

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： |  |
| 学 号： |  |
| 姓 名： |  |
| 指导教师： |  |

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

2022年 7月 2日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

|  |  |
| --- | --- |
| 总分 |  |

目 录

[1 课程任务概述 1](#_Toc107768930)

[2 任务实施过程与分析 2](#_Toc107768931)

[2.1 数据库、表、完整性约束的定义 2](#_Toc107768932)

[2.2 表结构与完整性约束的修改（ALTER） 3](#_Toc107768933)

[2.3 数据查询（Select） 4](#_Toc107768934)

[2.4 数据的插入、修改与删除（Insert, Update, Delete） 8](#_Toc107768935)

[2.5 视图 9](#_Toc107768936)

[2.6 存储过程与事务 10](#_Toc107768937)

[2.7 触发器 10](#_Toc107768938)

[2.8 用户自定义函数 11](#_Toc107768939)

[2.9 安全性控制 11](#_Toc107768940)

[2.10 并发控制与事务的隔离级别 13](#_Toc107768941)

[2.11 备份+日志：介质故障与数据库恢复 14](#_Toc107768942)

[2.12 数据库设计与实现 14](#_Toc107768943)

[2.13 数据库应用开发(JAVA篇) 15](#_Toc107768944)

[3 课程总结 17](#_Toc107768945)

# 课程任务概述

“数据库系统原理实践”是配合“数据库系统原理”课程独立开设的实践课，注重理论与实践相结合。本课程以MySQL为例，系统性地设计了一系列的实训任务，内容涉及以下几个部分：

1. 数据库、表、索引、视图、约束、存储过程、函数、触发器、游标等数据对象的管理与编程；
2. 数据查询，数据插入、删除与修改等数据处理相关任务；
3. 数据库的安全性控制，完整性控制，恢复机制，并发控制机制等系统内核的实验；
4. 数据库的设计与实现；
5. 数据库应用系统的开发(JAVA 篇)。

# 任务实施过程与分析

## 数据库、表、完整性约束的定义

本小节内容为熟悉创建数据库、表的创建和表的完整性约束的定义。

### 创建数据库

较为简单，一条create语句即可完成。

create database beijing2022;

### 创建表及表的主码约束

create table t\_tmp(

id int primary key,

name varchar(32),

deptId int,

salary float

);

### 创建外码约束

create table staff(

staffNo int primary key, staffName varchar(32),

gender char(1), dob date, salary numeric(8,2),

deptNo int, constraint FK\_staff\_deptNo foreign key(deptNo) references dept(deptNo)

);

### CHECK约束

create table products(

pid char(10) primary key,

name varchar(32),

brand char(10) constraint CK\_products\_brand CHECK(brand in ('A','B')),

price int constraint CK\_products\_price CHECK(price >0));

### DEFAULT约束

create table hr(

id char(10) primary key,

name varchar(32) not null,

mz char(16) default '汉族');

### UNIQUE约束

create table s(

sno char(10) primary key,

name varchar(32) not null,

ID char(18) unique);

## 表结构与完整性约束的修改（ALTER）

这部分的主要内容为使用alter语句对表的结构及完整性约束的修改。

### 修改表名

alter table your\_table rename as my\_table;

### 添加与删除字段

需使用一条drop语句和一条add语句。

alter table orderDetail drop orderDate;

alter table orderDetail add unitPrice numeric(10,2);

### 修改字段

alter table addressBook modify QQ char(12), rename column weixin to wechat;

### 添加、删除与修改约束

均为add constraint语句的使用。

alter table Staff add constraint primary key(staffNo);

alter table Dept add constraint FK\_Dept\_mgrStaffNo foreign key(mgrStaffNo) references Staff(staffNo);

alter table Staff add constraint FK\_Staff\_dept foreign key(dept) references Dept(deptNo);

alter table Staff add constraint CK\_Staff\_gender check(gender in ('M','F'));

alter table Dept add constraint UN\_Dept\_deptName unique(deptName);

## 数据查询（Select）

本部分的主要内容为使用select语句完成一些表的查询工作。

### 查询客户主要信息

使用最简单的select语句即可完成。

select c\_name,c\_phone,c\_mail

from client

order by c\_id;

