华中科技大学

数据库系统原理实践报告

专　　业：	计算机科学与技术
班　　级：	CS2003 班
学　　号：	U202015375
姓　　名：	汪宇飞
指导教师：	瞿彬彬

分数	
教师签名	

2022 年 12 月 31 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

“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)。

课程依托头歌实践教学平台，实践课程 url 见相关课堂教师发布链接及其邀请码。实验环境为 Linux 操作系统下的 MySQL 8.0.28（主要为 8.028 版本，部分关卡使用 8.022 版本，使用中基本无差别）。在数据库应用开发环节，使用 JAVA 1.8。

2 任务实施过程与分析

2.1 数据库、表与完整性约束的定义(Create)

本小节共 6 个任务，要求参考所给出的指引实现连接数据库、创建数据库、创建表和创建表的主码约束、外码约束、CHECK 约束、DEFAULT 约束以及 UNIQUE 约束。

2.1.1 创建数据库

本关任务：创建数据库。

按照指引，使用 create database 语句创建用于 2022 年北京冬奥会信息系统的数据库:beijing2022。，代码如下所示。

```
1. create database beijing2022;
```

2.1.2 创建表与表的主码约束

本关任务：在指定的数据库中创建一个表，并为表指定主码。

按照指引，使用 create table 语句在指定的数据库 TestDb 中创建表 t_emp，并在代码第 4 行为主码 id 添加 primary key 的约束。

```
1. create database TestDb;
2. use TestDb;
3. create table t_emp(
4.     id int primary key,
5.     name varchar(32),
6.     deptId int,
7.     salary float
8. );
```

2.1.3 创建外码约束

本关任务：创建外码约束（参照完整性约束）。

按照指引，使用 create table 语句在指定的数据库 MyDb 中创建表 dept 和表 staff，并在代码第 14 行为表 staff 添加外码约束

```
1. create database MyDb;
2. use MyDb;
3. create table dept(
4.     deptNo int primary key,
5.     deptName varchar(32)
```

```
6. );
7. create table staff(
8.     staffNo int primary key,
9.     staffName varchar(32),
10.    gender char(1),
11.    dob date,
12.    salary numeric(8,2),
13.    deptNo int,
14.    constraint FK_staff_deptNo foreign key(deptNo) references dep
   t(deptNo)
15.);
```

2.1.4 CHECK 约束

本关任务：为表创建 CHECK 约束.

按照指引，为表中的 brand 和 price 添加 CHECK 约束，核心代码如下。

```
1. constraint CK_products_brand check(brand in ('A','B')),
2. constraint CK_products_price check(price>0)
```

2.1.5 DEFAULT 约束

本关任务：创建默认值约束(Default 约束),为字段指定默认值。

按照指引，为表中的 name 和 mz 添加 DEFAULT 约束，核心代码如下。

```
1. name varchar(32) not null,
2. mz char(16) default '汉族'
```

2.1.6 UNIQUE 约束

本关任务：创建唯一性约束(UNIQUE 约束),以保证字段取值的唯一性。

按照指引，为表中的 ID 添加 UNIQUE 约束，核心代码如下。

```
1. ID char(18) unique
```

2.2 表结构与完整性约束的修改(ALTER)

本小节共 4 个任务，要求使用 alter 语句实现表的修改，包括表名的修改、字段的修改、约束的修改等。

2.2.1 修改表名

根据实验指引，修改表的语法 alter table 如下。

```
1. ALTER TABLE 表名
```

```
2. [修改事项 [, 修改事项] ...]
```

本关任务：修改表的名称。

按照指引，使用 alter table 语句将表名修改为 my_table，代码如下。

```
1. alter table your_table rename to my_table;
```

2.2.2 添加与删除字段

本关任务：为表添加和删除字段。

按照指引，使用 alter table 语句删除表 orderDetail 中的列 orderDate、添加列 unitPrice，代码如下。

```
1. alter table orderDetail drop orderDate;
2. alter table orderDetail add unitPrice numeric(10,2);
```

