

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 综合设计题目： |  |
| 姓 名： | 潘翔 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 物联网工程 |
| 班 级： | IOT1601 |
| 学 号： | U20161489 |
| 指导教师： |  |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2018 年 5月 日

**目 录**

[1 课程任务概述 4](#_Toc1857399291)

[2 软件功能学习部分 7](#_Toc1128720792)

[2.1 任务要求 7](#_Toc1309366992)

[2.2 完成过程 7](#_Toc211890703)

[2.2.1 环境配置 7](#_Toc2079242749)

[2.2.2 添加用户并授权 7](#_Toc653962238)

[2.2.3数据备份与恢复 8](#_Toc1679346388)

[2.3任务总结 8](#_Toc28981103)

[3 Sql练习部分 9](#_Toc1740321492)

[3.1 任务要求 9](#_Toc375124252)

[3.2 完成过程 10](#_Toc1775841113)

[3.2.1 创建表格 10](#_Toc1336679645)

[3.2.2 插入数据 12](#_Toc477836224)

[3.2.3 数据更新 13](#_Toc871285798)

[3.2.4 数据查询 17](#_Toc1861689275)

[3.2.5了解系统的查询性能分析功能（选做） 26](#_Toc1150147642)

[3.2.6 DBMS函数及存储过程和事务（选做） 26](#_Toc1235081392)

[3.3任务总结 26](#_Toc1086795171)

[4 数据库应用系统设计 27](#_Toc1150481755)

[4.1 系统设计目标 27](#_Toc1259629810)

[4.2 需求分析 27](#_Toc1896048099)

[4.2.1 环境配置 27](#_Toc920277642)

[4.3 总体设计 27](#_Toc86983937)

[4.4 数据库设计 27](#_Toc1104325171)

[4.5 详细设计与实现 27](#_Toc346146768)

[4.6 系统测试 27](#_Toc788028454)

[4.7 系统设计与实现总结 27](#_Toc1513630597)

[5 课程总结 29](#_Toc1941770601)

[4.1 任务要求 29](#_Toc360548541)

[4.2 完成过程 29](#_Toc928864687)

[4.2.1 环境配置 29](#_Toc1466268193)

# 1 课程任务概述

**一级标题，单独起页（下同）**

居中 小三号 宋体加粗

简要陈述介绍本实践课程的各项任务要求。

**关于正文排版要求：**

正文统一采用小四号宋体/Times New Roman和1.25倍行距，段前、段后均0磅间隔。

正文应论述清楚，文字简练通顺，插图简明，书写整洁。文中图、表按制图要求绘制。

**关于正文章节标题的排版要求：**

报告章标题称为一级标题，章内小节标题依次分为二级标题、三级标题等。一级标题的编号用数字1，2，…编制；二级标题的编号用1.1，1.2，…编制；三级标题的编号用1.1.1，1.2.1，… 编制；四级及以后各级标题可依此类推。建议标题不超过3级（如1.1.1），超出部分可根据需要使用(1)，①，A，a)，…等形式描述。

标题编号与标题文字之间均用空格隔开，如：“1 引言”、“2.1 需求分析”。报告正文的一级标题（章）须另起一页居中排版。

1级标题宋体小三号字加黑，单倍行距，段前、段后均12磅间隔。

2级标题宋体四号字加黑，1.25倍行距，段前6磅，段后0磅间隔。

3级标题宋体小四号字加黑，单倍行距，段前3磅、段后8磅间隔。

**关于正文中的图的排版要求：**

正文中所有插图要求图面整洁，布局合理，线条粗细均匀，圆弧连接光滑，尺寸标注规范。所有曲线、图表、线路图、流程图、程序框图、示意图等不准徒手画，必须按国家规定标准或工程要求采用计算机或手工绘制。

所有插图均应有图号和图名。图号按章编，如第2章的图为图2.1、图2.2、…，第3章的图为图3.1、图3.2、…等。图名是插图的名称，扼要概括图的内容，字数不宜太多。图号和图名写在图下方，并相对于图居中排版。少数图有图注，图注写在图下面且字号应比图号、图名的字小一号，图名和图注后面均不加标点符号。

所有插图均应在正文中予以引用。引用某插图时，一般写为“…见图x.y”或“图x.y是…”。正文中的插图均须安排在文中第一次引用到该图的正文下面，要求先见文，后见插图，且图一般不跨页绘制。

图中文字、图号和图名，统一采用小五号宋体。

**关于正文中的表的排版要求：**

表格由表号、表名、表头、表身等组成。表号按章编，如第2章的表为表2.1、表2.2、…，第3章的表为表3.1、表3.2、…等。表名是表格的名称，扼要概括表的内容，字数不宜太多。表号、表名放在表的正上方，相对于表体居中排版。表号及表名后不加标点。表头包括栏头、行头，与表身一起构成表格的主体。表中的竖称为栏，横格称为行。表身的内容，一般包括：数据、文字、公式和表图等。表内的数据对应位要对齐。少数表有表注，表注写在表下面且字号应比表号、表名的字小一号。

所有表格均应在正文中予以引用。引用某表格时，一般写为“…见表x.y”或“表x.y是…”。表格应尽量靠近正文的叙述，一般应先见文，后见表，表不跨节。表格允许转页。表格转页部分可以不写表号和表名，但要重复书写表头，并在表头右上角写“（续）”字标注。

表中文字、图号和图名，统一采用小五号宋体。

**关于正文中公式的排版要求：**

公式一般另行居中写，公式末不加标点。若公式前有文字，如例、解、证、假定等，文字顶格写，公式仍居中写。一行写不下时，公式允许转行。公式转行需处理得当，做到既意义正确，又使版面美观匀称。

公式要有编号，公式编号按章编，如第2章的公式为(2.1)、(2.2)、…，第3章的公式为(3.1)、(3.2)、…等。公式编号写在公式右侧行末顶边线，并加圆括号。