### 邮箱为null的用户

使用带where字句的select语句即可实现，使用is null判定邮箱为null

select c\_id,c\_name,c\_id\_card,c\_phone

from client

where c\_mail is null;

### 既买了保险又买了基金的客户

可以使用property表的自连接完成。

select c\_name,c\_mail,c\_phone

from client,property A,property B

where A.pro\_c\_id=B.pro\_c\_id and A.pro\_type=3 and B.pro\_type=2 and A.pro\_c\_id=client.c\_id

group by client.c\_id

order by client.c\_id;

### 办理了储蓄卡的客户信息

使用like进行字符串匹配。

select client.c\_name,client.c\_phone,bank\_card.b\_number

from client,bank\_card

where client.c\_id=bank\_card.b\_c\_id and b\_type like '储蓄卡'

order by client.c\_id;

### 每份金额在30000～50000之间的理财产品

直接查询finance\_product表找到符合条件的产品即可。

select p\_id,p\_amount,p\_year

from finances\_product

where p\_amount>=30000 and p\_amount<=50000

order by p\_amount asc,p\_year desc;

### 商品收益的众数

使用派生表查找到众数的数值，并使用同一个派生表统计众数出现的次数。

select pro\_income,presence

from(

select pro\_income,count(pro\_income) as presence

from property

group by pro\_income

) as A

where presence=(

select max(presence)

from (

select pro\_income,count(pro\_income) as presence

from property

group by pro\_income

) as B

);

### 未购买任何理财产品的武汉居民

用派生表查找到购买过理财产品的居民，那么只需要查找不在该派生表中的武汉居民即可。

select c\_name,c\_phone,c\_mail

from client

where c\_id\_card like '4201%' and c\_id not in(

select pro\_c\_id as c\_id

from property

where pro\_type = 1

)

order by c\_id;

### 持有两张信用卡的客户

用派生表查询持有两张信用卡的客户，在client表中查找对应的用户信息即可。

select c\_name,c\_id\_card,c\_phone

from client

where c\_id in(

select b\_c\_id

from bank\_card

where b\_type like '信用卡'

group by b\_c\_id

having count(\*)>1

);

### 购买了货币型基金的客户信息

在子查询中找到购买货币型基金的客户id，在client表中查找对应的客户信息即可。

select c\_name,c\_phone,c\_mail

from client

where c\_id in (

select pro\_c\_id

from property,fund

where pro\_type=3 and pro\_pif\_id=f\_id and f\_type like '货币型'

)

order by c\_id;

### 投资总收益前三名的客户

使用sum语句计算每名客户可用资产的总收益即可。使用order by字句按降序排列后，使用limit字句限制输出数量为3.

select c\_name,c\_id\_card,sum(pro\_income) as total\_income

from client,property

where c\_id = pro\_c\_id and pro\_status like '可用'

group by c\_id

order by total\_income desc

limit 3;

### 黄性客户持卡数量

为保留无银行卡用户的信息，将client表和bank\_card表进行左连接，随后使用sum语句计数即可。

select c\_id,c\_name,count(b\_number) as number\_of\_cards

from client left join bank\_card on (c\_id=b\_c\_id)

where c\_name like '黄%'

group by c\_id

order by number\_of\_cards desc;

### 客户理财、保险与基金投资总额

将client表与这样一个派生表进行左连接：派生表统计每名用户每种投资类型的金额，为防止合并时去掉相同元组，每种投资的结果用union all进行连接，最后按照用户id进行求和即可。若一名用户没有进行任何类型的投资，那么sum计算结果将为null，可使用IFNULL函数将null值置为0。

此查询语句过长，从略。

### 客户总资产

与2.3.12类似，除了计算客户的各种类型的投资产品总值外，也要计算客户所有投资产品的预计收益、客户所有储蓄卡的余额并减去客户所有信用卡的透支总金额。为使用sum函数进行整体计算，可将信用卡的透支金额视为负值参与运算。