2.2.3 修改字段

本关任务：修改字段。

按照指引，使用 alter table 语句修改 QQ 号的数据类型以及修改 weixin 的列名，代码如下。

```
1. alter table addressBook modify QQ char(12);
2. alter table addressBook rename column weixin to wechat;
```

2.2.4 添加、删除与修改约束

本关任务：添加、删除与修改约束。

按照指引，使用 alter table 语句为各表添加、删除以及修改各自相应的约束，代码如下。

```
1. #(1) 为表Staff 添加主码
2. alter table Staff add primary key(staffNo);
3. #(2) Dept.mgrStaffNo 是外码,对应的主码是Staff.staffNo,请添加这个外码,
   名字为FK_Dept_mgrStaffNo:
4. alter table Dept add constraint FK_Dept_mgrStaffNo foreign key(mg
   rStaffNo) references Staff(staffNo);
5. #(3) Staff.dept 是外码，对应的主码是Dept.deptNo. 请添加这个外码，名字
   为FK_Staff_dept:
6. alter table Staff add constraint FK_Staff_dept foreign key(dept)
   references Dept(deptNo);
7. #(4) 为表Staff 添加check 约束，规则为：gender 的值只能为F 或M；约束名
   为CK_Staff_gender:
```

```
8. alter table Staff add constraint CK_Staff_gender check (gender in
    ('F','M'));
9. #(5) 为表Dept 添加unique 约束：deptName 不允许重复。约束名为
   UN_Dept_deptName：
10.alter table Dept add constraint UN_Dept_deptName unique (deptName)
   ;
```

2.3 数据查询(Select)之一

本小节共 19 个任务，要求实现各自的查询任务，运用各种查询方法使用一条 select 查询语句实现每一个关卡的具体要求。

2.3.1 金融应用场景介绍，查询客户主要信息

本关任务：熟悉本实训数据库的内容；查询客户的主要信息。

按照指引，使用 select 语句查询所有的客户的名称、手机号和邮箱信息。查询结果按照客户编号排序，代码如下。

```
1. select c_name,c_phone,c_mail from client
2. order by c_id;
```

2.3.2 邮箱为 null 的用户

本关任务：查询客户表(client)中没有填写邮箱的客户的编号、名称、身份证号、手机号。

按照指引，使用 select 语句查询，使用 where 限定邮箱为 null 的用户，代码如下。

```
1. select c_id,c_name,c_id_card,c_phone
2. from client
3. where c_mail is null;
```

2.3.3 既买了保险又买了基金的用户

本关任务：查询既买了保险又买了基金的客户的名称和邮箱。

按照指引，使用 select 语句查询，使用 IN 的嵌套查询，分别查询买了保险和买了基金的客户的 id，然后使用 and 多条件查询获得交集，从而获得符合要求的客户，代码如下。

```
1. select c_name,c_mail,c_phone
2. from client
3. where c_id in(
4.     select pro_c_id
5.     from property
```

```
6.      where pro_type=2
7. )and c_id in(
8.      select pro_c_id
9.      from property
10.     where pro_type=3
11.)
12.order by c_id;
```

2.3.4 办理了储蓄卡的客户信息

本关任务：查询办理了储蓄卡的客户名称、手机号、银行卡号。

按照指引，使用 select 语句查询，使用多表连接查询同时在 client 和 bank_card 两个表中进行查询，代码如下。

```
1. select c_name,c_phone,b_number from client,bank_card
2. where c_id=b_c_id
3. and b_type="储蓄卡"
4. order by c_id;
```

2.3.5 每份金额在 30000~50000 之间的理财产品

本关任务：查询每份金额在 30000～50000 之间的理财产品，并对结果进行排序。

按照指引，使用 select 语句查询，使用 between 限制金额在 30000～50000 之间，使用 order by 来进行结果的排序。

```
1. select p_id,p_amount,p_year from finances_product
2. where p_amount between 30000 and 50000
3. order by p_amount,p_year desc;
```

2.3.6 商品收益的众数

本关任务：查询资产表中所有资产记录里商品收益的众数和它出现的次数。

按照指引，使用 select 语句查询，首先使用子查询用 count 函数获得所有资产记录里商品收益出现的次数，如果某个商品收益出现的次数大于等于所有出现的次数则为众数，代码如下。

```
1. select pro_income,count(*) as presence
2. from property
3. group by pro_income
4. having count(*) >= all(
5.      select count(*) from property
6.      group by pro_income
```

```
7. );
```

