



华中科技大学

数据库系统原理实践报告

综合设计题目：图书管理系统

姓 名：沙岩

学 院：计算机科学与技术学院

专 业：计算机科学与技术

班 级：CS1603

学 号：U201614588

指导教师：瞿彬彬

分数	
教师签名	

2019 年 6 月 14 日

任务书

1 软件功能学习部分（必做题）

完成下列 1~2 题，并在实践报告中叙述过程，可适当辅以插图（控制在 A4 三页篇幅以内）

1) 练习 sqlserver 的两种完全备份方式：数据和日志文件的脱机备份、系统的备份功能。

2) 练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的 SQL 语句验证自己的权限配置是否成功。

2 Sql 练习部分（必做题）

2.1 建表

1) 假设某微博平台的数据库中有列关系，请在 DBMS 中创建这些关系，包括主码和外码的说明，并写出指定关系的建表 SQL 语句：

用户【用户 ID，姓名，性别，出生年份，所在城市】记录所有注册用户的基本信息，英文表名和字段名如下：

USER(*UID* 整型, *NAME* 字符串, *SEX* 一位汉字, *BYEAR* 整型, *CITY* 字符串);

分类【分类 ID，分类名称】记录微博平台中所有可能涉及的微博的类型，例如文学、艺术、军事等，英文表名和字段名如下：

LABEL(*LID* 整型, *LNAME* 字符串);

博文【博文 ID，标题，用户 ID，年，月，日，正文】记录每一篇微博的基本信息，英文表名和字段名如下：

MBLOG(*BID* 整型, *TITLE* 字符串, *UID* 整型, *PYEAR* 整型, *PMONTH* 整型, *PDAY* 整型, *CONT* 字符串);

写出该关系的建表 SQL 语句；

博文标注【博文 ID，分类 ID】记录每一篇微博的作者给该微博贴上的分类标签，一篇微博可以涉及不止一种分类，英文表名和字段名如下：

B_L(*BID* 整型, *LID* 整型);

关注【用户 ID，被关注用户 ID】记录每位用户关注的其他用户，每位用户可关注多人，英文表名和字段名如下：

FOLLOW(*UID* 整型, *UIDFLED* 整型);

好友【用户 ID， 好友 ID】记录每位用户的好友（可多个），英文表名和

字段名如下：

FRIENDS(*UID* 整型, *FUID* 整型);

订阅【用户 ID, 订阅分类 ID】记录用户订阅的每一种分类，英文表名和字段名如下：

SUB(*UID* 整型, *LID* 整型);

点赞【用户 ID, 博文 ID】记录用户点赞的每一篇微博，英文表名和字段名如下：

THUMB(*UID* 整型, *BID* 整型),

写出该关系的建表 SQL 语句；

头条【年，月，日，博文 ID, 顺序号】记录每一天的热度排名前十的博文 ID 号以及该博文在热度前十名中的排名，英文表名和字段名如下：

TOPDAY(*TYEAR* 整型, *TMONTH* 整型, *TDAY* 整型, *BID* 整型, *TNO* 整型)。

2) 观察性实验

用户在订阅分类时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

3) 数据准备

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

2.2 数据更新

1) 分别用一条 sql 语句完成对博文表基本的增、删、改的操作；

2) 批处理操作

将关注 3 号用户的用户信息插入到一个自定义的新表 FANS_3 中。

3) 数据导入导出

通过查阅 DBMS 资料学习数据导入导出功能，并将任务 2.1 所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

在后续的上机实验环节，通过导入导出或者备份机制实现前次上机环节的数据恢复。

4) 观察性实验

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

5) 触发器实验

编写一个触发器，用于实现对点赞表的完整性控制规则：当插入或者被点赞博文时，如果博文作者就是点赞者本人，则拒绝执行。

2.3 查询

请分别用一条 SQL 语句完成下列各个小题的需求：

1) 查询“张三”用户关注的所有用户的 ID 号、姓名、性别、出生年份、所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户 ID 号升序排列。

2) 查找没有被任何人点赞的博文 ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

3) 查找 2000 年以后出生的武汉市用户发表的进入过头条的博文 ID；

4) 查找订阅了所有分类的用户 ID；

5) 查找出生年份小于 1970 年或者大于 2010 年的用户 ID、出生年份、所在城市，要求 where 子句中只能有一个条件表达式；

6) 统计每个城市的用户数；

7) 统计每个城市的每个出生年份的用户数，并将结果按照城市的升序排列，同一个城市按照出生用户数的降序排列其相应的年份；

8) 查找被点赞数超过 10 的博文 ID 号；

9) 查找被 2000 年后出生的用户点赞数超过 10 的博文 ID 号；

10) 查找被 2000 年后出生的用户点赞数超过 10 的每篇博文的进入头条的次数；

11) 查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户 ID，要求不能使用嵌套查询，且 where 子句中最多只能包含两个条件；

12) 查找标题中包含了“最多地铁站”和“_华中科技大学”两个词的博文基本信息；

13) 查找所有相互关注的用户对的两个 ID 号，要求不能使用嵌套查询；

14) 查找好友圈包含了 5 号用户好友圈的用户 ID；

15) 查找 2019 年 4 月 20 日每一篇头条博文的 ID 号、标题以及该博文的每一个分类 ID，要求即使该博文没有任何分类 ID 也要输出其 ID 号、标题；

16) 查找至少有 3 名共同好友的所有用户对的两个 ID 号。

17) 创建视图：查阅 DBMS 内部函数，创建一个显示当日热度排名前十的微博信息的视图，其中的属性包括：博文 ID、博文标题、发表者 ID、发表者姓名、被点赞数

2.4 了解系统的查询性能分析功能（选做）

选择上述 2.3 任务中某些较为复杂的 SQL 语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

目 录

1 课程任务概述	1
1 软件功能学习部分（必做题）	1
2 Sql 练习部分（必做题）	1
2.2 数据更新.....	1
2.3 查询.....	1
2.4 了解系统的查询性能分析功能（选做）	1
2 软件功能学习	2
2.1 任务要求.....	2
2.2 完成过程.....	2
2.3 任务总结.....	6
3 Sql 练习	7
3.1 任务要求.....	7
3.2 完成过程.....	7
3.3 任务总结.....	21
4 综合实践任务	23
4.1 系统设计目标.....	23
4.2 需求分析.....	23
4.3 总体设计.....	24
4.4 数据库设计.....	26
4.5 详细设计与实现.....	29
4.6 系统测试.....	37
4.7 系统设计与实现总结.....	47
5 课程总结	49
参考文献	50

1 课程任务概述

1 软件功能学习部分（必做题）

学会 SQL SERVER 的两种完全备份方式，并且在数据库上新增用户并且配置权限。

2 Sql 练习部分（必做题）

2.1 建表

- 1) 根据题目的要求，构建相关的表，并且要有主键外键等内容。
- 2) 检验用户在订阅分类时是否一定要参考被参照关系的主码。
- 3) 根据 17 条任务的要求，合理地插入相关数据，便于后面检验。

2.2 数据更新

- 1) 分别用一条 sql 语句完成对博文表基本的增、删、改的操作；
- 2) 将关注 3 号用户的用户信息插入到一个自定义的新表 FANS_3 中。
- 3) 学会数据的导入导出
- 4) 建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。
- 5) 编写一个符合要求的触发器，并且进行检验

2.3 查询

分别用一条 SQL 语句完成 17 个小题的需求，并且语句要符合小题要求。

2.4 了解系统的查询性能分析功能（选做）

选择上述 2.3 任务中某些较为复杂的 SQL 语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

2 软件功能学习

2.1 任务要求

1) 练习 sqlserver 的两种完全备份方式：数据和日志文件的脱机备份、系统的备份功能。

2) 练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的 SQL 语句验证自己的权限配置是否成功。

2.2 完成过程

(包括主要操作步骤描述及其执行效果，或者所用的 SQL 语句及语句执行、调试的主要过程、效果。)

2.2.1 sqlserver 的完全备份方式

(1) 脱机备份

右键数据库->任务->脱机，将数据库进行脱机处理，然后在文件夹中，找到数据库文件，我的路径为：

E:\apps\SQLServer\MSSQL14.MSSQLSERVER\MSSQL\DATA

找到两个文件：weibo.mdf 与 weibo_log.ldf，复制到另一个位置上去，就可以完成脱机备份。

之后再右键数据库->任务->联机，将数据库进行联机。

恢复数据库：

右键数据库，选择附加->然后再选择自己之前复制过的*.mdf 文件和 *_log.ldf 文件，按确认即可恢复。如下图 2.1 所示：

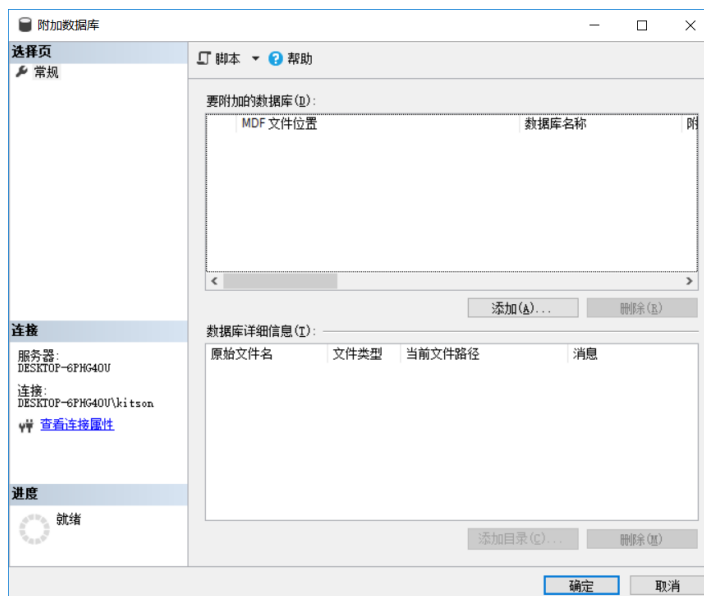


图 2.1 数据库恢复

(2) 系统备份:

对着数据库, 右键->任务->备份, 然后选择备份的路径, 按确认, 即可把数据库备份成.bak 文件, 如下图 2.2 所示:

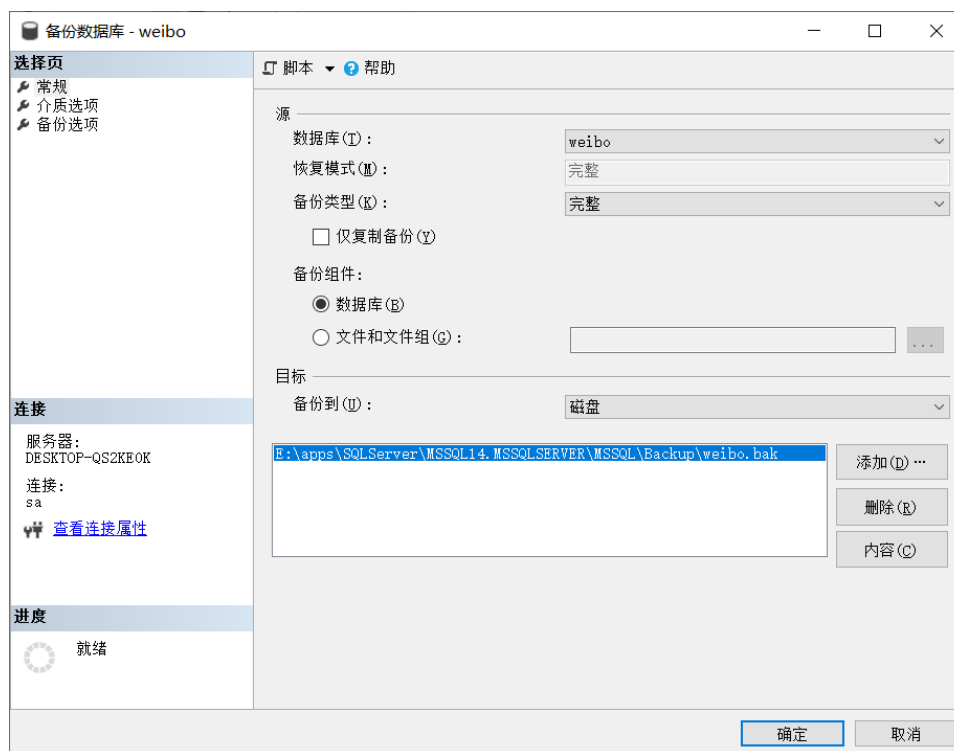


图 2.2 数据库备份

还原:

