursor()

    ticket_id = int(input("请输入要删除的影票 ID: "))
```

```
cursor.execute("""
    DELETE FROM tickets
    WHERE ticket_id = ?
    """, (ticket_id,))
```

```
conn.commit()
print("影票信息已删除")
```

```
conn.close()
```

```
def delete_screenings():
```

```
    conn = sqlite3.connect('电影院管理系统.db')
    cursor = conn.cursor()
```

```
    screening_id = int(input("请输入要删除的放映 ID: "))
```

```
    cursor.execute("""
        DELETE FROM screenings
        WHERE screening_id = ?
        """, (screening_id,))
```

```
    conn.commit()
    print("放映信息已删除")
```

```
conn.close()
```

```
def backup_data():
```

```
    # 备份数据库文件
```

```
    shutil.copyfile('电影院管理系统.db', '电影院管理系统_backup.db')
```

```
    print("数据库备份完成")
```

```
# 主菜单
```

```
def main_menu():
```

```
    while True:
```

```
        print("1. 查询数据")
```

```
        print("2. 插入数据")
```

```
        print("3. 更新数据")
```

```
        print("4. 删除数据")
```

```
        print("5. 备份数据")
```

```
        print("6. 用户注册")
```

```
        print("7. 用户登录")
```

```
        print("8. 其他操作")
```

```
        print("0. 退出程序")
```

```
choice = input("请输入选项: ")

if choice == '1':
    query_data()
elif choice == '2':
    insert_data()
elif choice == '3':
    update_data()
elif choice == '4':
    delete_data()
elif choice == '5':
    backup_data()
elif choice == '6':
    register_user()
elif choice == '7':
    login_user()
elif choice == '8':
    operation_user()
elif choice == '0':
    print("退出程序")
    break
else:
    print("无效输入, 请重新输入")
```

```
if __name__ == '__main__':
```



```
create_tables()

main_menu()
```

### 3.2 具体实现

```
1. 查询数据
2. 插入数据
3. 更新数据
4. 删除数据
5. 备份数据
6. 用户注册
7. 用户登录
8. 其他操作
9. 退出程序
请输入选项: 1
请输入要查询的表的序号:
1. 电影信息表
2. 顾客信息表
3. 影票信息表
4. 放映信息表
2
请问进行整表操作or操作行数据
1. 整表操作
2. 行数据操作
2
请输入顾客ID: 15