2.3.7 未购买任何理财产品的武汉居民

本关任务：查询未购买任何理财产品的武汉居民的信息。

按照指引，使用 select 语句查询，首先使用 like ‘4021%’来选择武汉居民，然后使用 not exist 谓词选择没有购买任何理财产品的用户，代码如下。

```
1. select c_name,c_phone,c_mail
2. from client
3. where c_id_card like '4201%' and not exists(
4.     select * from property
5.     where c_id=pro_c_id and pro_type=1
6. )order by c_id;
```

2.3.8 持有两张信用卡的用户

本关任务：查询在本行持有两张及以上信用卡的客户信息。

按照指引，使用 select 语句查询，首先在子查询中获得每个用户信用卡的数量，然后通过该信用卡数量获得持有两张及以上信用卡的客户信息，随后排序即可，代码如下。

```
1. select c_name,c_id_card,c_phone
2. from client,(
3.     select b_c_id,count(b_number)
4.     from bank_card
5.     where b_type="信用卡"
6.     group by b_c_id
7. )as a(b_c_id,b_num)
8. where c_id=b_c_id and b_num>=2
9. order by c_id;
```

2.3.9 购买了货币型基金的客户信息

本关任务：查询购买了货币型基金的客户信息。

按照指引，使用 select 语句查询，首先在子查询中使用多表查询获得购买了货币型基金的客户 id，然后根据 id 查询客户信息即可，代码如下。

```
1. select c_name,c_phone,c_mail
2. from client
3. where c_id in(
4.     select pro_c_id
5.     from property,fund
6.     where pro_pif_id=f_id
```

```
    and pro_c_id=c_id
    and pro_type=3
    and f_type="货币型"
)
order by c_id;
```

2.3.10 投资总收益前三名的客户

本关任务：查询投资总收益前三名的客户。

按照指引，使用 select 语句查询，首先将 client 表和 property 表根据用户 id 进行连接，选择 status 为可用的元组按照 id 进行分组统计，使用 sum 函数进行计算收益总和并排序，使用 limit 3 获得前三名的用户，代码如下。

```
select c_name,c_id_card,SUM(pro_income) as total_income
from client,property
where client.c_id = property.pro_c_id and property.pro_status like '可用'
group by c_name,c_id_card
order by total_income desc limit 3;
```

2.3.11 黄姓客户持卡数量

本关任务：查询黄姓用户的编号、名称、办理的银行卡的数量。

按照指引，使用 select 语句查询，首先使用子查询分组统计每一个用户 id 所拥有的银行卡数量，然后将 client 表和子查询获得的表根据用户 id 进行左外连接，并限定用户名为黄姓，使用 ifnull 函数对于银行卡数量为 0 的情况进行处理，最后进行排序，代码如下。

```
select c_id,c_name,ifnull(number_of_cards,0) number_of_cards
from client left outer join(
    select b_c_id,count(b_number) as number_of_cards
    from bank_card
    group by b_c_id
)as a(b_c_id,number_of_cards)
on c_id=b_c_id where c_name like "黄%"
order by number_of_cards desc,c_id;
```

2.3.12 客户理财、保险与基金投资总额

本关任务：查询客户理财、保险、基金投资金额的总和，并排序。

按照指引，使用 select 语句查询，首先需要分别计算理财、保险和基金投资的数额，将 property 表分别和 finance_product 表、insurance 表和 fund 表根据 id 进行左外连接，使用 sum 函数计算各自的数额；然后使用 union all 将这三个查