对着数据库, 右键->还原数据库->选择设备->选择 bak 文件的路径, 按确认, 即可把数据库还原, 如下图 2.3 所示:

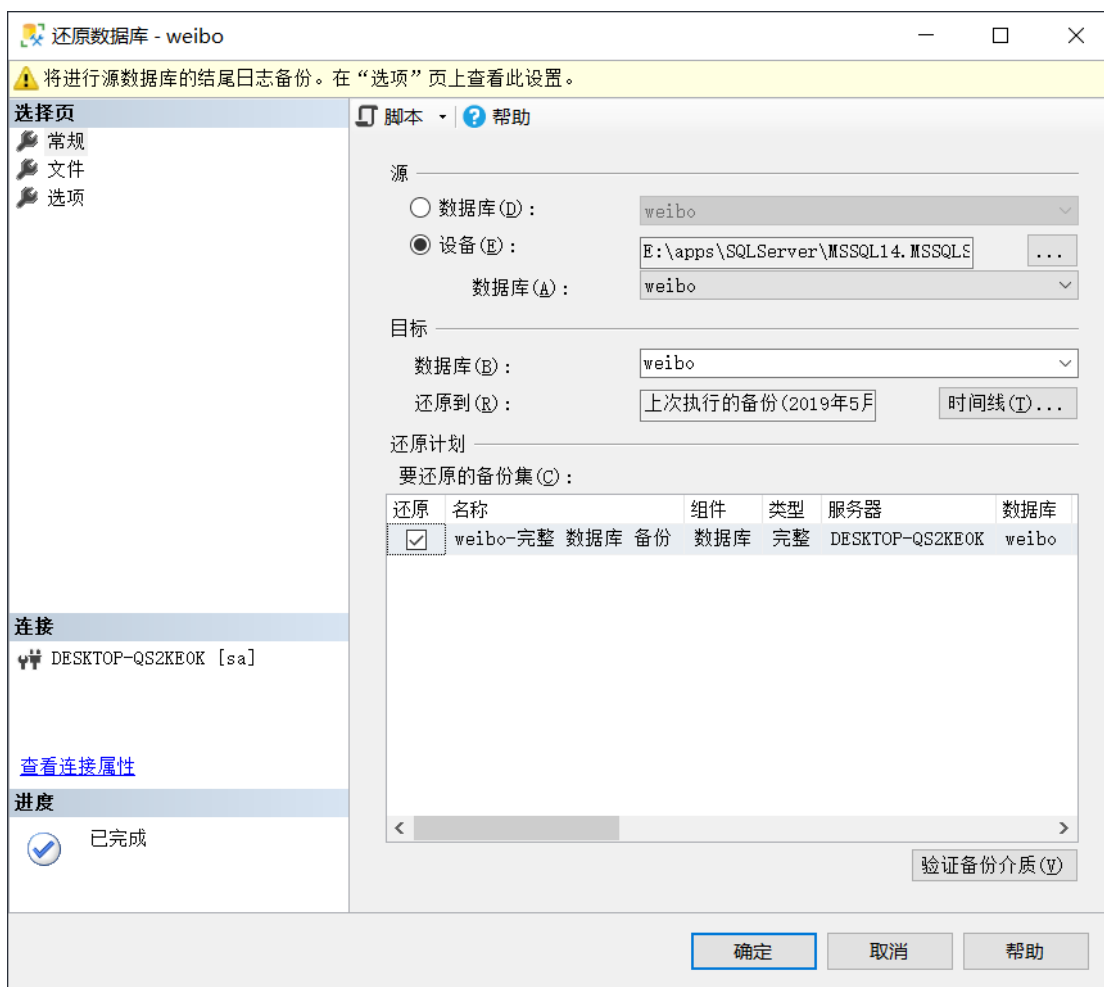


图 2.3 数据库还原

2.2.2 新添加数据库用户

先用 windows 用户登录，然后右键安全性->新建->登录名，然后选择 SQL Server 身份验证，登录名为 kitson，改密码，默认数据库为 lab1，用户映射 √ lab1，确认之后重启服务器，如下图 2.4、2.5 所示：

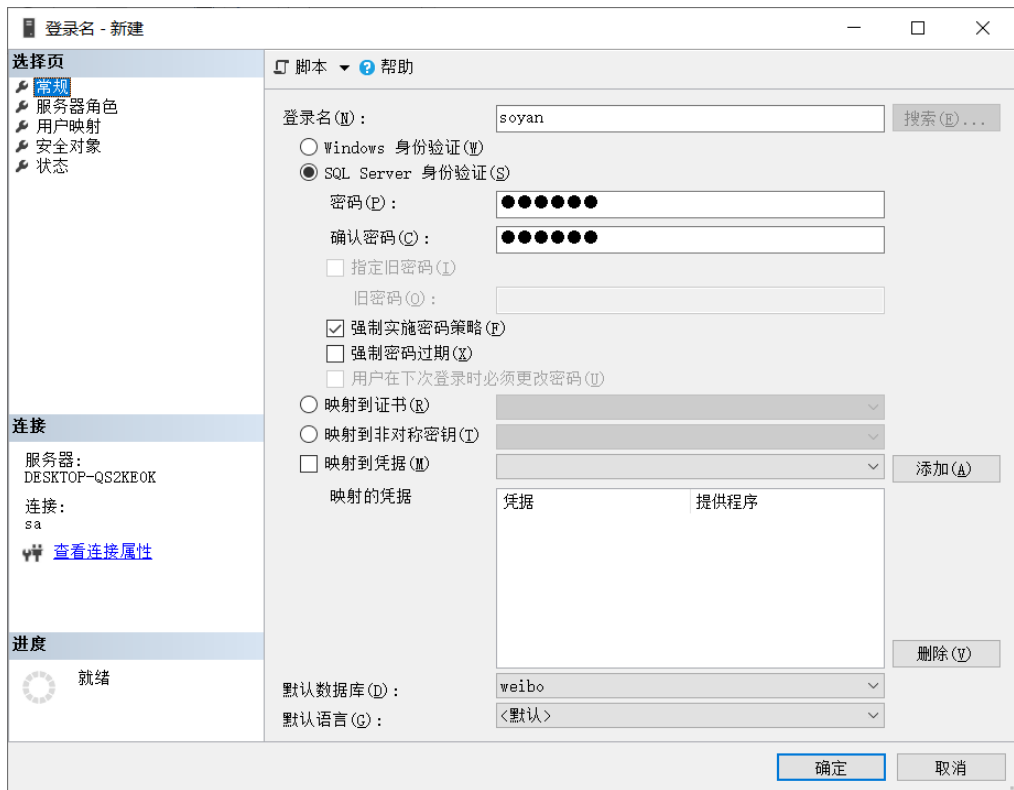


图 2.4 新建用户（1）

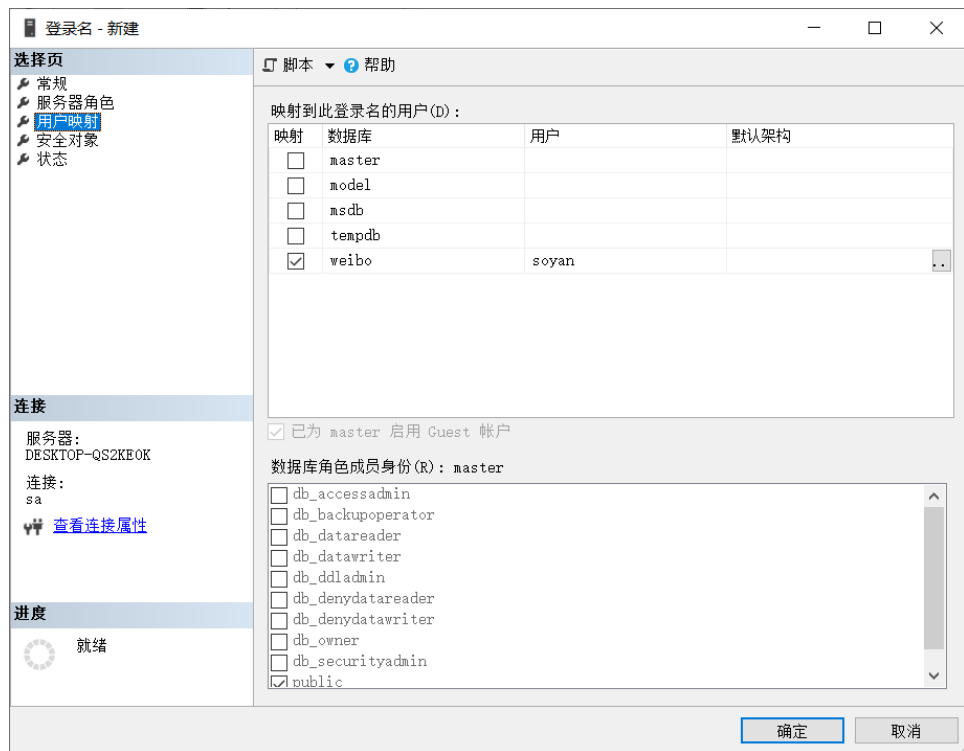


图 2.5 新建用户（2）

重启服务器后，右键服务器，给权限给用户，如图 2.6 所示：

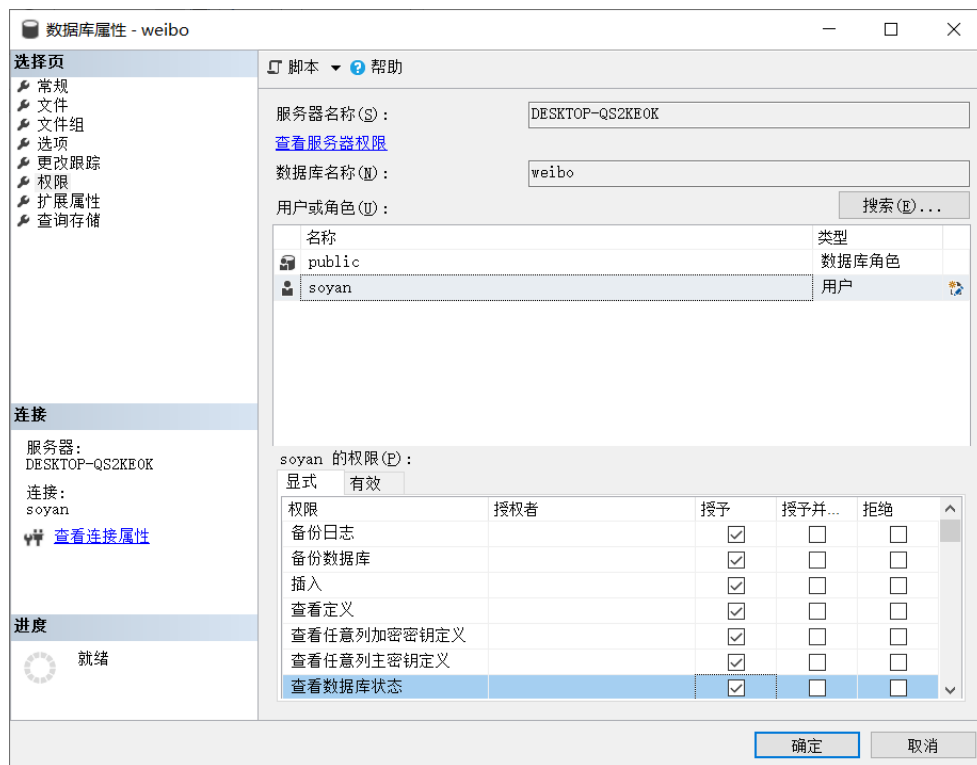


图 2.6 给予权限

之后运行 SQL 语句，检验是否可以使用。

2.3 任务总结

本任务遇到的主要问题是一开始的备份，只会系统备份如何操作，不知道脱机备份是什么。但是通过上完查资料后知道了原来是复制*.ldf 和*_log.ldf 来进行备份，然后在恢复数据库的时候，加载这两个文件就可以了。