公式一般应在正文中予以引用，引用时以公式编号指示公式。正文中常有公式中表示量的符号说明，采用“式中”二字作为标志。一般可写成接排形式，如“式中，A指……；B指……”。

**关于实践报告的目录格式要求：**

目录是课程实验报告的纲要。正文的各级标题（一般最多取三级）、附录应编入目录，但目录本身不出现在其中。

目录要求层次清晰，含标题及对应的起始页号。目录的最后 “附录”无章节号。

课程实验报告正文、参考文献和附录页面，使用“1，2，3，… ”编连续页码。页码应标在页面的右下角。

目录中正文的各级标题名称、参考文献和附录及其对应的起始页号，务必与报告中正文的各级标题名称、附录及其对应的起始页号保持一致。

正文（下同）

小四号 宋体/Times New Roman 1.25倍行距

# 2 软件功能学习部分

## 2.1 任务要求

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。

## 2.2 完成过程

### 2.2.1 环境配置

1. 操作系统：Arch Linux x64
2. 数据库：MariaDB
3. 可视化工具：MySQL-workbrench
4. 安装过程：

yaourt -S mariadb

systemctl start mariadb.service

### 2.2.2 添加用户并授权

1. 创建用户

CREATE USER `hover`@`localhost` IDENTIFIED BY '123456';

1. 创建数据库

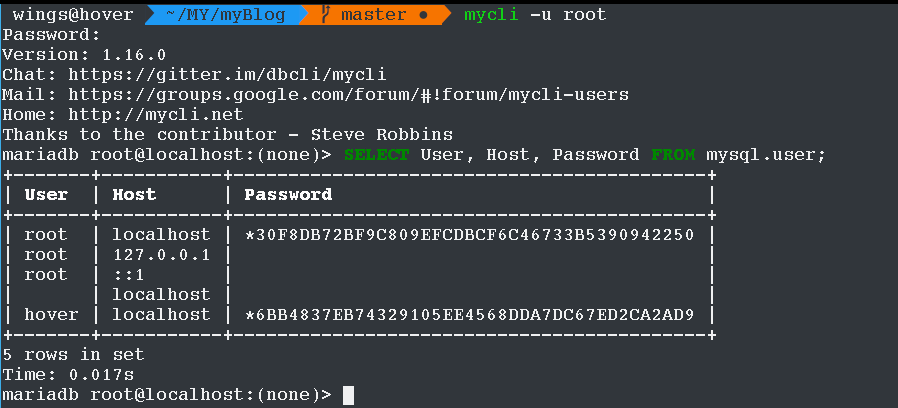
CREATE DATABASE DBlabs;

1. 用户授权

GRANT ALL ON DBlabs.\* TO `hover`@'locallost';

1. 查看当前所有的用户

SELECT User, Host, Password FROM mysql.user;



**图2-1 添加用户情况**



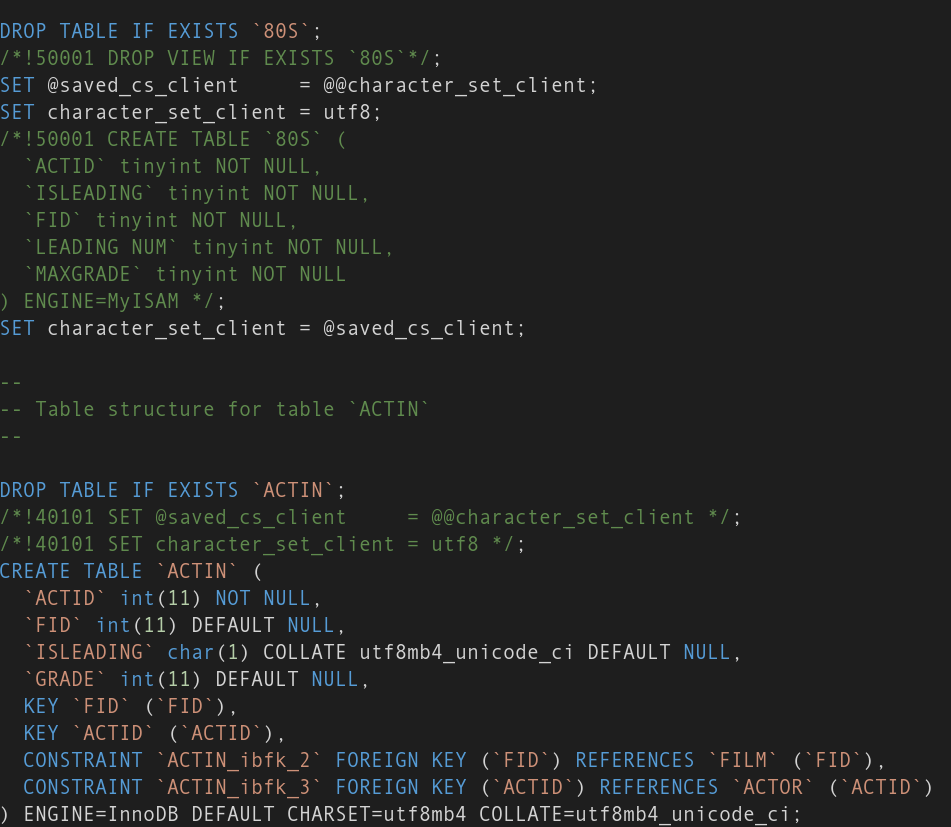
**图2-2 显示表格**

### 2.2.3数据备份与恢复

1. 逻辑备份 mysqldump

使用逻辑备份，备份恢复所需的SQL

mysqldump -u hover -p  --databases DBlabs > DBlabsBack.sql



**图2-3 操作文件结果**

1. 物理备份mysqlhotcopy

yaourt -S perl-dbi

yaourt -S perl-dbd-mysql

## 2.3任务总结

（总结实验中遇到的主要问题及所采用的解决方案）

# 3 Sql练习部分

## 3.1 任务要求

**1）创建下列跟电影相关的关系，包括主码和外码的说明**

**电影表**【电影编号，电影名称，电影类型，导演姓名，电影时长（以分钟计），是否3D，用户评分】

FILM(FID int, FNAME char(30), FTYPE char(10), DNAME char(30), length int, IS3D char(1)，GRADE int)。