询结果合并获得派生表；将 client 表与派生表进行左外连接，使用 ifnull 函数处理总额为 0 的情况，最后进行排序，代码如下。

```
1. select c_name,c_id_card,ifnull(sum(total),0) as total_amount
2. from client left outer join (
3.     select pro_c_id,sum(pro_quantity*p_amount) as total
4.     from property left outer join finances_product on(pro_pif_id
   = p_id)
5.     where pro_type = 1
6.     group by pro_c_id
7.
8.     union all
9.
10.    select pro_c_id,sum(pro_quantity*i_amount) as total
11.    from property left outer join insurance on(pro_pif_id = i_id)
12.    where pro_type = 2
13.    group by pro_c_id
14.
15.    union all
16.
17.    select pro_c_id,sum(pro_quantity*f_amount) as total
18.    from property left outer join fund on(pro_pif_id = f_id)
19.    where pro_type = 3
20.    group by pro_c_id
21.) as P on (c_id = pro_c_id)
22.group by c_id
23.order by total_amount desc;
```

2.3.13 客户总资产

本关任务：查询客户在本行的总资产。

按照指引，使用 select 语句查询，客户总资产为储蓄卡总余额，投资总额，投资总收益的和，再扣除信用卡透支的总金额。分别查询储蓄卡的总额、信用卡的总额以及投资收益总额，将 client 表基于客户 id 左外连接查询所得的三个表，使用 ifnull(bc_dep_property,0)-ifnull(bc_cre_property,0)+ifnull(total_income,0)+ifnull(total_amount,0)计算客户总资产，具体代码如下。

```
1. select c_id,c_name,ifnull(bc_dep_property,0)-ifnull(bc_cre_proper
   ty,0)+ifnull(total_income,0)+ifnull(total_amount,0) as total_prop
   erty
2. from client left outer join (
3.     select b_c_id as bc_dep_id,sum(b_balance) as bc_dep_property
4.     from bank_card
```

```
    where b_type like '储蓄卡'
    group by b_c_id
) as dep on(bc_dep_id = c_id)
left outer join (
    select b_c_id as bc_cre_id,sum(b_balance) as bc_cre_property
    from bank_card
    where b_type like '信用卡'
    group by b_c_id
) as cre on(bc_cre_id = c_id)
left outer join (
    select pro_c_id as pro_shouxi_id,sum(pro_income) as total_income
    from property
    group by pro_c_id
) as shouyi on(pro_shouxi_id = c_id)
left outer join (
    select c_id as pro_amount_id,ifnull(sum(total),0) as total_amount
    from client left outer join (
        select pro_c_id,sum(pro_quantity*p_amount) as total
        from property left outer join finances_product on(pro_pif_id = p_id)
        where pro_type = 1
        group by pro_c_id
        union all
        select pro_c_id,sum(pro_quantity*i_amount) as total
        from property left outer join insurance on(pro_pif_id = i_id)
        where pro_type = 2
        group by pro_c_id
        union all
        select pro_c_id,sum(pro_quantity*f_amount) as total
        from property left outer join fund on(pro_pif_id = f_id)
        where pro_type = 3
        group by pro_c_id
    ) as P on (c_id = pro_c_id)
    group by c_id
) as amount on(pro_amount_id = c_id)
order by c_id;
```

2.3.14 第 N 高问题

该关卡任务已完成，实验情况本报告略过。

2.3.15 基金收益两种方式排名

本关任务：对客户基金投资收益实现两种方式的排名次。

按照指引，使用 select 语句查询，定义相关变量 currank、rankcount 和 rank_revenue 进行排序的实现，具体代码如下。

```
-- (1) 基金总收益排名(名次不连续)
select pro_c_id,total_revenue,myrank as 'rank'
from (
 select pro_c_id,total_revenue,@rankcount := @rankcount + 1,if(@rank_revenue = total_revenue, @currank, @currank := @rankcount) as myrank,@rank_revenue := total_revenue
 from (
  select pro_c_id,sum(pro_income) as total_revenue
  from property
  where pro_type = 3
  group by pro_c_id
  order by total_revenue desc,pro_c_id
 ) as i(pro_c_id,total_revenue),(select @currank := 0,@rankcount := 0,@rank_revenue := null) as r
) as temp;

-- (2) 基金总收益排名(名次连续)
select pro_c_id,total_revenue,myrank as 'rank'
from (
 select pro_c_id,total_revenue,if(@rank_revenue = total_revenue, @currank, @currank := @currank + 1) as myrank,@rank_revenue := total_revenue
 from (
  select pro_c_id,sum(pro_income) as total_revenue
  from property
  where pro_type = 3
  group by pro_c_id
  order by total_revenue desc,pro_c_id
 ) as i(pro_c_id,total_revenue),(select @currank := 0,@rank_revenue := null) as r
) as temp;
```