3 Sql 练习

3.1 任务要求

1. 建表：根据任务书要求构建相关的关系。
2. 数据更新：学会对内容进行增删改操作，学会数据的导入导出，并且写一个触发器实现对点赞表的完整性控制
3. 查询：分别用一条 SQL 语句完成 17 个小题的要求

3.2 完成过程

3.2.1 建表

①使用 SQL 语句，根据任务书的要求来建立相关的表，每一个表的语句如下：

```
create table USER_INFO
  (UID int primary key,
   NAME char(30),
   SEX char(2),
   BYEAR int,
   CITY char(30)
  );
```

```
create table LABEL
  (LID int primary key,
   LNAME char(30)
  );
```

```
create table MBLOG
  (BID int primary key,
   TITLE char(30),
   UID int,
   PYEAR int,
   PMONTH int,
   PDAY int,
   CONT char(1000),
   foreign key (UID)references USER_INFO(UID)
  );
```

```
create table B_L
  (BID int,
```

```
LID int,  
primary key (BID,LID),  
foreign key (BID)references MBLOG(BID),  
foreign key (LID)references LABEL(LID)  
);
```

```
create table FOLLOW
```

```
(UID int,  
UIDFLED int,  
primary key (UID,UIDFLED),  
foreign key (UID)references USER_INFO(UID),  
foreign key (UIDFLED)references USER_INFO(UID)  
);
```

```
create table FRIENDS
```

```
(UID int,  
FUID int,  
primary key (UID,FUID),  
foreign key (UID)references USER_INFO(UID),  
foreign key (FUID)references USER_INFO(UID)  
);
```

```
create table SUB
```

```
(UID int,  
LID int,  
primary key (UID,LID),  
foreign key (UID)references USER_INFO(UID),  
foreign key (LID)references LABEL(LID)  
);
```

```
create table THUMB
```

```
(UID int,  
BID int,  
primary key (UID,BID),  
foreign key (UID)references USER_INFO(UID),  
foreign key (BID)references MBLOG(BID)  
);
```

```
create table TOPDAY
```

```
(TYEAR int,
```

```

TMONTH int,
TDAY int,
BID int,
TNO int,
primary key (TYEAR,TMONTH,TDAY,BID),
foreign key (BID)references MBLOG(BID)
);

```

(2) 观察性实验

用户在订阅分类时一定要参考被参照关系的主码，按照关系模型的参照完整性，外码必须要参考被参考关系的主码，如果没参照，则会出现 SQL 语句报错，如果参考的不是主码，也会报错。如下图 3.1、3.2 所示：

```

create table sub(
    uids int not null,
    lid int not null,
    foreign key (uids)
    foreign key (lid)
);

```

“附近有语法错误。 应为 REFERENCES.”

图 3.1 缺少参照

```

create table sub(
    uids int not null,
    lid int not null,
    foreign key (uids) REFERENCES users(sex),
    foreign key (lid) REFERENCES label
);

```

被引用表“users”中不存在与外键“FK_sub_0f30b992c08331a77ce4”中的引用列表匹配的主键或候选键。

图 3.2 参照的不是主键

(3) 数据准备：向各个表格录入一定量的数据，如图 3.3：

UID	NAME	SEX	BYEAR	CITY	LID	LNAME	UID	UIDFLED
1	赵一	男	1967	武汉	1	文学	1	2
2	钱二	女	1968	上海	2	艺术	2	1
3	张三	男	1999	南京	3	军事	3	1
4	李四	女	2000	北京	4	历史	3	2
5	王五	男	2001	武汉	5	地理	4	11
6	孙六	女	2001	武汉	6	自然科学	5	1
7	周七	男	2002	武汉	7	工程技术	5	2
8	吴八	男	2002	杭州	8	经济	6	1
9	郑九	男	2003	南京	9	教育	6	3
10	冯十	女	2004	上海	10	哲学	7	1
11	陈十一	男	2011	武汉	11	音乐	7	2
12	诸十二	女	2012	北京	8	NULL	8	1
13	卫十三	女	2011	上海	9	NULL	8	3
14	蒋十四	女	2011	武汉	10	NULL	9	1
* NULL	NULL	NULL	NULL	NULL	11	NULL	10	2
					12	NULL	10	3
					13	NULL	11	3
					14	NULL	11	4
					15	NULL	12	1
					16	NULL	13	1
					17	NULL	14	1
					18	NULL	15	1
					19	NULL	16	1
					20	NULL	17	1
					21	NULL	18	1
					22	NULL	19	1
					23	NULL	20	1
					24	NULL	21	1
					25	NULL	22	1
					26	NULL	23	1
					27	NULL	24	1
					28	NULL	25	1
					29	NULL	26	1
					30	NULL	27	1
					31	NULL	28	1
					32	NULL	29	1
					33	NULL	30	1
					34	NULL	31	1
					35	NULL	32	1
					36	NULL	33	1
					37	NULL	34	1
					38	NULL	35	1
					39	NULL	36	1
					40	NULL	37	1
					41	NULL	38	1
					42	NULL	39	1
					43	NULL	40	1
					44	NULL	41	1
					45	NULL	42	1
					46	NULL	43	1
					47	NULL	44	1
					48	NULL	45	1
					49	NULL	46	1
					50	NULL	47	1
					51	NULL	48	1
					52	NULL	49	1
					53	NULL	50	1
					54	NULL	51	1
					55	NULL	52	1
					56	NULL	53	1
					57	NULL	54	1
					58	NULL	55	1
					59	NULL	56	1
					60	NULL	57	1
					61	NULL	58	1
					62	NULL	59	1
					63	NULL	60	1
					64	NULL	61	1
					65	NULL	62	1
					66	NULL	63	1
					67	NULL	64	1
					68	NULL	65	1
					69	NULL	66	1
					70	NULL	67	1
					71	NULL	68	1
					72	NULL	69	1
					73	NULL	70	1
					74	NULL	71	1
					75	NULL	72	1
					76	NULL	73	1
					77	NULL	74	1
					78	NULL	75	1
					79	NULL	76	1
					80	NULL	77	1
					81	NULL	78	1
					82	NULL	79	1
					83	NULL	80	1
					84	NULL	81	1
					85	NULL	82	1
					86	NULL	83	1
					87	NULL	84	1
					88	NULL	85	1
					89	NULL	86	1
					90	NULL	87	1
					91	NULL	88	1
					92	NULL	89	1
					93	NULL	90	1
					94	NULL	91	1
					95	NULL	92	1
					96	NULL	93	1
					97	NULL	94	1
					98	NULL	95	1
					99	NULL	96	1
					100	NULL	97	1

a. USRE_INFO

b.LABEL

c.FOLLOW

BID	TITLE	UID	PYEAR	PMONTH	PDAY	CONT	BID	LID
1	微博1	1	2019	4	18	“最多地铁...”	1	9
2	微博2	2	2019	4	18	“华中科技大...”	2	9
3	微博3	3	2019	4	18	“最多地铁...”	3	9
4	微博4	4	2019	4	19	“学校校园占...”	4	9
5	微博5	5	2019	4	19	“校学科齐全...”	5	9
6	微博6	6	2019	4	19	“学校实施”...	6	9
7	微博7	7	2019	4	19	“学生”的教...	7	9
8	微博8	8	2019	4	19	“按照”应用...	8	9
9	微博9	9	2019	4	19	“学校坚持”...	9	9
10	微博10	10	2019	4	19	“学校坚持开...	10	9
11	微博11	11	2019	4	19	“《巢》：“...	11	1
12	微博12	12	2019	4	19	“派特·巴克...	12	1
13	微博13	13	2019	4	19	“文学巨匠翻...	13	1

d.MBLOG

UID	FUID
1	7
1	11
1	13
1	14
2	6
2	10
2	11
2	12
2	13
2	14
3	5
3	9
3	11
3	12
4	2
4	8
4	10
4	11
5	1
5	7

UID	LID
1	1
1	4
1	7
2	2
2	5
2	8
3	3
3	6
3	9
4	4
4	7
4	10
5	5
5	8
5	11
6	1
6	6

e.B_L

UID	BID
1	7
1	21
1	35
2	7
2	8
2	22
2	36
3	7
3	9
3	23
3	37
4	7
4	10
4	24
4	38
5	7
5	11
5	25

YEAR	TMONTH	TDAY	BID	TNO
2019	4	18	1	1
2019	4	18	2	2
2019	4	18	3	3
2019	4	19	5	1
2019	4	19	6	2
2019	4	19	7	3
2019	4	19	8	4
2019	4	19	9	5
2019	4	19	10	6
2019	4	19	11	7
2019	4	19	12	8
2019	4	19	13	9
2019	4	19	14	10
2019	4	20	15	1
2019	4	20	16	2
2019	4	20	17	3
2019	4	20	18	4
2019	4	20	19	5

f.FRIENDS

g.SUB

h.THUMB

i.TOPDAY

图 3.3 部分数据

3.2.2 数据更新

(1) 分别用一条 sql 语句完成对博文表基本的增、删、改的操作；

① 对博文进行增操作，代码如下：

insert into MBLOG values(44,'蔡徐坤打篮球',7,2019,5,4,'蔡徐坤打篮球世界第一');查看 mblog 表，发现新项目成功插入，如图 3.4 所示：

43	微博43 ...	2	2019	4	21	西班牙画家...
44	蔡徐坤打篮...	7	2019	5	4	蔡徐坤打篮...
*	NULL	NULL	NULL	NULL	NULL	NULL

图 3.4 增加操作

② 对博文进行改操作，代码如下：

update MBLOG

set PDAY=3

where BID=44;

查看 mblog 表，发现新项目成功修改，如图 3.5 所示：

43	微博43 ...	2	2019	4	21	西班牙画家...
44	蔡徐坤打篮...	7	2019	5	4	蔡徐坤打篮...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 3.5 修改操作

③ 对博文进行删除操作，代码如下：

```
delete
from MBLOG
where BID=44;
```

查看 mblog 表，发现新项目成功删除，如图 3.6 所示：

42	微博42	...	1	2019	4	21	意大利画家...
43	微博43	...	2	2019	4	21	西班牙画家...
NULL	NULL		NULL	NULL	NULL	NULL	NULL

图 3.6 删除操作

（2）批处理操作

将关注 3 号用户的用户信息插入到一个自定义的新表 fan_3 中，代码如下：

```
create table FANS_3
```

```
(UID int primary key,
NAME char(30),
SEX char(2),
BYEAR int,
CITY char(30)
);
```

```
insert into FANS_3
```

```
select *
```

```
from USER_INFO
```

```
where exists
```

```
(select *
```

```
from FOLLOW
```

```
where FOLLOW.UID=USER_INFO.UID and FOLLOW.UIDFLED=3);
```

查看 FAN_3 表，发现插入的是正确的，如下图 3.7 所示：

UID	NAME	SEX	BYEAR	CITY
4	李四	女	2000	北京
6	孙六	女	2001	武汉
8	吴八	男	2002	杭州
10	冯十	女	2004	上海
11	陈十一	男	2011	武汉
NULL	NULL	NULL	NULL	NULL