主码为电影编号，IS3D取值为’Y’表示是3D电影，’N’表示不是，用户评分规定为0~100分之间或者为空值。

**演员表**【演员编号，演员姓名，性别，出生年份】

ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int)

主码为演员编号

**参演表**【演员编号，电影编号，是否主角，用户对该演员在该电影中的评分】

ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int)

主码、外码请依据应用背景合理定义。ISLEADING取值为’Y’表示是，’N’表示不是主角，也可能取空值，表示不太确定该演员在该电影中是否主角。GRADE规定为0~100分之间或者为空值。

**电影院表**【电影院编号，电影院名字，影院所在行政区，影院地址】

[THEATER](javascript:;) (TID int, TNAME char(20), TAREA char(20), ADDRESS char(30))

主码为电影院编号，影院所在行政区取值如“洪山区”、“武昌区”等等。

**上映表**【电影编号，影院编号，上映年份，上映月份】

SHOW(FID int, TID int , PRICE int, YEAR int , MONTH int)

假定一部电影在一家影院只上映一次，主码、外码请依据应用背景合理定义。

**2）观察性实验**

验证在建立外码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。

## 3.2 完成过程

### 3.2.1 创建表格

1. 创建表格

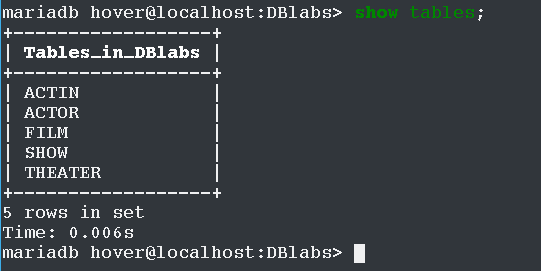
CREATE TABLE FILM(FID int, FNAME char(30), FTYPE char(10), DNAME char(30), length int, IS3D char(1)，GRADE int);

CREATE TABLE ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int);

CREATE TABLE ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int);

CREATE TABLE THEATER(TID int, TNAME char(20), TAREA char(20), ADDRESS char(30));

CREATE TABLE `SHOW`(FID int, TID int , PRICE int, YEAR int , MONTH int);



**图3.1 显示表格创建结果**

1. 定义主键

ALTER TABLE FILM ADD PRIMARY KEY (FID);

ALTER TABLE ACTOR ADD PRIMARY KEY (ACTID);

ALTER TABLE ACTIN ADD PRIMARY KEY (ACTID);

ALTER TABLE `[THEATER](javascript:;)` ADD PRIMARY KEY (TID);

ALTER TABLE `SHOW`ADD PRIMARY KEY (FID);

使用

SELECT

t.TABLE\_NAME,

c.COLUMN\_NAME

FROM

INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS ASt,

information\_schema.TABLES AS ts,

information\_schema.KEY\_COLUMN\_USAGE As c

WHERE

t.TABLE\_NAME = ts.TABLE\_NAME

AND ts.TABLE\_NAME  = c.TABLE\_NAME

AND t.CONSTRAINT\_TYPE = 'PRIMARY KEY'

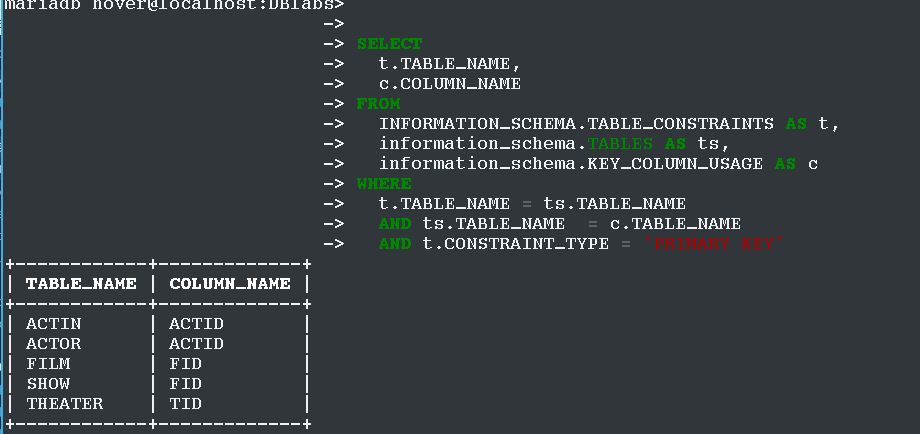


图3.2 显示主键结果

1. 定义外键

ACTIN外键:

ALTER TABLE ACTIN ADD FOREIGN KEY (ACTID) REFERENCES ACTOR (ACTID);

ALTER TABLE ACTIN ADD FOREIGN KEY (FID) REFERENCES FILM(FID);

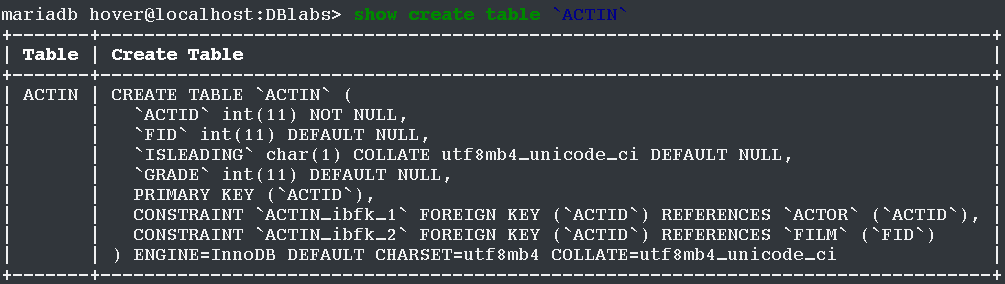
SHOW外键:

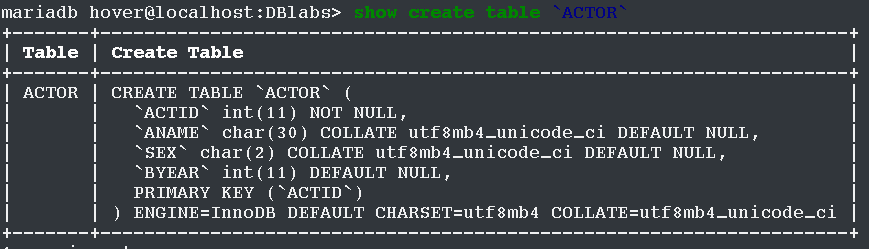
ALTER TABLE `SHOW` ADD FOREIGN KEY (FID) REFERENCES FILM(FID);

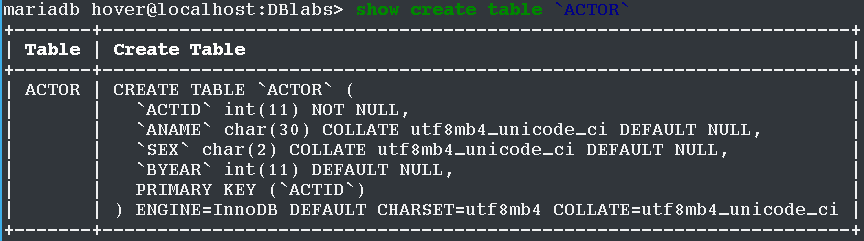
ALTER TABLE `SHOW` ADD FOREIGN KEY (TID) REFERENCES THEATER (TID);

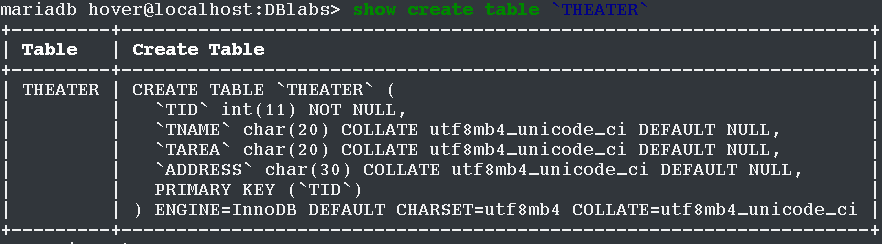
使用

show create table (表名)查看外键









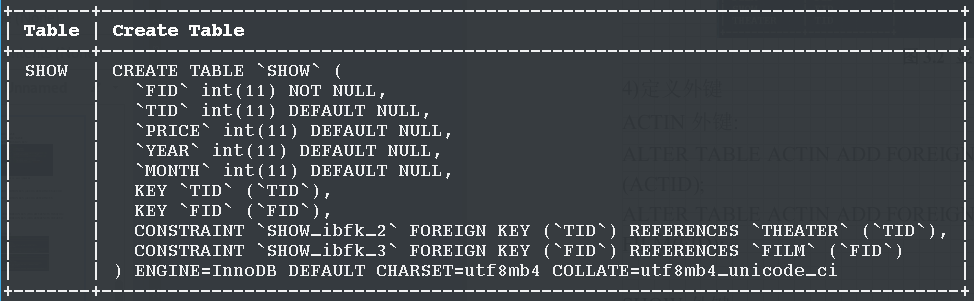


图3.2 定义外键结果

### 3.2.2 插入数据

INSERT INTO FILM(FID , FNAME , FTYPE , DNAME , length , IS3D , GRADE ) VALUES(4, 'film4', 'Adventure' , 'NH3' ,122, 'Y', 93)



图3.3 FILM插入结果

INSERT INTO ACTOR(ACTID, ANAME, SEX, BYEAR)    
                         VALUES(4, 'Perch', 'M', 2005 )

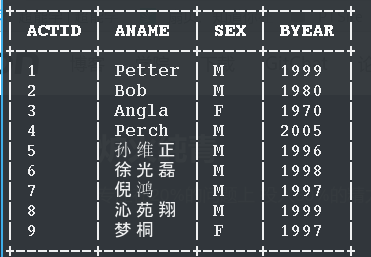


图3.4 Actor插入结果



图3.5 Theater插入结果



图3.6 Show插入结果

### 3.2.3 数据更新

1. 分别用一条sql语句完成对电影表基本的增、删、改的操作；

INSERT INTO FILM(FID , FNAME , FTYPE , DNAME , length , IS3D , GRADE ) VALUES(4, 'film4', 'Adventure' , 'NH3' ,122, 'Y', 93)

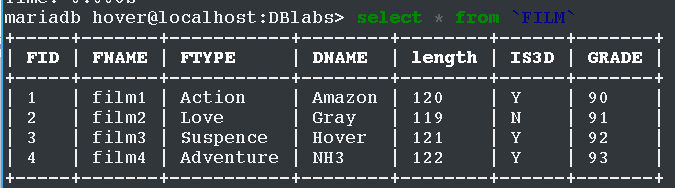


图3.4 sql插入

update `FILM`  set `IS3D`='N' where `FID` = 4;

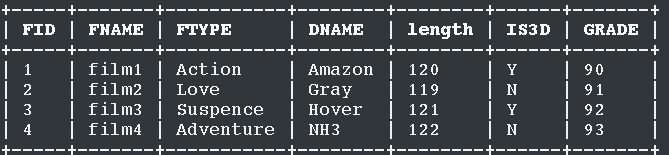


图3.6 sql修改

DELETE FROM `FILM` WHERE FID = 4;



图3.5 sql删除

1. 批处理操作:将演员表中的90后演员记录插入到一个新表YOUNG\_ACTOR中)
2. 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。