2.3.16 持有完全相同基金组合的客户

本关任务：查询持有完全相同基金组合的客户。

按照指引，使用 select 语句查询，对基金组合进行两次查询，分别为 p1 和 p2，对于 p1 和 p2 中基金组合相同但用户 id 不同的，即是符合要求的客户，具体代码如下所示。

```
select pro_c_id1 as c_id1,pro_c_id2 as c_id2
from(
 select pro_c_id,group_concat(pro_pif_id order by pro_pif_id SEPARATOR ' ') as pifset
 from property
 where pro_type = 3
 group by pro_c_id
) as p1(pro_c_id1,pifset1),
(
 select pro_c_id,group_concat(pro_pif_id order by pro_pif_id SEPARATOR ' ') as pifset
 from property
 where pro_type = 3
 group by pro_c_id
) as p2(pro_c_id2,pifset2)
where p1.pifset1 = p2.pifset2 and p1.pro_c_id1 < p2.pro_c_id2
order by pro_c_id1;
```

2.3.17 购买基金的高峰期

该关卡任务已完成，实验情况本报告略过。

2.3.18 至少有一张信用卡余额超过 5000 元的客户信用卡总余额

本关任务：至少有一张信用卡余额超过 5000 元的客户信用卡总金额。

按照指引，使用 select 语句查询，使用子查询获得余额超过 5000 的信用卡的卡号，使用多表查询获得拥有这种卡的客户的信用卡总余额，代码如下。

```
select b_c_id,sum(b_balance) as credit_card_amount
from bank_card,(
 select b_c_id
 from bank_card
 where b_type = '信用卡' and b_balance >= 5000
 order by b_c_id
) as id(sc_id)
where b_type = '信用卡' and bank_card.b_c_id = id.sc_id
group by b_c_id
order by b_c_id;
```

2.3.19 以日历表格式显示每日基金购买总金额

该关卡任务已完成，实验情况本报告略过。

2.4 数据查询(Select)之二

本小节共 6 个任务，是在 2.3 的基础之上另外再添加的 6 个查询任务。

2.4.1 查询销售总额前三的理财产品

该关卡任务已完成，实验情况本报告略过。

2.4.2 投资积极且偏好理财类产品的客户

该任务关卡跳过。

2.4.3 查询购买了所有畅销理财产品的客户

该关卡任务已完成，实验情况本报告略过。

2.4.4 查找相似的理财产品

该关卡任务已完成，实验情况本报告略过。

2.4.5 查询任意两个客户的相同理财产品数

该关卡任务已完成，实验情况本报告略过。

2.4.6 查找相似的理财用户

该关卡任务已完成，实验情况本报告略过。

2.5 数据的插入、修改与删除(Insert,Update,Delete)

本小节共 6 个任务，要求使用 insert、update 和 delete 语句进行数据的插入、修改和删除操作。

2.5.1 插入多条完整的客户信息

本关任务：向客户表 client 插入数据。

使用 insert 语句将客户信息进行插入，代码如下。

```
insert into client
values('1','林惠雯','960323053@qq.com','411014196712130323','15609032348','Mop5UPkl');
insert into client
values('2', '吴婉瑜','1613230826@gmail.com','420152196802131323','17605132307','QUTPhxgVNlXtMxN');
insert into client
values('3','蔡贞仪','252323341@foxmail.com','160347199005222323','17763232321','Bwe3gyhEErJ7');
```

2.5.2 插入不完整的客户信息

本关任务：向客户表 client 插入一条数据不全的记录。

使用 insert 语句将不完整的客户信息插入，代码如下。

```
insert into client(c_id,c_name,c_phone,c_id_card,c_password)
values('33','蔡依婷
','18820762130','350972199204227621','MKwEuc1sc6');
```