图 3.7 FAN_3 表内容

（3）数据的导入导出

对着数据库，右键任务->导出数据/导入数据，可以实现数据的导入导出，导出为平面文件，如下图 3.8 所示：

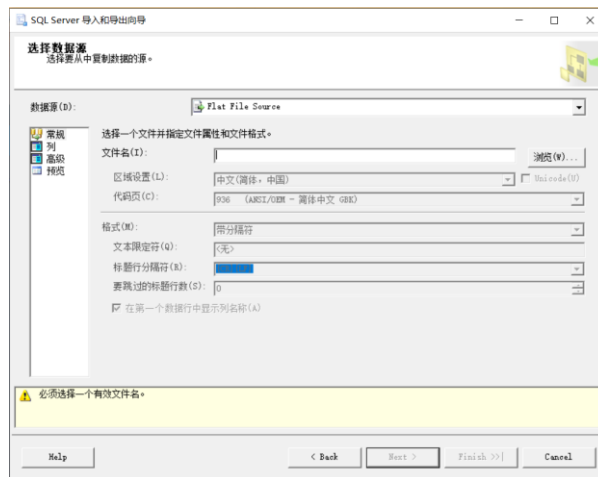


图 3.8 导出数据

(4) 观察性试验

建立一个新的表 new，但未设置主码，代码如下：

```
create table new(
id int,
name char(30)
);
```

插入三条元组后，发现已有元组都能正常地显示出来，但是在可视化视图中无法对已有的元组进行修改。

(5) 触发器实验

代码如下：

```
create trigger use_in_THUMB
on THUMB
after insert
as
begin
    if(not exists(select * from inserted where not exists(select * from MBLOG
where inserted.UID=MBLOG.UID and inserted.BID = inserted.UID)))
        rollback transaction
end
```

然后插入一个自己关注自己的项目，代码如下：

```
insert into THUMB values('1','1');
```

然后给触发器终止了，如下图 3.9 所示：

消息 3609，级别 16，状态 1，第 34 行
事务在触发器中结束。批处理已中止。

图 3.9 触发器终止

3.2.3 查询

查询任务共有 17 小题，每一小题都要用一条 SQL 语句完成下列各小题的需求：

1) 查询“张三”用户关注的所有用户的 ID 号、姓名、性别、出生年份，所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户 ID 号升序排列。

代码如下：

```
select *
from USER_INFO x
where exists
    (select *
     from USER_INFO y,FOLLOW
     where x.UID=FOLLOW.UIDFLED and y.UID=FOLLOW.UID and
y.NAME='张三'
    )
order by BYEAR desc,UID;
```

结果如下图 3.10 所示：

	UID	NAME	SEX	BYEAR	CITY
1	11	陈十一	男	2011	武汉
2	4	李四	女	2000	北京
3	2	钱二	女	1968	上海
4	1	赵一	男	1967	武汉

图 3.10 查询 3 的关注人的信息

2) 查找没有被任何人点赞的博文 ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

代码如下：

```
select BID,TITLE,NAME
from MBLOG,USER_INFO
where MBLOG.UID=USER_INFO.UID and BID not in
    (select distinct BID
     from THUMB
    )
order by TITLE;
```

结果如下图 3.11 所示：

	BID	TITLE	NAME
1	1	微博1	赵一
2	2	微博2	钱二
3	3	微博3	张三
4	4	微博4	李四
5	5	微博5	王五
6	6	微博6	孙六

图 3.11 查询没有被点赞过的微博

3) 查找 2000 年以后出生的武汉市用户发表的进入过头条的博文 ID;

代码如下:

```
select distinct MBLOG.BID
```

```
from MBLOG,USER_INFO,TOPDAY
```

```
where MBLOG.UID=USER_INFO.UID and BYEAR>2000 and CITY='武
```

```
汉' and MBLOG.BID in
```

```
(select distinct BID
```

```
from TOPDAY);
```

结果如下图 3.13 所示:

	BID
1	5
2	6
3	7
4	11
5	14
6	17
7	18
8	20
9	24
10	28
11	29
12	30
13	31
14	36
15	37

图 3.13 查询 3) 结果

4) 查找订阅了所有分类的用户 ID;

代码如下:

```
select distinct UID
```

```
from USER_INFO
```

```
where not exists
```

```
(select *
```

```

from LABEL
where not exists
(select *
from SUB
where SUB.UID=USER_INFO.UID and SUB.LID=LABEL.LID
)
);

```

结果如下图 3.14 所示：

	UID
1	14

图 3.14 查询订阅所有分类的用户

5) 查找出生年份小于 1970 年或者大于 2010 年的用户 ID、出生年份、所在城市，要求 where 子句中只能有一个条件表达式；

代码如下：

```

select UID,BYEAR,CITY
from USER_INFO
where BYEAR not between 1970 and 2010;

```

结果如下图 3.15 所示：

	UID	BYEAR	CITY
1	1	1967	武汉
2	2	1968	上海
3	11	2011	武汉
4	12	2012	北京
5	13	2011	上海
6	14	2011	武汉

图 3.15 小于 1970 年或者大于 2010 年的用户

6) 统计每个城市的用户数；

代码如下：

```

select CITY,COUNT(UID)
from USER_INFO
group by CITY;

```

结果如下图 3.16 所示：

	CITY	(无列名)
1	北京	2
2	杭州	1
3	南京	2
4	上海	3
5	武汉	6

图 3.16 各城市的用户数

7) 统计每个城市的每个出生年份的用户数, 并将结果按照城市的升序排列, 同一个城市按照出生用户数的降序排列其相应的年份;

代码如下:

```
select CITY,BYEAR,COUNT(UID)
from USER_INFO
group by CITY,BYEAR
order by CITY desc,COUNT(UID)
```

结果如下图 3.17 所示:

	CITY	BYEAR	(无列名)
1	武汉	1967	1
2	武汉	2002	1
3	武汉	2001	2
4	武汉	2011	2
5	上海	2004	1
6	上海	2011	1
7	上海	1968	1
8	南京	1999	1
9	南京	2003	1
10	杭州	2002	1
11	北京	2000	1
12	北京	2012	1

图 3.17 统计结果

8) 查找被点赞数超过 10 的博文 ID 号;

代码如下:

```
select BID
from THUMB
group by BID
having COUNT(UID)>10;
```

结果如下图 3.18 所示:

	BID
1	7

图 3.18 点赞数超过 10 的微博

9) 查找被 2000 年后出生的用户点赞数超过 10 的博文 ID 号;

代码如下:

```
select BID
from THUMB,USER_INFO
```

where THUMB.UID=USER_INFO.UID and USER_INFO.BYEAR >= 2000
group by BID
having COUNT(THUMB.UID)>10;
结果如下图 3.19 所示:

	BID
1	7

图 3.19 查询 9) 结果

10) 查找被 2000 年后出生的用户点赞数超过 10 的每篇博文的进入头条的次数;

代码如下:

```
select BID,COUNT(*)
from TOPDAY
where BID in
(select BID
from THUMB,USER_INFO
where THUMB.UID=USER_INFO.UID and USER_INFO.BYEAR >=
2000
group by BID
having COUNT(THUMB.UID)>10)
group by BID;
```

结果如下图 3.20 所示:

	BID	(无列名)
1	7	1

图 3.20 查询 10) 结果

11) 查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户 ID, 要求不能使用嵌套查询, 且 where 子句中最多只能包含两个条件

代码如下:

```
select distinct UID
from SUB,LABEL
where SUB.LID=LABEL.LID and LABEL.LNAME in ('文学','艺术','哲学','
音乐');
```

结果如下图 3.21 所示:

	UID
1	1
2	2
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14

图 3.21 查询 11) 结果

12) 查找标题中包含了“最多地铁站”和“_华中科技大学”两个词的博文基本信息;

代码如下:

```
select *
from MBLOG
where CONT like '%最多地铁站%' and CONT like '%_华中科技大学%'
escape '\';
```

结果如下图 3.22 所示:

	BID	TITLE	UID	PYEAR	PMONTH	PDAY	CONT
1	1	微博1	1	2019	4	18	“最多地铁站”和“_华中科技大学”两个词
2	3	微博3	3	2019	4	18	“最多地铁站”和“_华中科技大学”两个词

图 3.22 查询 13) 结果

13) 查找所有相互关注的用户对的两个 ID 号, 要求不能使用嵌套查询;

代码如下:

```
select x.UID,x.UIDFLED
from FOLLOW x,FOLLOW y
where x.UID=y.UIDFLED and x.UIDFLED=y.UID and
x.UID<x.UIDFLED;
```

结果如下图 3.23 所示:

	UID	UIDFLED
1	1	2
2	3	4
3	3	11

图 3.23 查询相互关注的两个 id 号

14) 查找好友圈包含了 5 号用户好友圈的用户 ID。

代码如下：

```
select UID
from FRIENDS x
where not exists
(select *
from FRIENDS y
where (y.FUID=5 or y.UID=5) and
not exists
(select *
from FRIENDS z
where (z.UID=x.UID and z.FUID=y.FUID) or (z.FUID=x.FUID and
z.UID=y.UID)));
```

结果如下图 3.24 所示：

	UID
1	3

图 3.24 包含了 5 号用户好友圈的用户 ID

15) 查找 2019 年 4 月 20 日每一篇头条博主的 ID 号、标题以及该博主的每一个分类 ID，要求即使该博文没有任何分类 ID 也要输出其 ID 号、标题；

代码如下：

```
select MBLOG.BID,TITLE,LID
from MBLOG,B_L
where MBLOG.BID=B_L.BID and MBLOG.BID in
(select BID
from TOPDAY
where TYEAR=2019 and TMONTH=4 and TDAY=20)
union
select MBLOG.BID,TITLE,null
from MBLOG
where MBLOG.BID not in
(select BID
from B_L)
and MBLOG.BID in
```



```
(select BID
from TOPDAY
where TYEAR=2019 and TMONTH=4 and TDAY=20);
```

结果如下图 3.25 所示：

	BID	TITLE	LID
1	15	微博15	3
2	16	微博16	3
3	17	微博17	3
4	18	微博18	3
5	19	微博19	4
6	20	微博20	4
7	21	微博21	6
8	22	微博22	6
9	23	微博23	6

图 3.25 查询 15) 结果

16) 查找至少有 3 名共同好友的所有用户对的两个 ID 号。

代码如下：

```
select x.UID,y.UID
from USER_INFO x,USER_INFO y
where x.UID<y.UID and 2<
(select COUNT(*)
from USER_INFO z
where x.UID in
((select UID
from FRIENDS
where FRIENDS.FUID=z.UID)
union
(select FUID
from FRIENDS
where FRIENDS.UID=z.UID)))
and y.UID in
((select UID
from FRIENDS
where FRIENDS.FUID=z.UID)
union
(select FUID
from FRIENDS
where FRIENDS.UID=z.UID))));
```

结果如下图 3.26 所示：

	UID	UID
1	1	2
2	2	8

图 3.26 至少有 3 名共同好友的所有用户对的两个 ID 号

17) 创建视图：查阅 DBMS 内部函数，创建一个显示当日热度排名前十的微博信息的视图，其中的属性包括：博文 ID、博文标题、发表者 ID、发表者姓名、被点赞数。

代码如下：

```
create view TOPTODAY
as
select
TOPDAY.BID,MBLOG.TITLE,USER_INFO.UID,USER_INFO.NAME,COUNT
(*) as NUM
from TOPDAY,MBLOG,USER_INFO,THUMB
where TOPDAY.TYEAR = YEAR(GETDATE())
and TOPDAY.TMONTH = MONTH(GETDATE())
and TOPDAY.TDAY = DAY(GETDATE())
and TOPDAY.BID = MBLOG.BID
and MBLOG.UID = USER_INFO.UID
and THUMB.BID = MBLOG.UID
group
TOPDAY.BID,MBLOG.TITLE,USER_INFO.UID,USER_INFO.NAME;
```