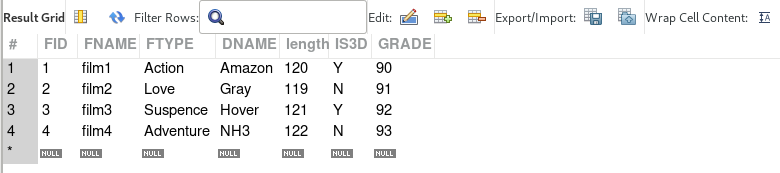


图3.7 操作界面import/export

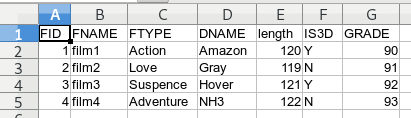


图3.8 导出csv文件

create table import\_table like `FILM`

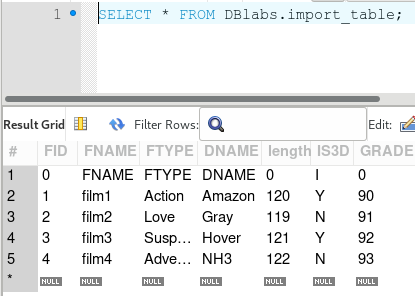


图3.9 导入至新表

1. 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。



图3.10 重复元素插入结果

**update** `FORTEST` set id=3 **where** id=1;

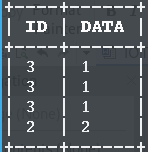


图3.10 重复元组修改

**delete** **from** `FORTEST` **where** id=3;

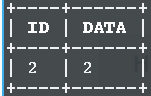


图3.11 重复元组删除

1. 创建视图：创建一个有80后演员作主角的参演记录视图，其中的属性包括：演员编号、演员姓名、出生年份、作为主角参演的电影数量、这些电影的用户评分的最高分。

**create** view `80S` **as**   
**select** `ACTID`,`ISLEADING`,`FID`,count(**case** `ISLEADING` **when** 'Y' **then** 1 end) **as** 'LEADING NUM' ,max(`GRADE`) **as** 'MAXGRADE'   
**from** `ACTIN`    
**WHERE** `FID` **in**   
(**select** `FID` **from** `ACTOR`   
**join** `ACTIN`   
**on** `ACTIN`.`ACTID`=`ACTOR`.`ACTID`   
**where** 1980<=`BYEAR` **and** `BYEAR`<1989 **and** `ISLEADING`='Y'   
**group** **by** `FID`)   
**group** **by** `ACTID`;

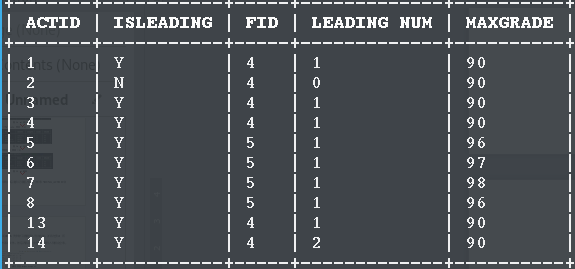


图3.11 视图创建结果

1. 触发器实验：编写一个触发器，用于实现对电影表的完整性控制规则：当增加一部电影时，若导演的姓名为周星驰，则电影类型自动设置为“喜剧”。

**create** **trigger** StephenChowComedy   
after **insert** **on** `FILM`    
**for** **each** row   
begin   
**update** `FILM` SET `FTYPE`='Comdedy' **where** new.`DNAME`='周星驰';   
end;

**insert** **into** `FILM`(`FID`,`FNAME`,`DNAME`) **VALUES**(13,'喜剧','周星驰');

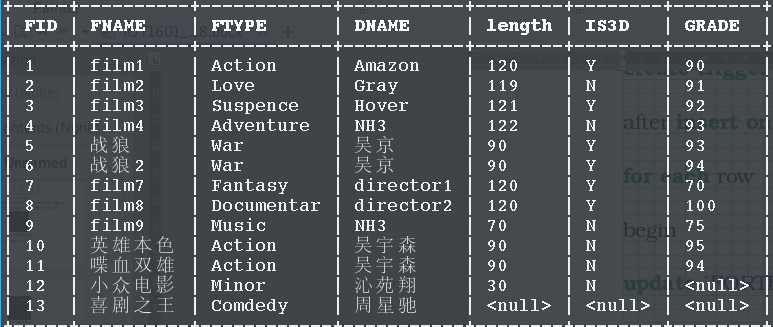


图3.12 触发器结果

### 3.2.4 数据查询

1. 查询“战狼”这部电影在洪山区各家影院的2017年的上映情况，并按照上映的月份的降序排列；

select TID, PRICE, MONTH from `SHOW`where `YEAR` =2017 and `TID` in (select `TID` from `THEATER` where `TAREA` = '洪山区')   
order by `MONTH` desc;

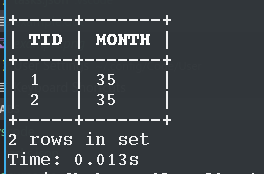


图3.1 查询1结果

1. 查询所有无参演演员信息的电影的基本信息，并且将结果按照电影类型的升序排列，相同类型的电影则按照用户评分的降序排列；

select \* from `FILM` where `FID` not in (select `FID` from `ACTIN`) order by `FTYPE` ASC ,`GRADE` DESC

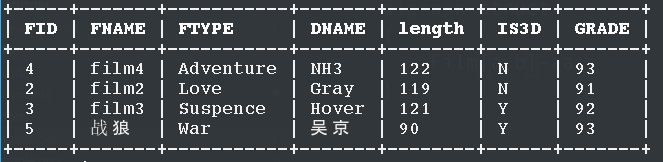


图3.1 查询2结果

1. 查询所有直到2017年仍未上映的电影编号、电影名称、导演姓名；

select  \* from `SHOW`  group by `FID` having `YEAR`<2018;