2.5.3 批量插入数据

本关任务：向客户表 client 批量插入数据。

使用 insert 语句将全部客户信息进行插入，代码如下。

```
insert into client
select * from new_client;
```

2.5.4 删除没有银行卡的客户信息

本关任务：删除在本行没有银行卡的客户信息。

使用 delete 语句将 client 中没有银行卡的客户信息进行删除，代码如下。

```
delete from client
where c_id not in(
    select b_c_id from bank_card
);
```

2.5.5 冻结客户资产

本关任务：冻结客户的投资资产。

使用 update 语句将客户的状态更新为冻结，代码如下。

```
update property,client
set pro_status='冻结'
where pro_c_id=c_id and c_phone='13686431238';
```

2.5.6 连接更新

本关任务：根据客户表的内容修改资产表的内容。

使用 update 语句对 property 表和 client 表进行更新，代码如下。

```
update property,client
set pro_id_card=c_id_card
where pro_c_id=c_id;
```

2.6 视图

本小节共 2 个任务，要求实现视图的创建以及基于视图的查询，

2.6.1 创建所有保险资产的详细记录视图

本关任务：创建所有保险资产的详细记录视图。

使用 create view 语句，将查询获得的结果建立视图，代码如下。

```
create view v_insurance_detail
as select c_name,c_id_card,i_name,i_project,pro_status,pro_quanti
ty,i_amount,i_year,pro_income,pro_purchase_time
from client,property,insurance
where c_id=pro_c_id and pro_pif_id=i_id and pro_type=2;
```

2.6.2 基于视图的查询

本关任务：基于视图 v_insurance_detail 查询每位客户保险资产的总额和保险总收益。

使用 select 语句进行查询，使用 sum 函数对于总额进行计算，最后排序，代码如下。

```
select c_name,c_id_card,sum(pro_quantity*i_amount) as insurance_t
otal_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;
```

2.7 存储过程与事务

本小节共 3 个任务，要求分别实现使用流程控制语句、游标和事务的存储过程。

2.7.1 使用流程控制语句的存储过程

本关任务：创建一个存储过程，向表 fibonacci 插入斐波拉契数列的前 n 项。

```
drop procedure if exists sp_fibonacci;
delimiter $$
create procedure sp_fibonacci(in m int)
begin
declare n,fibn,a,b int;
set n = 2,a = 0,b = 1;
if m > 0 then
 insert into fibonacci values(0,0);
```

```
9. end if;
10.if m > 1 then
11. insert into fibonacci values(1,1);
12.end if;
13.while m > n do
14. set fibn = a + b;
15. insert into fibonacci values(n,fibn);
16. set a = b;
17. set b = fibn;
18. set n = n + 1;
19.end while;
20.end $$
21.delimiter ;
```

2.7.2 使用游标的存储过程

该任务关卡跳过。

2.7.3 使用事务的存储过程

本关任务：编写实现转账功能的存储过程。

```
1. delimiter $$
2. create procedure sp_transfer(
3.       IN applicant_id int,
4.       IN source_card_id char(30),
5.       IN receiver_id int,
6.       IN dest_card_id char(30),
7.       IN amount numeric(10,2),
8.       OUT return_code int)
9. BEGIN
10.set autocommit = off;
11.start transaction;
12.update bank_card set b_balance = b_balance - amount where b_numbe
   r = source_card_id and b_c_id = applicant_id and b_type = '储蓄卡
   ';
13.update bank_card set b_balance = b_balance + amount where b_numbe
   r = dest_card_id and b_c_id = receiver_id and b_type = '储蓄卡';
14.update bank_card set b_balance = b_balance - amount where b_numbe
   r = dest_card_id and b_c_id = receiver_id and b_type = '信用卡';
15.
16.if not exists(select * from bank_card where b_number = source_car
   d_id and b_c_id = applicant_id and b_type = '储蓄卡
   ' and b_balance >= 0) then
17. set return_code = 0;
18. rollback;
```

```
elseif not exists(select * from bank_card where b_number = dest_card_id and b_c_id = receiver_id) then
 set return_code = 0;
 rollback;
else
 set return_code = 1;
 commit;
end if;
END$$
delimiter ;
```