结果如下图 3.27 所示：

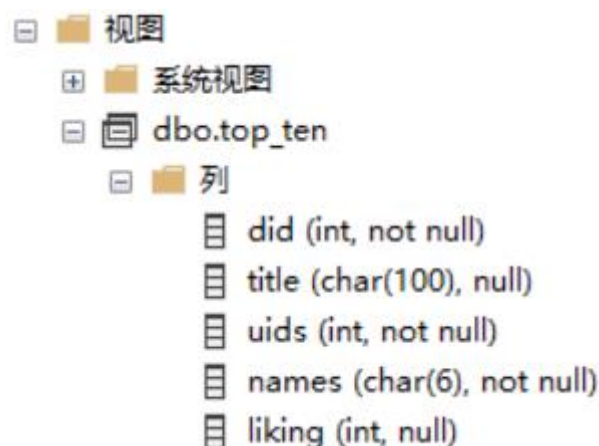


图 3.37 当日排名热度前 10 微博视图

3.3 任务总结

通过本次任务，我对 SQL 语言的应用有了一个更加深刻的了解。在完成 17 条 sql 查询语句的任务的时候遇到的困难是最多的，因为一开始对 SQL 语句

的不熟悉，但是经过不断地看书，或者和同学一起讨论，然后总结出了一些查找的经验，比如查询的时候，要注意一步一步地筛选，先筛选哪一部分，再筛选哪一部分。然后还要对 SQL 的一些操作熟悉，比如升序降序、group by 等等，还有几个聚集函数 count 等，这些都要运用好。

第二个难点就是数据内容的设计，要好好地设计数据的内容以便于对 17 条语句进行检验。有一些语句需要内容比较全面，才能测试得完整，这样比较繁琐。

第三个难点就是使用触发器。因为之前没有接触过触发器，要在网上和书上找很多资料，然后对触发器有所了解后再进行编写 SQL 的代码。

总的来说这一次的任务对我来说是有很大的作用，对我完成下一个综合性任务有着很大的意义，让我熟练了各种 SQL 的查询，为下一个任务打下结实的基础！

4 综合实践任务

4.1 系统设计目标

(1) 应用背景：

传统的用手写的方式录入、管理图书信息和借阅信息的图书馆管理系统在图书馆藏书量与读者数量都很大的情况下无法满足需求，必须借助图书管理系统来完成这些工作。

(2) 总体目标：

系统的使用者分为管理员和读者。管理员负责新书的录入、旧书的清理、借书审核、还书审核，以及日常的维护，有权限增删图书的信息；读者要能够根据书名查找图书，并选择中意的图书进行借阅，读完以后还可以还书；读者没有权限修改图书的信息，他对图书的操作仅限于借、还。这就是一个简单的图书管理系统的大致功能。

4.2 需求分析

(1) 系统总体功能需求列表如表 4.1 所示。

表 4.1 系统总体需求

编号	功能名称	功能描述	操作者
1	录入图书	将新的书籍信息录入系统	管理员
2	删除图书	将旧的或损毁的图书移出系统	管理员
3	修改图书	修改图书信息	管理员
4	审核还书	审核还书请求，检查图书开出罚单	管理员
5	审核借书	审核读者能否借书，并操作	管理员
6	读者信息维护	录入修改注销读者信息	管理员
7	查找书籍	通过书名查找书籍信息	读者
8	查询个人信息	查看个人信息	读者
9	查询借书记录	查看借书记录	读者
10	续借	延长还书时间	读者
11	交款	缴纳因没有及时还书造成的罚金	读者

(2) 性能需求，能满足同一本书被借时不会出现错误。

(3) 数据完整性，具体在数据库设计中说明。

(4) 数据流图，如图 4.1 所示。

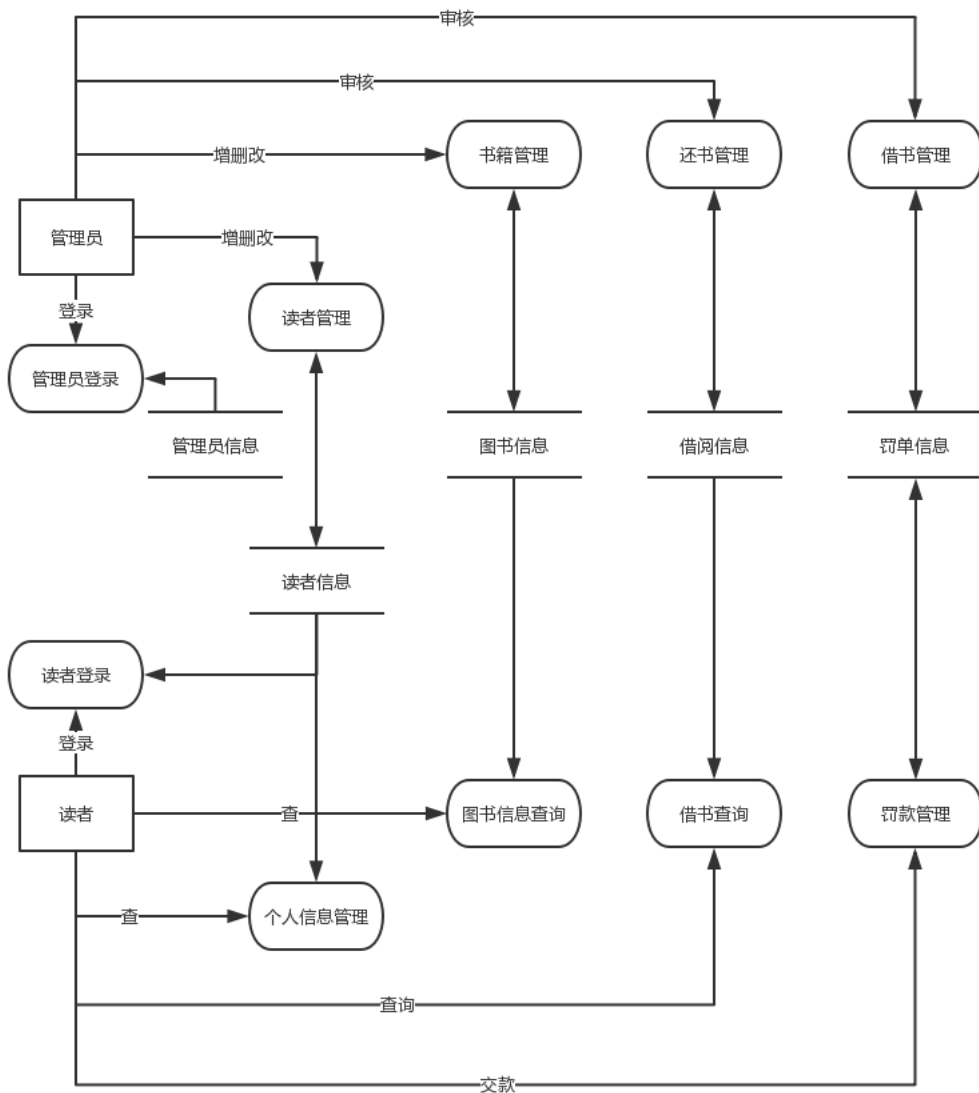


图 4.1 图书管理系统数据流图

4.3 总体设计

(1) B/S 架构：

总体而言，系统为 B/S 结构包含 3 个部分：数据库、web 服务端、浏览器。只有服务端能够直接访问数据库，而浏览器只能通过与服务端的沟通来间接的访问数据库，从而达到较高的安全性。架构图如图 4.2。

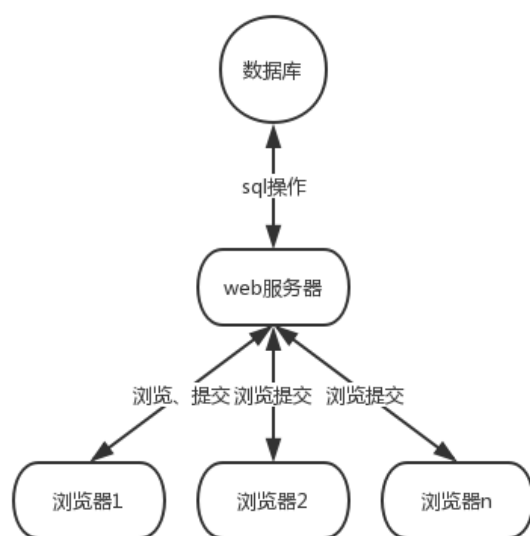


图 4.2 B/S 架构图

(2) 系统总体结构:

系统分为两大功能板块：管理员和读者，总体结构图如图 4.3 所示。

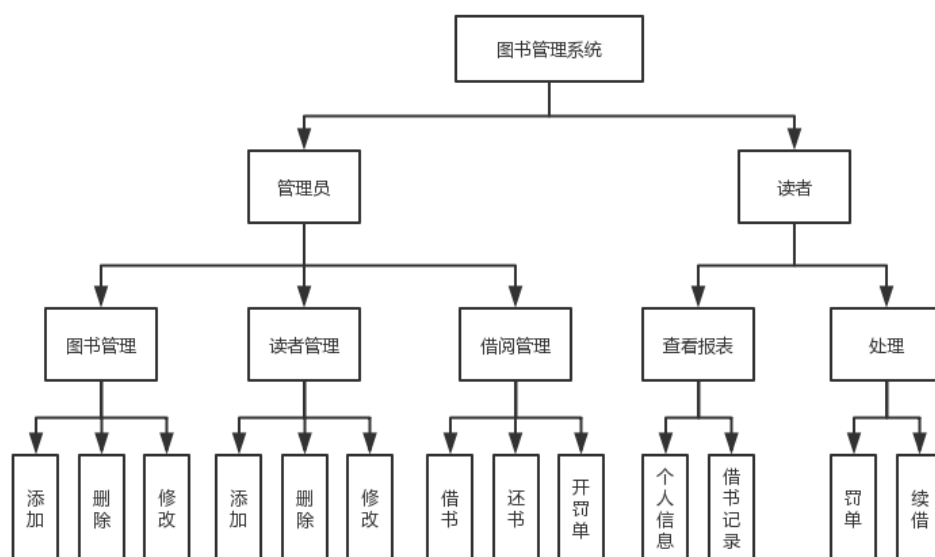


图 4.3 总体设计图

(3) 功能设计:

管理员

1. 图书信息管理

具体分为两个部分：新书上架，也就是将新书的各项信息录入到数据库；查询维护，通过索书号、书名、作者等信息进行查询，得到查询结果后可以查看信

息或者删除图书。

2. 读者信息管理

与图书信息管理类似，可以录入、查找并修改或注销读者信息。

3. 借阅管理

分为借书和还书两部分。借书时，管理员输入读者 ID 查看有无借书权限，若无结束；若有，登记图书 ID 完成结束。还书时，管理员输入图书 ID 并查看图书有无损坏以及有无逾期，并开出相应罚单。

读者

1. 信息查看

可以查看自己的个人信息以及借书记录。

2. 处理

可以续借图书，以及处理罚单。

4.4 数据库设计

(1) 功能需求转化为 E-R 图，如图 4.4。

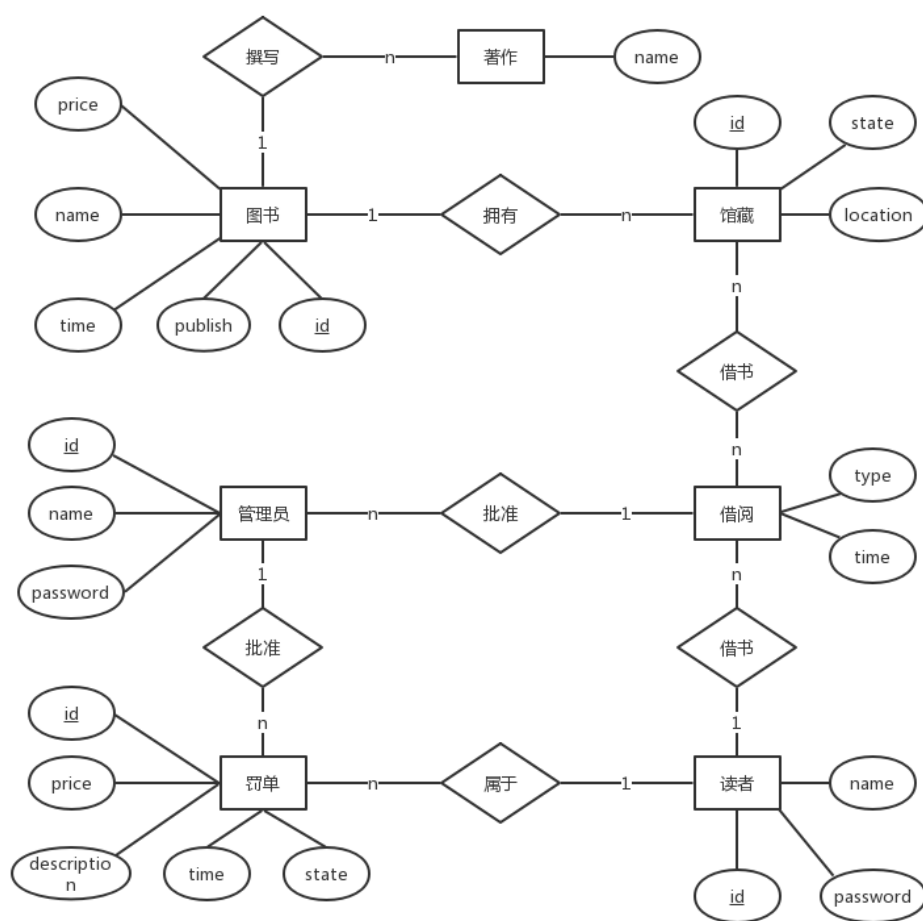


图 4.4 E-R 图

(2) 权限控制

web 服务器会根据当前用户的身份对相应操作进行控制。

(3) 数据库逻辑结构设计

数据库主要分成三大部分：图书信息、借阅信息和用户信息，其中借阅信息可用一个表实现。

对于用户信息，由于要实现不同的用户登录并赋予他们不同的权限，可以使用 django 内置的 User 与 Group 模型完成，创建两个用户组分别是读者与管理员，赋予两个组相应的权限，创建用户时，将用户添加到相应的组即可。