查询语句过长，从略。

### 第N高问题

该任务关卡跳过。

### 基金收益两种方式排名

该任务关卡跳过。

### 持有完全相同基金组合的客户

该任务关卡跳过。

### 购买基金的高峰期

该任务关卡跳过。

### 至少有一张信用卡余额超过5000元的客户信用卡总余额

该任务关卡跳过。

### 以日历表格式显示每日基金购买总金额

该任务关卡跳过。

## 数据的插入、修改与删除（Insert, Update, Delete）

本部分内容为熟悉Insert、Update、Delete语句的使用，对表中数据进行增加、修改或删除。

### 插入多条完整的客户信息

直接使用Insert语句即可实现。代码主体为插入的信息，故省略。

### 插入不完整的客户信息

使用Insert语句，但是在表名后需要指定数据对应的列。

insert into client (c\_id,c\_name,c\_id\_card,c\_phone,c\_password)

values (33,'蔡依婷','350972199204227621','18820762130','MKwEuc1sc6');

### 批量插入数据

使用from语句指定从已知表中读取数据并进行插入。

insert into client

select \*

from new\_client;

### 删除没有银行卡的客户的信息

在子查询中查找有银行卡客户的id，删除id不在该派生表中的用户的信息。

delete from client

where c\_id not in (

select b\_c\_id as c\_id

from bank\_card

);

### 冻结客户资产

在子查询中对应客户的id，在property表中将该客户的所有资产状态设置为冻结即可。

update property

set pro\_status = '冻结'

where pro\_c\_id in (

select c\_id

from client

where c\_phone like '13686431238'

);

### 连接更新

在子查询中按与c\_id对应的顺序建立子查询表，令待插入列等于这个派生表即可。

update property

set pro\_id\_card = (

select c\_id\_card as pro\_id\_card

from client

where c\_id = pro\_c\_id

);

## 视图

本节的主要内容为视图的创建和基于视图的查询

### 创建所有保险资产的详细记录视图

创建视图操作即为select查询操作前增加create view语句，较为简单。

create view v\_insurance\_detail

as

select c\_name,c\_id\_card,i\_name,i\_project,pro\_status,pro\_quantity,i\_amount,i\_year,pro\_income,pro\_purchase\_time

from client,insurance,property

where c\_id=pro\_c\_id and pro\_pif\_id=i\_id and pro\_type=2;

### 基于视图的查询

select语句的from字句采用之前创建的视图表即可。

select c\_name, c\_id\_card, sum(i\_amount\*pro\_quantity) as insurance\_total\_amount, sum(pro\_income) as insurance\_total\_revenue

from v\_insurance\_detail

group by c\_name, c\_id\_card

order by insurance\_total\_amount desc;

## 存储过程与事务

使用流程控制、游标、事务等较高级操作完成任务。

### 使用流程控制语句的存储过程

利用斐波那契数列公式进行循环插入数据。在进入循环前插入数据（0，0），后续循环中插入的n值递增1，fibn值为上轮循环中插入值和上上轮插入的值。

### 使用游标的存储过程

该任务关卡跳过。

### 使用事务的存储过程

该任务关卡跳过。

## 触发器

本小节内容为熟悉触发器的定义和使用。

### 为投资表property实现业务约束规则-根据投资类别分别引用不同表的主码

根据题目要求，设立一个时机为向property表插入前的触发器。当出现插入动作时，检查pro\_type的值并根据题目描述判断pro\_type和pro\_pif\_id值是否正确，若出错则输出报错信息，否则正常执行插入操作。

代码过长，省略。

## 用户自定义函数

本小节的主要内容为熟悉用户自定义函数的创建和使用。

### 创建函数并在语句中使用它

这个函数体的内容本质仍然为select语句的使用。只需要额外定义一个变量储存查询结果并将其作为返回值即可。在begin-end中填写的代码如下：

declare result numeric(10,2);

select sum(b\_balance) into result

from bank\_card

where b\_c\_id = client\_id and b\_type = '储蓄卡';

return result;

调用该函数时的写法则与其他编程语言类似，在需要处进行调用即可。

select c\_id\_card, c\_name, get\_deposit(c\_id) as total\_deposit

from client

where get\_deposit(c\_id) >= 1000000

order by total\_deposit desc;

## 安全性控制

本小节主要内容为熟悉MySQL中的权限控制，即grant、revoke语句的使用。

### 用户和权限

#(1) 创建用户tom和jerry，初始密码均为'123456'；

create user tom identified by '123456';

create user jerry identified by '123456';