2.8 触发器

本小节共 1 个任务，要求实现一个触发器，实现合法性检查，即在表上进行 insert、delete、update 等操作时触发器被触发执行判断操作是否合法。

2.8.1 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

本关任务：为表 property(资产表)编写一个触发器，以实现以下完整性业务规则：

- 如果 pro_type = 1，则 pro_pif_id 只能引用 finances_product 表的 p_id；
- 如果 pro_type = 2，则 pro_pif_id 只能引用 insurance 表的 i_id；
- 如果 pro_type = 3，则 pro_pif_id 只能引用 fund 表的 f_id；
- pro_type 不接受(1,2,3)以外的值。

按照指引，参照实验所提供的创建触发器的语句，实现所要求的触发器。对于 pro_type 为 1、2、3 以外的值，触发器进行非法提示，如下列代码 8-11 所示；对于 pro_type 为 1、2、3 之内某一值，则按照对应的 id 是否存在进行相应的提示，如下列代码 13-26 所示。

```
use finance1;
drop trigger if exists before_property_inserted;
delimiter $$
CREATE TRIGGER before_property_inserted BEFORE INSERT ON property
FOR EACH ROW
BEGIN
declare msg varchar(40);
if new.pro_type <> 1 and new.pro_type <> 2 and new.pro_type <> 3 then
 set msg = concat('type ',new.pro_type,' is illegal!');
signal sqlstate '45000' set message_text = msg;
end if;

```

```
13.if new.pro_type = 1 and not exists(select * from finances_product
    where finances_product.p_id = new.pro_pif_id) then
14. set msg = concat('finances product #',new.pro_pif_id,' not found!
   ');
15. signal sqlstate '45000' set message_text = msg;
16.end if;
17.
18.if new.pro_type = 2 and not exists(select * from insurance where
   insurance.i_id = new.pro_pif_id) then
19. set msg = concat('insurance #',new.pro_pif_id,' not found!');
20. signal sqlstate '45000' set message_text = msg;
21.end if;
22.
23.if new.pro_type = 3 and not exists(select * from fund where fund.
   f_id = new.pro_pif_id) then
24. set msg = concat('fund #',new.pro_pif_id,' not found!');
25. signal sqlstate '45000' set message_text = msg;
26.end if;
27.
28.END$$
29.
30.delimiter ;
```

2.9 用户自定义函数

本小节共 1 个任务，要求完成用户自定义函数的实现并使用自定义的函数完成相应功能。

2.9.1 创建函数并在语句中使用它

本关任务：编写一个依据客户编号计算其在本金融机构的存储总额的函数，并在 SELECT 语句使用这个函数。

按照指引，根据 create function 语句的语法，在给出的待完成的代码中填入用户自定义函数体，如图下代码第 9-14 行所示，使用 select into 语句传递参数进行查询，用 return 语句返回所查询的结果即指定用户的储蓄卡余额之和。

在完成 get_deposit 函数的函数体定义之后，调用该函数完成预期功能，如下代码 18-21 行所示，将所定义的函数运用于 select 查询语句中，使用 where 语句进行限定存款总额在 100 万以上的客户，使用 order by 语句按照存储总额进行排序。

```
1. use finance1;
2. set global log_bin_trust_function_creators=1;
3. drop function IF EXISTS get_deposit;
```

```
4.
5. delimiter $$
6. create function get_deposit(client_id int)
7. returns numeric(10,2)
8. begin
9.     declare deposit numeric(10,2);
10.    select sum(b_balance) into deposit
11.    from bank_card
12.    where b_type = '储蓄卡' and b_c_id = client_id
13.    group by b_c_id;
14.    return deposit;
15.end$$
16.delimiter ;
17.
18.select c_id_card,c_name,get_deposit(c_id) as total_deposit
19.from client
20.where get_deposit(c_id) >= 1000000
21.order by get_deposit(c_id) desc;
```