图3.1 查询3结果

1. 查询在各家电影院均上映过的电影编号；

select `FID` FROM `SHOW` group by `FID` having COUNT(\*)=(SELECT COUNT(\*) FROM `THEATER` );



图3.1 查询4结果

1. 查询所有用户评分低于80分或者高于89分的电影编号、电影名称、导演姓名及其用户评分，要求where子句中只能有一个条件表达式；

**select** `FID`,`FNAME`,`DNAME`,`GRADE` **from** `FILM`   
**where** `GRADE` **not** **between** 80 **and** 89;

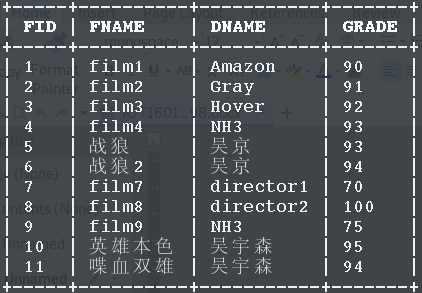


图3.1 查询5结果

1. 查询每个导演所执导的全部影片的最低和最高用户评分；

select `DNAME`,MAX(`GRADE`) AS MAXGRADE,MIN(`GRADE`) AS MINGRADE  FROM `FILM` GROUP BY `DNAME`;

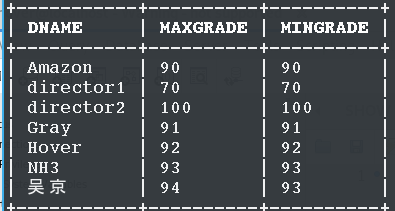


图3.1 查询6结果

1. 查询至少执导过2部电影的导演姓名、执导电影数量；

select `DNAME`,count(\*) as movieNum FROM `FILM` GROUP BY `DNAME` HAVING count(\*)>=2;

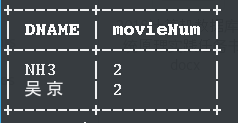


图3.1 查询7结果

1. 查询至少2部电影的用户评分超过80分的导演及其执导过的影片数量、平均用户评分；

select `DNAME`,count(\*) as greatMovieNum FROM `FILM` where `GRADE`>80 GROUP BY `DNAME` having count(\*)>=2;

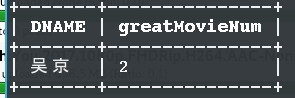


图3.1 查询8结果

1. 查询至少执导过2部电影的导演姓名以及跟这些导演合作过的演员编号、姓名；

select ACTIN.ACTID,ANAME   
from   
(select `DNAME`,`FID`  from `FILM` where `GRADE`>80 group by `DNAME` having count(\*)>=2) as a,   
ACTIN,   
ACTOR   
WHERE a.`FID`=ACTIN.`FID`    
AND ACTIN.`ACTID`=ACTOR.`ACTID`;

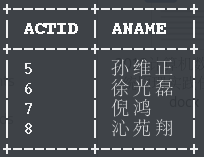


图3.1 查询9结果

1. 查询每个演员担任主角的电影中的平均用户评分；

用户评分：用户对于演员的评分

(select a.`ACTID`, AVG(`GRADE`) from `ACTIN` as a,`ACTOR` as b where a.`ACTID` = b.`ACTID` and a.`ISlEADING`='Y'  group by  `ACTID`);

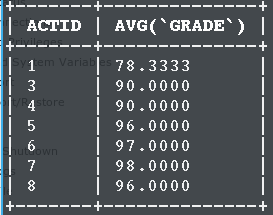


图3.1 查询10结果

1. 查询用户评分超过90分的电影的最早上映年月；

**select** `SHOW`.`FID` ,min(str\_to\_date(concat(CAST(`YEAR` **AS** CHAR ),CAST(`MONTH` **AS** CHAR )) ,'%Y%m')) **as** 'earliestTIme' **from**    
`FILM`    
**join**    
`SHOW`    
**on** `FILM`.`FID`=`SHOW`.`FID`    
**where** `GRADE`>90    
**group** **by** `SHOW`.`FID`;

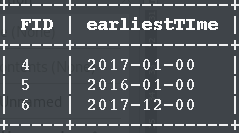


图3.1 查询11结果

1. 查询用户评分超过90分的电影的最早上映年月及其相应的上映影院编号；

**select** `SHOW`.`FID`,`SHOW`.`TID` ,min(str\_to\_date(concat(CAST(`YEAR` **AS** CHAR ),CAST(`MONTH` **AS** CHAR )) ,'%Y%m')) **as** 'earliestTIme' **from** `FILM`   
**join**   
`SHOW`   
**on** `FILM`.`FID`=`SHOW`.`FID`   
**where** `GRADE`>90   
**group** **by** `SHOW`.`FID`;



图3.1 查询10结果

1. 查询每个电影的上映总次数；

**select** `FID`,COUNT(`FID`) **AS** 'totalShowTimes' **from** `SHOW` **group** **by** `FID`

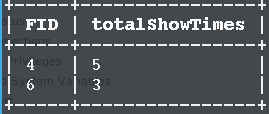


图3.1 查询10结果

1. 查询执导过动作片，或者警匪片，或者枪战片的导演的姓名，要求where子句中只能有一个条件表达式；

**select** `DNAME` **from** `FILM` **where** `FTYPE` **in** ('Action','Shooting','Police' ) **group** **by** `DNAME`;



图3.1 查询14结果

1. 查询所有“战狼”系列的电影的编号、电影名称、上映电影院名称及其上映年月，结果按照电影名称的升序排列；

**select** a.`FID`,a.`FNAME`,b.`TID`,b.`YEAR`,b.`MONTH`,c.`TNAME` **from** `FILM` **as** a ,`SHOW` **as** b, `THEATER` **as** c   
**where** a.`FNAME` **LIKE** '%战狼%'  
**and** a.`FID`=b.`FID`   
**and** b.`TID`=c.`TID`   
**order** **by** `FNAME` **asc**;