#(2) 授予用户tom查询客户的姓名，邮箱和电话的权限,且tom可转授权限；

grant select(c\_name,c\_mail,c\_phone)

on client

to tom

with grant option;

#(3) 授予用户jerry修改银行卡余额的权限；

grant update(b\_balance)

on bank\_card

to jerry;

#(4) 收回用户Cindy查询银行卡信息的权限。

revoke select

on bank\_card

from Cindy;

### 用户、角色和权限

# (1) 创建角色client\_manager和fund\_manager；

create role client\_manager;

create role fund\_manager;

# (2) 授予client\_manager对client表拥有select,insert,update的权限；

grant select,insert,update

on client

to client\_manager;

# (3) 授予client\_manager对bank\_card表拥有查询除银行卡余额外的select权限；

grant select(b\_number,b\_type,b\_c\_id)

on bank\_card

to client\_manager;

# (4) 授予fund\_manager对fund表的select,insert,update权限；

grant select,insert,update

on fund

to fund\_manager;

# (5) 将client\_manager的权限授予用户tom和jerry；

grant client\_manager to tom,jerry;

# (6) 将fund\_manager权限授予用户Cindy.

grant fund\_manager to Cindy;

## 并发控制与事务的隔离级别

本小节内容为通过使用sleep操作认为构造并行事务的调度，体验mysql不同的隔离级别下可能出现的异常情况。

### 并发事务控制和事务的隔离级别

较为简单，省略。

### 读脏

根据要求，事务1需要使读取余票发生在事务2修改后，可在读取前增加1秒的延迟。事务2需要在事务1读取后进行撤销，可在事务2更新后加入3秒的延迟再撤销。

### 不可重复读

根据要求，事务1的修改要在事务2的第一次读之后、第二次读之前发生，且事务1在事务2提交后进行一次查询。可在事务1修改前加入2秒延迟、事务1读取后提交前加入2秒延迟、事务1提交后加入2秒延迟，并在事务2两次读取之间加入2秒延迟。

### 幻读

幻读的发生原因为事务的两次读取期间有另一事务insert或delete了被读取的数据集。根据题目，事务2执行insert前有1秒的延迟，若要insert发生在事务1的两次读取之前，可在两次读取之间加入2秒的延迟。

### 主动加锁保证可重复读

为使事务1两次读取结果一样，可在事务1读取时加入share锁。为确保事务2的update操作发生在事务1的两次读取之间，可在事务1两次读取间加入2秒的延迟，在事务2update语句前加入1秒延迟。要在事务2提交后查询航班的余票，可在事务1查询前加入1秒的延迟。

### 可串行化

要使两个事务的并发执行结果与t1→t2串行执行结果相同，可以在事务1第一条select语句前加入1秒的延迟即可。

## 备份+日志：介质故障与数据库恢复

本小节的主要内容数据库的备份和恢复操作。

### 备份与恢复

较为简单，使用mysqldump进行备份，使用mysql自带恢复功能即可。

test1\_1.sh

mysqldump -h127.0.0.1 -uroot --databases residents > residents\_bak.sql;

test1\_2.sh

mysql -h127.0.0.1 -uroot < residents\_bak.sql;

### 备份+日志：介质故障的发生与数据库的恢复

按照实验指导中的内容填写相关命令即可。

test2\_1.sh

mysqldump -h127.0.0.1 -uroot --databases train > train\_bak.sql;

mysqladmin flush-logs;

test2\_2.sh

mysql -h127.0.0.1 -uroot < train\_bak.sql;

mysqlbinlog --no-defaults log/binlog.000018 | mysql -uroot -h127.0.0.1;

## 数据库设计与实现

### 从概念模型到MySQL实现

较为简单，按照逻辑ER图上的内容建立各个表，和外码约束，根据应用背景内的说明建立数据类型约束即可。

代码过长，予以省略。

### 从需求分析到逻辑模型

较为简单。根据描述绘制ER图（如图 2.1）并转化出各个关系模式即可。

movie(movie\_ID, title, type, runtime, release\_date, director,starring), 主码: movie\_id

customer(c\_ID, name, phone), 主码: c\_ID

hall(hall\_ID, mode, capacity, location), 主码: hall\_ID

schedule(schedule\_ID, date, time, price, number, movie\_ID, hall\_ID),主码: schedule\_ID, 外码: movie\_ID, hall\_ID

ticket(ticket\_ID, seat\_num, movie\_ID, c\_ID),主码: (ticket\_num),外码：(movie\_ID,c\_ID)