顾客信息:
(15, '许十七', '男', 23, '12500125000', 'xushiqi@example.com')
1. 查询数据
2. 插入数据
3. 更新数据
4. 删除数据
5. 备份数据
6. 用户注册
7. 用户登录
8. 其他操作
9. 退出程序
请输入选项: |
```

图 8

## 4 总 结

总结：

在设计电影院信息管理系统的团队合作过程中，我与团队成员共同深入理解了数据库设计的原理和应用，并通过实践加深了对 SQL 和 SQLite 的掌握。我们系统的设计旨在为电影院提供一个全面的管理平台，涵盖了管理人员、顾客、电影、放映信息和影票信息五个主要方面。

首先，团队明确了系统的设计目标，即通过数据库实现对电影院各类信息的有效管理和查询。在确定目标后，我们进行了需求分析，详细列出了每个表格的字段和约束条件。例如，管理人员表需要存储用户名和密码哈希值，以确保登录的安全性；顾客表需要包括姓名、性别、年龄、联系电话和电子邮件，以便于联系和服务顾客；电影表记录了电影的详细信息，包括标题、导演、主要演员、类型、上映日期和时长；放映信息表和影票信息表则分别记录了电影的放映安排和影票的销售情况。

在数据库结构设计阶段，我们采用了关系型数据库模型，通过设置主键和外键来确保表与表之间的关系。每个表格的字段类型和约束条件都经过团队仔细讨论和设计，以确保数据的完整性和一致性。例如，我们使用 ENUM 类型来限制性别和放映状态的取值范围，使用 FOREIGN KEY 来关联电影和放映信息、影票和顾客信息等。

实际编码过程中，团队成员分工合作，创建了各个表格，并编写了 SQL 语句来实现数据的插入、查询、更新和删除操作。在这个过程中，

我们不仅巩固了对 SQL 语法的理解，也提升了解决实际问题的能力。我们在创建和管理 SQLite 数据库时，遇到了一些挑战，例如处理外键约束、优化查询性能等，通过团队协作、查阅资料和反复调试，我们逐步解决了这些问题，积累了宝贵的经验。

本系统的设计和实现，不仅达到了预期的目标，还展示了数据库设计在实际应用中的重要性。通过这个项目，团队深刻体会到一个良好设计的数据库结构在数据管理中的重要作用。系统通过清晰的表结构和字段定义，确保了数据的完整性和一致性，能够支持复杂查询和数据分析，为电影院管理提供了强有力的技术支持。

总结这段团队合作经历，我不仅提升了数据库设计和管理的技能，还增强了分析和解决问题的能力，更重要的是提升了团队协作的能力。在未来的学习和工作中，我会继续深入学习数据库技术，不断提升自己的专业水平，为更复杂的系统设计和开发做好准备。此次项目的完成，是我们团队数据库学习中的一大步，也为我们未来的发展奠定了坚实的基础。。

陈梓欣

总结：

在本次团队合作中，编写电影院数据管理系统的过程中，我深刻体会到团队合作的重要性和高效协作的力量。我们团队首先进行了详细的任务分解，将系统分为多个模块，每个模块由专门的成员负责，这种明确分工提高了工作效率。

我们定期召开团队会议，讨论进展、分享问题并交流解决方案，确保了大家对项目的理解一致，减少了因信息不对称导致的错误。每个成员都有自己的分工，每个人都有自己擅长的方面，正是团队的通力合作保障了代码的完整性和一致性。

通过团队合作，我们不仅完成了项目，还在相互交流中学到了新的知识和技能，增强了技术能力和团队凝聚力。我们在开发过程中遇到的技术难题，通过共同努力得以解决，这种积极的团队氛围激励了我们继续前进。

通过这次团队合作，我深刻认识到明确分工、有效沟通、工具使用、相互学习和及时反馈的重要性。这些都是成功完成项目的关键因素，未来工作中我会继续应用这些宝贵经验，不断提升团队合作能力。

陆浩然

总结：

我们本次作业主题为电影院系统，通过建立数据库来对电影院的票根，顾客信息等进行管理，对电影院的信息进行分析总结，有利于电影院管理人员的工作。

在本次大作业中，我体会到了技术的融合，对于数据库的本次作业，不止可以用 `sqlserver` 写代码，也可以用其他软件实现功能。在实现代码过程中，我们利用表格来进行数据的录入，避免了人工书写 `sql` 语句录入，大大节省了人工力度，缩短了代码长度，给阅读者提供了便利。

在代码实验过程中，我们通过 `e-r` 图对整体系统进行概括，再通过代码一步步实现。本次代码中，通过对用户权限的管理，对电影院的信息隐私进行管理，保障了信息的安全性。

本次大作业中，让我体会到了团队合作的重要性，团队中的每个人都精心负责着相应的板块内容，有效的分工与安排让本次大作业可以在短暂的时间内来确立基础的框架，并在团队小组讨论的过程中经行有效的沟通，去将大作业的内容经行完善以及问题出现的地方进行有效的修改.使得大作业更加趋于完善。团队合作实现了一加一大于二的表现，团队合作是有效提出问题并可以为有效解决问题提供良好的条件，也是我们能有效解决各种难题的伟大力量，高举团队合作大旗，一起去面对诸多问题并将问题逐个征服。

方艺安

总结：

本次的大作业课题为电影院售票管理系统。在这次作业中，我参与了数据库的整体设计，深入学习了 ER 图以及更深的了解了 SQL 语言。本次电影院售票系统的设计采用了模块化、层次化的设计理念。首先进行了系统的需求分析，明确了系统的基本功能和性能要求。随后，根据需求分析结果，设计了系统的整体架构。

在数据库的搭建方面，选择了关系型数据库作为存储数据的方案，并设计了包括用户表、影片表、放映时间表等在内的一系列数据表，以及它们之间的关联关系。同时，考虑到售票系统的实时性和并发性，我们采用了索引优化、查询优化等技术手段，提高了数据库的查询效率。

在系统的设计过程中，我们团队成员紧密合作，协作负责完成不同的部分。此外，在团队协作中，我学会了如何更好地与团队成员沟通协作，共同解决问题。通过定期的会议和交流，我们及时分享各自的工作进展和遇到的问题，共同寻求解决方案，从而提高了整个团队的工作效率。

综上所述，本次数据库电影院售票系统设计经历让我收获颇丰。我不仅提升了自己的技术能力和团队协作能力，还深刻认识到了需求分析、系统设计和优化等方面的重要性。在未来的学习和工作中，我将继续努力深入学习和探索数据库知识，为以后的工作和学习打下坚实基础。

刘书男

## 参考文献

- [1] 李璋,陈龙,陈逸凡,等.SQL 语言在数据库实践课程中的应用[J].科技风,2024,(08):98-100.DOI:10.19392/j.cnki.1671-7341.202408033.
- [2] 肖宁,周琴,胡方宇,等.基于 SQL Server 的教学评价管理系统设计[J].无线互联科技,2023,20(20):46-49.
- [3] 周杰洋,邹鹏辉,尹鹏飞.基于 SQL Server 的智慧图书馆数据库系统[J].信息与电脑(理论版),2023,35(19):120-123.
- [4] 李艳杰.基于 MySQL 数据库的数据安全应用设计[J].现代信息技术,2023,7(12):151-154.DOI:10.19850/j.cnki.2096-4706.2023.12.037.
- [5] 刘湘龙,曾丽.电影院系统数据库设计与实现[J].电脑知识与技术,2022,18(06):16-18.DOI:10.14004/j.cnki.ckt.2022.0332.
- [6] 施燕娜.某高校图书馆管理信息系统的设计与实现[D].江西财经大学,2019.DOI:10.27175/d.cnki.gjxcu.2019.000183.
- [7] 钟仙.实验室管理信息系统的设计与实现[D].电子科技大学,2018.