对于图书信息，每本书有一个图书编号。每本书对应多个实体，使用实体编号区分图书实体，对实体还要增加馆藏地点、馆藏状态。

同样的，一本书也可能有多个作者，所以也需要用一个单独的表来存放作者，使用图书编号到作者的映射实现。

借阅信息用于存放读者的借阅记录，由于借阅过程存在罚款，所以还有一个罚款表。

最终图书管理系统的数据库共有 7 个表：图书信息表、著作表、馆藏信息表、借阅信息表、罚款信息表、读者信息表、管理员信息表和罚单表，表项分别如下，主码使用下划线标注，外码使用斜体标注：

1. 图书信息表： 图书编号，书名，出版社，出版时间，价格；
2. 馆藏信息表： 实体编号， *图书编号*，馆藏地点，状态；
3. 著作表： 图书编号，作者；
4. 借阅信息表： 借书编号， *读者编号*， *审核管理员号*， *实体编号*，时间，操作类型；
5. 读者信息表： 编号，姓名，密码；
6. 管理员信息表： 编号，姓名，密码；
7. 罚款信息表： 罚款编号， *管理员编号*， *读者编号*，金额，时间，描述，状态。

(4) 数据库物理设计

图书表设计如表 4.2：

表 4.2 图书表

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
NAME	VARCHAR(150)			书籍名
PUBLISH	VARCHAR(150)			出版社
TIME	INTEGER		正整数	出版年份

PRICE	DECIMAL		正数	书籍价格
-------	---------	--	----	------

馆藏信息设计如表 4.3:

表 4.3 馆藏信息

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
BOOK_ID	INTEGER	外键		书籍外键
LOCATION	VARCHAR(150)			馆藏地点
STATE	SMALLINT		0,1,2	0 借出, 1 在馆, 2 损毁

著作表设计如表 4.4:

表 4.4 著作表

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
BOOK_ID	INTEGER			书籍外键（与作者名的组和唯一）
WRITER	VARCHAR(150)			作者名

用户表采用 django 内置模型 User，结构如表 4.5:

表 4.5 用户表

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
USERNAME	VARCHAR(150)			用户名
PASSWORD	VARCHAR(128)			密码
FIRSTNAME	VARCHAR(30)			名
LASTNAME	VARCHAR(150)			姓
EMAIL	VARCHAR(254)			邮箱

借阅表设计如表 4.7:

表 4.7 借阅表

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
STORE_ID	INTEGER	外键		馆藏外键
READER_ID	INTEGER	外键		用户外键
TIME	DATETIME			记录时间
TYPE	SMALLINT		0,1,2	0 借出, 1 归还, 2 续借

罚单表设计如表 4.8:

表 4.8 罚单表

字段	类型	主键/外键	取值	备注
ID	INTEGER	主键		自增主键
READER_ID	INTEGER	外键		用户外键
PRICE	DECIMAL			金额
DESCRIPTION	VARCHAR(150)			描述
TIME	DATETIME			时间
STATE	SMALLINT		0,1	0 未完成, 1 完成

4.5 详细设计与实现

系统使用 Python 语言实现, 采用 B-S 架构, 使用了 Python 的 Django 框架, 数据库使用了其默认的 sqlite3, 由于其非常轻量、基于单个文件且不需要登录, 适合作为开发测试使用。界面设计使用了 Bootstrap 库以及相应的设计器设计而成, 还使用了一些 js 辅助实现。

Django 数据库设计主要在应用目录下的 models.py, 定义了每张表的内容。

url.py 主要负责 url 地址到视图函数的映射;

views.py 主要是具体的视图函数, 也就是功能函数。

(1) 用户登录登出

使用 Django 内置登录登出视图, url 匹配规则如下:

```
path('login/', LoginView.as_view(template_name='login.html'), name='login')
```

```
path('logout/', auth_views.logout_then_login, name='logout')
```

登录界面包括用户名输入框、密码输入框以及登录按钮, 错误提示包含在 html 模板中, 发生错误时将显示, 分别为用户名密码不匹配, 用户权限不足以访问页面被重定向, 用户未登录被重定向:

```

{% if form.errors %}
<p>Your username and password didn't match. Please try again.</p>
{% endif %}
{% if next %}
{% if user.is_authenticated %}
<p>Your account doesn't have access to this page. To proceed, please login with
an account that has access.</p>
{% else %}
<p>Please login to see this page.</p>
{% endif %}
{% endif %}

```

成功登陆后 url 被重定向到/is_admin/, 匹配视图 is_admin_view(), 此视图判断用户属于哪个用户组, 再对读者重定向到读者首页/, 对管理员重定向到管理员首页/administrator/。

用户登出后重定向到登录页面/login/

(2) 查询设计

本项目设计的查询主要包括读者查询书籍, 管理员查询书籍, 管理员查询读者, 设计思路基本一致, 只是涉及的表不同, 给出的链接不同, 下面以管理员查询书籍为例, 视图方法如下。

```

def books_manage_view(request):
    context = {"is_empty": False, "books": None}
    if 'name' in request.GET:
        name = request.GET['name']
        books = Book.objects.filter(name__contains=name).order_by('-time')
        context['books'] = books
        if books.count() == 0:
            context['is_empty'] = True
    return render(request, 'books_manage.html', context)

```

首先定义了一个字典 context, "is_empty"字段用于说明有无查询结果, "books"字段用于放置查询结果;

然后, 如果有查询的话, 首先从 GET 方法中获取查询内容, 然后使用 django 的模型查询函数, 查找书名中包含查询内容的对象。

然后处理包装字典, 并返回 html 模板的填充, 模板部分内容如下:

```

{% if 'name' in request.GET %}
<div class="py-4">
    <div class="container">
        <div class="row">
            <div class="col-12">

```

```

<h2>搜索结果</h2>
<hr>
{% if is_empty %}
<p class="lead">搜索无结果</p>
{% endif %}
{% for book in books %}
<p class="lead">
    <a href="{% url 'book_manage' book.id %}">
        {{ book.name }}
    </a>
    <a class="btn btn-outline-primary ml-4" href="{% url
'delete_book' book.id %}">删除</a>
</p>
<p>
    作者:
    {% for write in book.write_set.all %}
        {{ write.writer }}
    {% endfor %}
    出版社: {{ book.publish }} 出版时间: {{ book.time }}
</p>
{% endfor %}
</div>
</div>
</div>
{% endif %}

```

首先{% if is_empty %}根据 context 中的"is_empty"字段判断查询是否有结果，如果无结果显示“查询无结果”的提示。然后{% for book in books %}循环"books"字段中的内容。

```

<p class="lead">
    <a href="{% url 'book_manage' book.id %}">
        {{ book.name }}
    </a>
    <a class="btn btn-outline-primary ml-4" href="{% url 'delete_book'
book.id %}">删除</a>
</p>

```

显示了一个显示为书籍名称，指向书籍具体内容的 url 的链接，以及一个包含删除这书籍链接的按钮。

```

<p>

```

作者:

```
{% for write in book.write_set.all %}
{{ write.writer }}
{% endfor %}
```

出版社: {{ book.publish }} 出版时间: {{ book.time }}

</p>

循环显示了所有的书籍作者，显示了书籍出版社以及出版时间信息。

(3) 修改设计

本项目设计的修改主要包括修改书籍、修改读者、修改作者、修改藏书、修改密码，设计思路基本一致，只是涉及的表不同，下面以修改书籍为例，url 规则如下：

```
path('administrator/books_manage/edit/<int:pk>/',views.edit_book_view)
```

视图方法如下。

```
def edit_book_view(request, pk):
    if request.method == 'POST':
        try:
            book = Book.objects.get(id=pk)
            form = forms.BookForm(request.POST, instance=book)
        except:
            messages.error(request, '书籍 id 不存在')
        else:
            if form.is_valid():
                try:
                    form.save()
                except:
                    messages.error(request, '已有属性完全一致的书籍')
                else:
                    messages.success(request, '修改成功')
            else:
                messages.error(request, '此 url 不能使用 post 以外方法')
    return redirect('book_manage', pk=pk)
```

此方法先从 url 中提取待修改的书籍 id，再从 POST 方法中接收修改后的书籍名称，出版社，出版时间和价格信息：

```
book = Book.objects.get(id=pk)
form = forms.BookForm(request.POST, instance=book)
```

这里使用了一个自定义的表单类 forms.BookForm 接收信息，使用参数 instance=book 表明此表单是对已有的 book 对象的替换。如果表单信息有效，尝试使用 form.save()将修改写入数据库，如果发生异常，则说明已经存在信息相同

的书籍，无法修改，并提示；没有异常则提示修改成功，并重定向到书籍信息网址。

（4）新建设计

本项目设计的新建主要包括新建书籍、新建读者、新建作者、新建藏书，设计思路基本一致，只是涉及的表不同，下面以管理员新建书籍为例，url 规则如下：

```
path('administrator/books_manage/new/',views.new_book_view)
```

视图方法如下：

```
def new_book_view(request):
    if request.method == 'POST':
        form = forms.BookForm(request.POST)
        if form.is_valid():
            try:
                book = form.save(commit=False)
                book.save()
            except:
                messages.error(request, '已有属性完全一致的书籍')
            else:
                messages.success(request, '新建成功')
                return redirect('book_manage',pk=book.id)
        else:
            form = forms.BookForm()
    return render(request, 'new_book.html', {'form': form})
```

方法判断 http 方法是 get 还是 post 如果是 get 就返回一个空表单；如果是 post 则使用 form 接收信息，这与上面的修改类似，只是因为新建，没有使用 instance=book 表明此表单是对已有的 book 对象的替换。

（5）删除设计

本项目设计的新建主要包括删除书籍、删除读者、删除作者、删除藏书，设计思路基本一致，只是涉及的表不同，给出的链接不同，下面以管理员新建书籍为例，url 规则如下：

```
path('administrator/books_manage/edit/<int:pk>',views.edit_book_view)
```

视图方法如下：

```
def delete_book_view(request, pk):
    try:
        book = Book.objects.get(id=pk)
    except:
        messages.error(request, '书籍 id 不存在')
    else:
```

```
book.delete()
messages.success(request, '删除成功')
return redirect('books_manage')
```

首先根据 url 中提供的书籍 id 查找对应书籍, 如果没找到, 提示书籍不存在, 否则使用 `book.delete()` 删除书籍, 并提示删除成功, 重定向到书籍搜索界面。