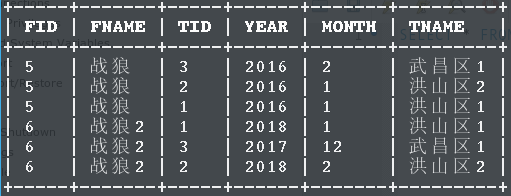


图3.1 查询10结果

1. 查询在同一个年月上映1号和2号电影的影院编号；

**select** a.`FID`,b.`FID`,a.`YEAR`,a.`MONTH` **from** `SHOW` **as** a ,`SHOW` **as** b **where** a.`MONTH`=b.`MONTH` **and** a.`YEAR`=b.`YEAR` **and**   
a.`FID`<b.`FID` **and** a.`TID`=b.`TID` **group** **by** a.`TID`;



图3.1 查询10结果

1. 查询所有没参演过用户评分85分以下电影的演员的编号、姓名；

**select** `ACTID`,`ANAME` **from** `ACTOR` **where** `ACTID` **not** **in** (**select** b.`ACTID` **from** `FILM` **as** a, `ACTIN` **as** b **where** a.`FID`=b.`   
FID` **and** a.`GRADE`<85);

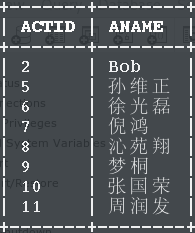


图3.1 查询10结果

1. 查询参演过“吴宇森”执导过的所有电影的演员姓名；

**select** `ANAME` **from** `ACTOR`    
**where** `ACTID` **in**    
(**select** `ACTID` **from** `ACTIN`    
**where**`FID` **in** (**select** `FID` **from** `FILM` **where** `DNAME`='吴宇森')   
**group** **by** `ACTID`   
**having** count(\*)=(**select** count(\*) **from** `FILM` **where** `DNAME`='吴宇森'));



图3.1 查询10结果

1. 查询所有的演员的编号、姓名及其参演过的电影名称，要求即使该演员未参演过任何电影也要能够输出其编号、姓名；

**select** `ACTID`,`ANAME`,`FNAME` **from**   
(**select** a.`ACTID`,`ANAME`,b.`FID` **from** `ACTOR` **as** a   
**left** **join**   
`ACTIN` **as** b   
**on** a.`ACTID`=b.`ACTID`) **as** c   
**left** **join**   
`FILM` **as** d   
**on** c.`FID`=d.`FID`;

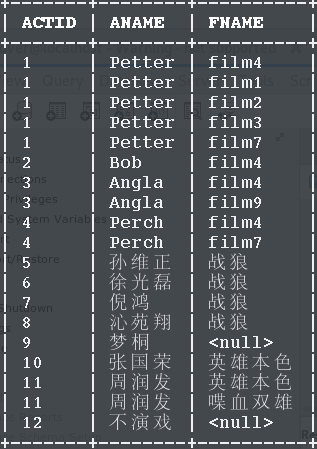


图3.1 查询19结果

1. 查询所有上映超过3次但没有用户评分的电影编号、名称。

**select** a.`FID`,b.`FNAME` **from** `SHOW`  **as** a  
**left** **join** `FILM` **as** b     
**on** a.`FID`=b.`FID`   
**where** b.`GRADE` **is** NULL   
**group** **by** a.`FID`   
**having** count(\*)>=3;



图3.1 查询20结果

### 3.2.5了解系统的查询性能分析功能（选做）

目的: 选择上述3.2.4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。

使用explain查询3.2.4(20)SQL语句执行计划:

**explain** **select** a.`FID`, b.`FNAME` **from** `SHOW` **as** a    
**left** **join** `FILM` **as** b   
**on** a.`FID`=b.`FID`    
**where** b.`GRADE` **is** NULL    
**group** **by** a.`FID`    
**having** count(\*)>=3;



图3.1 查询20结果

其中:

table:显示关于哪张表

type:连接使用的类型 此处为 index连接(group操作) 和 等值连接(left join操作)

possible\_keys :显示可能应用在这张表中的索引

key :实际使用的索引。如果为NULL，则没有使用索引

key\_len :使用的索引的长度

ref :显示索引的哪一列被使用了,如果可能,为常数

rows :MYSQL认为必须检查的用来返回请求数据的行数

Extra :关于MYSQL如何解析查询的额外信息

### 3.2.6 DBMS函数及存储过程和事务（选做）

1. 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数，为电影表增加首映时间属性，然后查询下个月首映的电影信息。

**alter** **table** `FILM` **add** premiere date;

**update** `FILM` set `premiere`=date\_format("2018-1-1","%y-%m-%d") **where** `FID`=1;

**update** `FILM` set `premiere`=date\_format("2018-6-1","%y-%m-%d") **where** `FID`=6;

**select** \* **from** `FILM` **where** month(`premiere`)-month(now())=1;



图3.1 下月首映电影

1. 编写一个依据演员编号计算在其指定年份参演的电影数量的自定义的函数，并利用其查询2017年至少参演过5部电影的演员编号。

自定义函数:

**create** function countActorFilmNumByYear(actid int,yearnum int)   
returns int   
begin   
**return** (**select** count(\*) **from** `ACTOR`   
**join** `ACTIN`    
**on** `ACTOR`.`ACTID`=`ACTIN`.`ACTID`    
**join** `FILM`   
**on** `ACTIN`.`FID`=`FILM`.`FID` **and** year(`premiere`)=yearnum   
**where** `ACTOR`.`ACTID`=actid);   
end;



图3.1 自定义函数测试

**select** `ACTID`,countActorFilmNumByYear(`ACTID`,2017) **from** `ACTOR` **where** countActorFilmNumByYear(`ACTID`,2017)>=5;