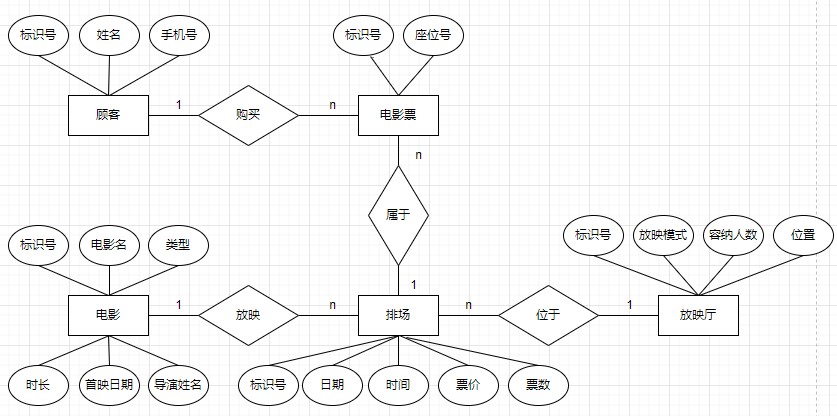


图 2.1 绘制的ER图

### 建模工具的使用

较为简单，在gitee平台下载给定的模型文件，使用MySQL WorkBench建模模块的Forward engineering功能即可导出SQL脚本。

## 数据库应用开发(JAVA篇)

### JDBC体系结构和简单的查询

根据讲解中描述的函数进行数据库的登录和查询操作。由于查询操作返回的是一组数据集，需采用游标的方式进行逐行读取。可用resultSet的next方法来判定游标是否到达数据集的结尾。

代码较长，省略。

### 用户登录

即用select语句查询用户名所对应的密码，若查询失败或密码错误，则报错并弹出错误提示。否则提示登陆成功。构造查询语句的语句为

String sql = "select c\_password from client where c\_mail = '"+loginName+"';";

### 添加新客户

添加新客户也即对数据库执行insert操作。其他操作与2.13.2一致。

### 银行卡销户

银行卡销户也即对bank\_card表实行delete操作。与2.13.3一致。

### 客户修改密码

客户修改密码。也即对表client执行update操作。需注意的是，在进行update操作前需要先判断修改密码的客户是否存在以及两次输入的密码是否一致。若客户不存在或密码不一致则需报错。剩余的登录等操作与2.13.2一致。

### 事务与转账操作

本题即为在java下实现MySQL的事务操作。主要需注意的问题为转账能否成功实现的判断。当转出或转入账号不存在、转出账号为信用卡或转出账号余额不足时，转账操作均无法实现，需要进行报错。

### 把稀疏表格转为键值对存储

较为简单，只需遍历稀疏表的每一行，提取其中的有效信息，按题目要求格式插入到新表中即可。

# 课程总结

本次课程实践的总体任务为熟悉MySQL的基本操作语句如create、alter、select等，以及一些较为高级的技巧，如存储过程与事务、触发器、函数等，以及MySQL与编程语言JAVA的结合使用。在这些任务中，除了select的一些较复杂查询和事务中的游标的使用和事务的创建以外均顺利完成。完成效果较好，使我了解和掌握了MySQL的基本操作，也对高级操作有了基本的认识和体验。但受限于时间和题目难度，未能将实验内容全部完成。

本课程实验实践采用线上的方式进行，使操作更为方便，也容易判断实验语句的正确与否。但部分实验关卡难度设计较不合理。如select查询大关中均有仅使用一条select的限制，这会给一些较复杂查询带来较大的复杂度，主要体现在一些子查询语句的反复使用，这对程序代码的可读性不利。游标关卡的难度也较大，主要体现在MySQL中多游标操作中，游标的结束信号并不对不同游标进行区分，这给程序实现带来了较大的难度也是促使我放弃这部分内容的重要原因。希望以后的实验设计中能够多注意难度梯度的设计，而不是直接进行大难度的任务。