(6) 借书设计

url 规则如下:

```
path('administrator/borrow_manage/', views.borrow_manage_view)
```

视图方法过长, 不予列出, 以流程图的方式展出

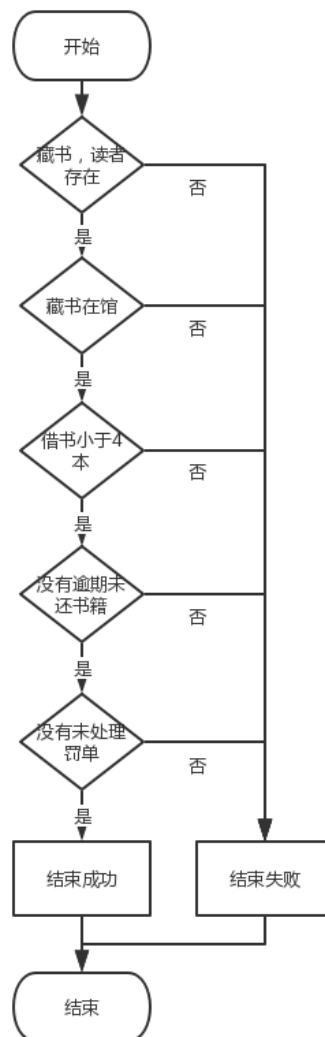


图 4.5 借书流程图

首先使用模型查找方法根据 POST 方法提交的藏书 id 与用户 id 找到相应对象, 查找失败, 发出提示。

然后查看书籍是否处于 1 状态 (在馆), 如果不是, 给出错误信息。

然后根据用户的借书记录与罚单记录，判断其是否能借书，遍历此用户的罚单，有 0 状态（未处理）不能借书；使用定义的 `get_loaned()` 方法，找出所有未还的书籍，如果数量等于 4，不能借书；如果存在逾期书籍不能借书，并给出提示；否则，插入一条借书记录，并提示借书成功。

（7）还书设计

url 规则如下：

`path('administrator/return_manage/', views.return_manage_view)`

视图方法过长，不予列出，以流程图的方式展出

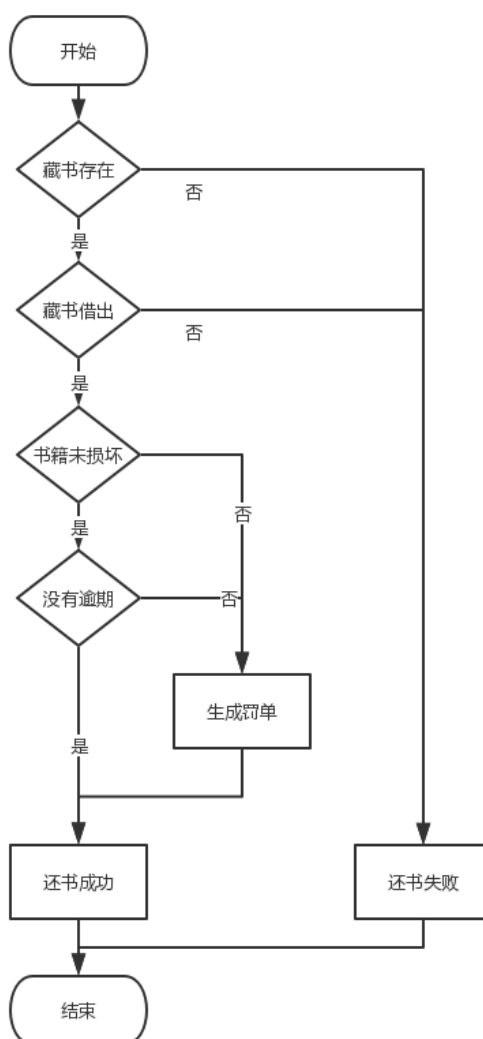


图 4.6 还书流程图

首先使用模型查找方法根据 POST 方法提交的藏书 id 找到相应对象，查找失败，发出提示。

然后查看书籍是否处于 0 状态（借出），如果不是，给出错误信息。

然后根据用户的借书是否逾期与书籍是否损坏，判断生成罚单。书籍损坏生

成一个等于书籍价格的罚单；借书逾期，每逾期一天罚单增加一元，最高一百元，插入一条还书记录，并提示还书成功。

(8) 续借设计

url 规则如下：

```
path('loaned/renew/<int:pk>', views.renew_view, name='renew')
```

视图方法如下：

```
def renew_view(request, pk):
    try:
        loan = Loan.objects.get(id=pk, reader__id=request.user.id)
    except:
        messages.error(request, '借书单号无效')
    else:
        out_date = (timezone.now()-loan.time).days > 30
        wrong_loan = Loan.objects.filter(
            store__id=loan.store.id).order_by('-time')[0].id != loan.id
        if loan.state != 0 or out_date or wrong_loan:
            messages.error(request, '无法续借')
        else:
            renew = Loan(time=timezone.now(), state=2,
                          store=loan.store, reader=loan.reader)
            renew.save()
            messages.success(request, '续借成功')
    return redirect('loaned')
```

首先查找借书记录对象，查找失败，发出提示。

如果书籍以逾期，或书籍没有被此用户借出或已被续借，则无法续借；否则续借成功，插入一条续借记录。

(9) 处理罚单设计

(10) url 规则如下：

```
path('tickets/handle/<int:pk>/', views.ticket_handle_view, name='ticket_handle')
```

视图方法如下：

```
def ticket_handle_view(request, pk):
    try:
        ticket = Ticket.objects.get(id=pk, reader__id=request.user.id)
    except:
        messages.error(request, '罚单号无效')
    else:
        if ticket.state != 0:
            messages.error(request, '罚单已完成')
```

```
else:
    ticket.state = 1
    ticket.save()
    messages.success(request, '操作成功')
```

```
return redirect('tickets')
```

找到罚单号对应罚单，且罚单的主人为当前用户，若未找到，提示错误。

然后查看罚单状态是否为 0（未完成），否则提示错误。

然后修改罚单状态为 1，使用 ticket.save()写入数据库，提示操作成功

（11）权限与登录约束

本项目有较为完备的权限与登录约束设计，每一个视图函数有有约束。

Django 中通过在每个视图函数前添加@login_required 修饰器来限制视图函数必须登录后才能访问；通过添加@permission_required('some_permission')修饰器来限制视图函数必须拥有相应权限才能使用，权限具体来说就是对每个表的增删改查，在表被新建时由系统定义当不满足条件时，网页会被重定向到登录页面。

（12）界面设计

主要使用了 Bootstrap 库，由于使用的是 cdn 提供，测试项目时必须联网。

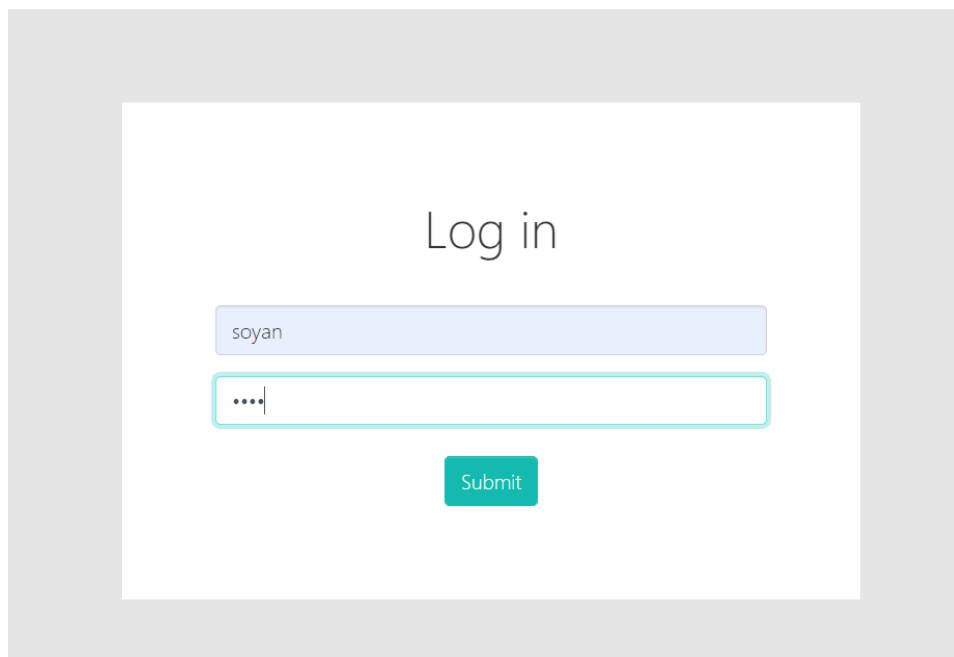
由于界面设计不涉及数据库操作，也就不是重点内容，具体请见项目。

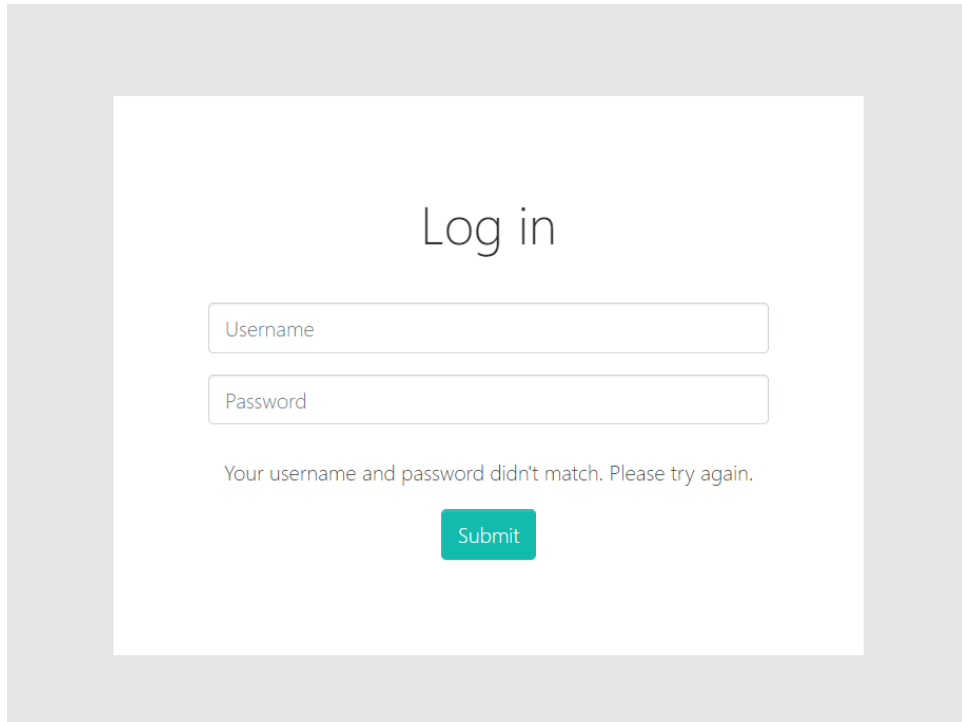
4.6 系统测试

（1）登录测试

a) 错误密码测试

输入用户名与错误密码：



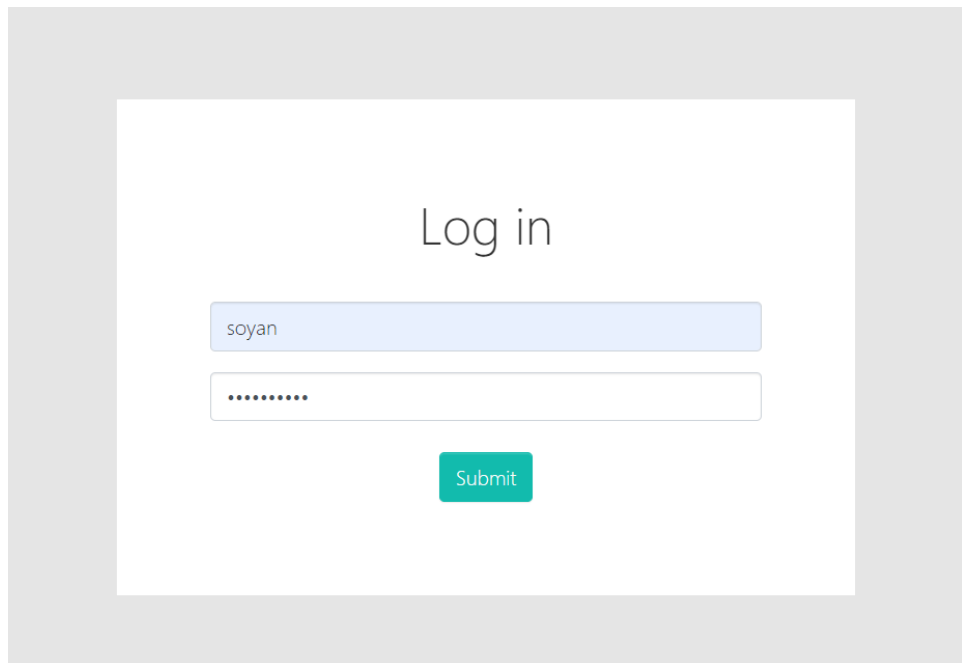


The image shows a login form titled "Log in". It has two input fields: "Username" and "Password". Below the fields is an error message: "Your username and password didn't match. Please try again." and a green "Submit" button.