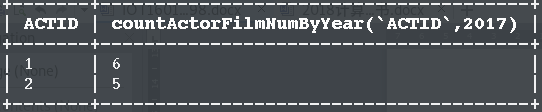


图3.1 2017年上映电影中出演5场以上演员及电影数

1. 尝试编写DBMS的存储过程，建立每家影院的上映电影总数的统计表，并通过存储过程更新该表。

**create** **or** **replace** **procedure** count\_filmNum(stime date)   
begin   
**create** **or** **replace** **table** `countFilm` **select** `TID`,count(\*) **as** 'filmCount',`YEAR`,`MONTH` **from** `SHOW`    
**where** str\_to\_date(concat(CAST(`YEAR` **AS** CHAR ),CAST(`MONTH`**AS** CHAR )) ,'%Y%m')< stime  **group** **by** `TID`;   
end;

**call** count\_filmNum(now());

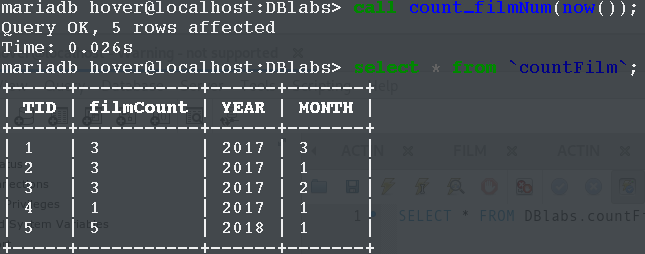


图3.1 截止当前时间之前(now())的各电影院数目统计

4）尝试在DBMS的交互式界面中验证事务机制的执行效果。

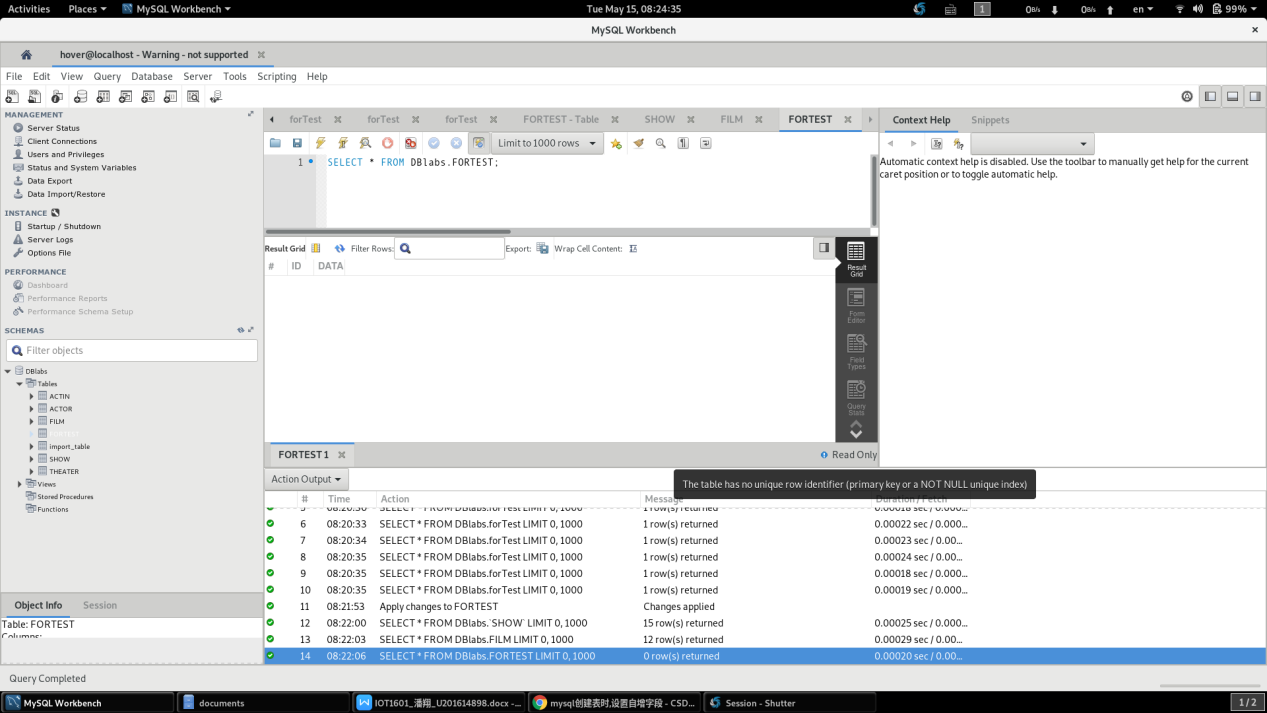
## 3.3任务总结

1. 当存在外键时，删除主键出错：



删除外键，然后删除主键

1. 当无unique和无主键时，无法使用可视化终端插入数值



1. 时间函数的使用
2. 当列重复如何压缩
3. 创建触发器时需要root权限
4. 自定义函数权限

This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled



图3.1 错误函数

set global log\_bin\_trust\_function\_creators=TRUE;

# 4 数据库应用系统设计

## 4.1 系统设计目标

自行选择所擅长的DBMS软件以及数据库应用系统（客户端程序或者网站）的程序开发工具，参考后面的题目例子，拟定一个自己感兴趣的数据库应用系统题目，完成该小型数据库应用系统的设计与实现工作。主要内容包括：需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

选择题目：

## 4.2 需求分析

### 4.2.1 环境配置

## 4.3 总体设计

## 4.4 数据库设计

## 4.5 详细设计与实现

## 4.6 系统测试

## 4.7 系统设计与实现总结

# 5 课程总结

## 4.1 任务要求

自行选择所擅长的DBMS软件以及数据库应用系统（客户端程序或者网站）的程序开发工具，参考后面的题目例子，拟定一个自己感兴趣的数据库应用系统题目，完成该小型数据库应用系统的设计与实现工作。主要内容包括：需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

选择题目：

## 4.2 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。）

### 4.2.1 环境配置