图 4.7 输入错误密码

登陆界面给出了错误提示。

b) 正确密码测试:



The image shows the same login form titled "Log in". The "Username" field now contains the text "soyan". The "Password" field is masked with dots. The green "Submit" button is still present.



图 4.8 输入正确密码

成功进入用户主页，可看见右上角的提示“soyan 您好”。

（2） 查询测试

以管理员查询书籍为例：



华中科技大学图书馆

Bookname

Q

搜索结果

数据库系统概论（第五版）

删除

作者: 萨师煊 王珊 出版社:高等教育出版社 出版时间:2014

数据库系统概论（第四版）

删除

作者: 王珊 萨师煊 出版社:高等教育出版社 出版时间:2010

图 4.9 查询书籍

(3) 修改测试

以管理员修改书籍为例：

图书信息管理

图书信息

修改信息

作者信息

馆藏信息

书名

数据库系统概论（第五）

出版社

高等教育出版社

出版时间

2014

价格

99.60

提交

图书信息管理

图书信息

修改信息

作者信息

馆藏信息

编号	4
书名	数据库系统概论（第五）
作者	萨师煊 王珊
出版社	高等教育出版社
出版时间	2014
定价	99.60

图 4.10 修改书籍

（4） 新建测试

以管理员新建书籍为例：

新建书籍

书名

数据库系统概论（第三版）

出版社

高等教育出版社

出版时间

2008

价格

19.0

新建

图书信息管理

图书信息

修改信息

作者信息

馆藏信息

编号	10
书名	数据库系统概论（第三版）
作者	
出版社	高等教育出版社
出版时间	2008
定价	19.00

馆藏书籍

没有馆藏

图 4.11 新建书籍

（5） 删除测试

以管理员删除书籍为例：

搜索结果

数据库系统概论（第五）

删除

作者: 萨师煊 王珊 出版社:高等教育出版社 出版时间:2014

数据库系统概论（第四版）

删除

作者: 王珊 萨师煊 出版社:高等教育出版社 出版时间:2010

数据库系统概论（第三版）

删除

作者: 出版社:高等教育出版社 出版时间:2008



图 4.12 删除书籍

(6) 用户测试

a) 借书测试

管理员为用户 soyan 借 14 号书后查看用户待还书籍:

借阅书籍

藏书号

用户名

借阅

待还书籍

书名	藏书号	还书日期	剩余天数	续借
数据结构 (C语言版)	14	2019年7月14日	30	续借
编译原理 (第3版)	6	2019年7月12日	28	无法续借

图 4.13 借阅书籍

b) 续借测试

点击续借按钮，查看借阅记录：

借阅记录			
书名	藏书号	时间	操作
数据结构（C语言版）	14	2019年6月14日 20:16	续借
数据结构（C语言版）	14	2019年6月14日 20:14	借书

图 4.14 续借书籍

c) 还书测试

在管理员处归还刚才所借书籍，并勾选书籍损坏

查看读者借阅记录发现刚才的还书操作；查看罚单发现新生成的因为书籍损坏的罚单

归还书籍			
藏书号			
<input type="text" value="14"/>			
<input checked="" type="checkbox"/> 书籍损坏			
<input type="button" value="还书"/>			

借阅记录			
书名	藏书号	时间	操作
数据结构（C语言版）	14	2019年6月14日 20:22	还书
数据结构（C语言版）	14	2019年6月14日 20:16	续借
数据结构（C语言版）	14	2019年6月14日 20:14	借书

罚单			
描述	金额	时间	状态
藏书号:14 书名:数据结构（C语言版） 损坏	35.00	2019年6月14日 20:22	待处理

图 4.15 归还书籍

d) 限制罚单未缴纳读者借书测试

尝试继续借书，被提示先处理罚单。

借阅书籍

藏书号

13

用户名

soyan

借阅

127.0.0.1:8000 显示

请先处理罚单：藏书号:14 书名:数据结构（C语言版） 损坏

确定

图 4.16 借阅书籍

e) 缴纳罚单测试

点击处理，显示已完成

罚单			
描述	金额	时间	状态
藏书号:14 书名:数据结构（C语言版） 损坏	35.00	2019年6月14日 20:22	已完成

图 4.17 处理罚单

f) 再次借书测试

尝试借 13 号书，成功：

127.0.0.1:8000 显示

借阅成功

确定

图 4.18 再次借书

g) 查看个人信息测试

点击欢迎下拉栏中的个人信息，显示如下：

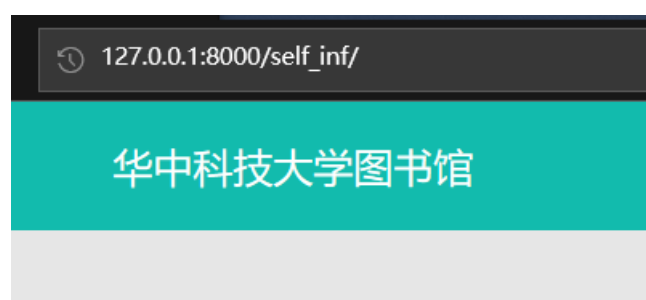
个人信息	
用户名	soyan
姓	沙
名	岩
邮箱	soyan1999@foxmail.com

图 4.19 个人信息

(7) 未登录限制与权限控制测试

a) 未登录限制测试:

在未登录状态下，访问/self_inf/，被跳转到登录界面



Log in

Please login to see this page.

Submit

图 4.20 未登录限制测试

b) 权限限制测试:

在用户登录状态下，访问/administrator/books_manage/被跳转到登录界面

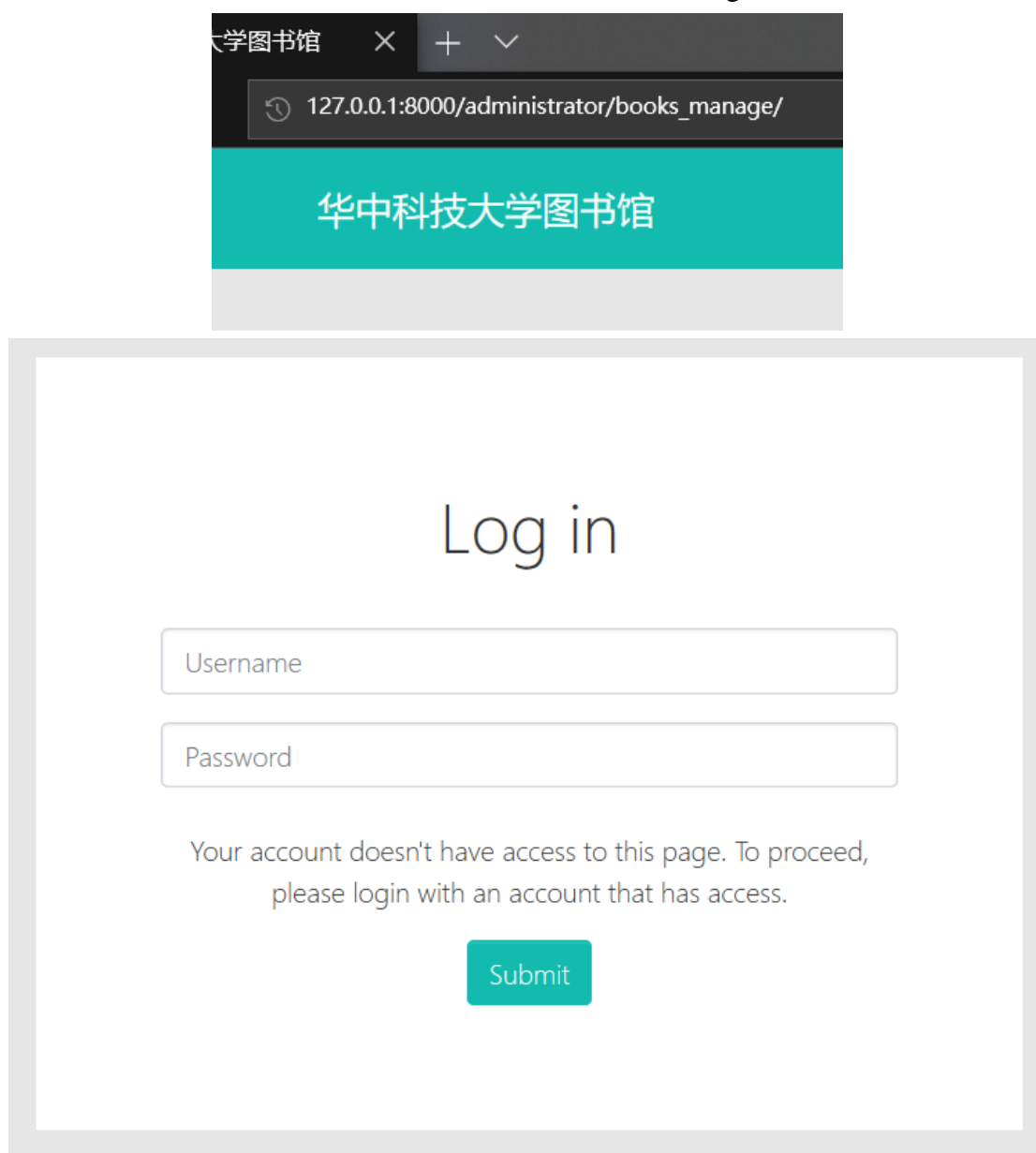


图 4.21 权限限制测试

4.7 系统设计与实现总结

- (1) 根据任务书，结合学校图书馆的实际情况，进行了需求分析。
- (2) 对于任务书中的各项功能，可以合并的进行了合并，方便用户使用系统。
- (3) 根据各实体间的关系，绘制了 ER 图，由此设计出了各个数据库表。
- (4) 根据需求分析的结果，对系统进行了总体设计，确定系统的大致功能。

-
- (5) 将总体设计细化，划分出子模块，用 Python 语言配合 Django，逐个具体实现了功能模块。
 - (6) 向数据库中录入一定数据，进行功能测试，适当修改代码，逐步完善系统。

5 课程总结

- (1) 主要熟悉了 SQL server 的使用，学会了新增用户并设置用户权限；掌握了 sql 查询语句的多种用法以及较为复杂的查询逻辑的实现；掌握了触发器的简单使用。
- (2) 熟悉了 Python 的 Django 工具使用，学会使用 Django 制作较为完备的管理系统；同时熟悉了使用 Django 模型函数操作数据库；熟悉了使用 Django 的 html 模板语言。
- (3) 熟悉了前端 html 与 css，学会使用 Bootstrap 库制作好看的前端界面，对 js 也略有熟悉。
- (4) 总体来说这次试验我学到了很多东西，对我来说也是个很大的挑战，我对于自己制作的成果也比较满意。

参考文献

- [1] 萨师煊, and 王珊. 数据库系统概论.Vol.2.No.000.高等教育出版社, 2000.