2.10 安全性控制

本小节共 2 个任务，要求实现对数据库中用户、角色和权限进行相关安全性控制操作。

2.10.1 用户和权限

本关任务：在金融应用场景数据库环境中，创建用户，并给用户授予指定的权限。

使用 create 语句进行角色的创建，使用 grant 语句进行权限的授予，使用 revoke 语句收回权限，代码如下。

```
1. #(1) 创建用户 tom 和 jerry，初始密码均为'123456';
2. create user 'tom' identified by '123456';
3. create user 'jerry' identified by '123456';
4. #(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;
5. grant select(c_name,c_phone,c_mail)
6. on client
7. to tom
8. with grant option;
9. #(3) 授予用户 jerry 修改银行卡余额的权限;
10.grant update(b_balance)
11.on bank_card
12.to jerry;
13.#(4) 收回用户 Cindy 查询银行卡信息的权限。
```

```
14.revoke select
15.on bank_card
16.from Cindy;
```

2.10.2 用户、角色和权限

本关任务：创建角色，授予角色一组权限，并将角色代表的权限授予指定的一组用户。

使用 create 语句进行角色的创建，使用 grant 语句进行权限的授予，代码如下。

```
1. # (1) 创建角色 client_manager 和 fund_manager;
2. create role client_manager;
3. create role fund_manager;
4. # (2) 授予 client_manager 对 client 表拥有 select,insert,update 的权限;
5. grant select,insert,update
6. on client
7. to client_manager;
8. # (3) 授予 client_manager 对 bank_card 表拥有查询除银行卡余额外的 select
   权限;
9. grant select(b_number,b_type,b_c_id)
10.on bank_card
11.to client_manager;
12.# (4) 授予 fund_manager 对 fund 表的 select,insert,update 权限;
13.grant select,insert,update
14.on fund
15.to fund_manager;
16.# (5) 将 client_manager 的权限授予用户 tom 和 jerry;
17.grant client_manager
18.to tom,jerry;
19.# (6) 将 fund_manager 权限授予用户 Cindy.
20.grant fund_manager
21.to Cindy;
```

2.11 并发控制与事务的隔离级别

本小节共 6 个任务，要求实现并发控制与事务的隔离级别相关内容，包括读脏、不可重复读、幻读、主动加锁保证可重复读、可串行化等内容。

2.11.1 并发控制与事务的隔离级别

该任务关卡跳过。

2.11.2 读脏

该任务关卡跳过。

2.11.3 不可重复读

该任务关卡跳过。

2.11.4 幻读

该任务关卡跳过。

2.11.5 主动加锁保证可重复读

该任务关卡跳过。

2.11.6 可串行化

该任务关卡跳过。

2.12 备份+日志：介质故障与数据库恢复

本小节共 2 个任务，要求实现介质故障与数据库恢复相关内容。

2.12.1 备份与恢复

该任务关卡跳过。

2.12.2 备份+日志：介质故障的发生与数据库的恢复

该任务关卡跳过。

2.13 数据库设计与实现

本小节共 3 个任务，要求实现数据库的设计与实现相关的内容，包括从概念模型到 MySQL 模型、从需求分析到逻辑模型以及建模工具的使用。除此之外也对制约因素分析与设计和工程师责任及其分析进行了相关的讨论。

2.13.1 从概念模型到 MySQL 实现

该关卡任务已完成，实验情况本报告略过。

2.13.2 从需求分析到逻辑模型

该任务关卡跳过。

2.13.3 建模工具的使用

该关卡任务已完成，实验情况本报告略过。

2.13.4 制约因素分析与设计

在数据库的设计与实现过程中有诸多制约因素需要考虑并进行实现。例如最基础的数据实体完整性、参照完整性以及用户定义完整性，需要基于实际情况设置主码约束、外码约束、CHECK 约束等约束，实现有效且全面的完整性约束。

除此之外也需要考虑安全性、隐私性和保密性的问题，对于一些重要信息，如用户个人隐私、身份证号、密码等信息，需要考虑如何对其进行保护，采用加密等方法。

2.13.5 工程师责任及其分析

作为工程师，不仅需要使用自身技能完整高效地实现所分配的具体任务，实现数据结构、系统结构等设计，开发所预期的系统，还需要基于所处的环境、基于开发情况承担社会责任。

工程师在开发设计时，需要尽可能考虑到受众用户的具体情况以及开发产品运行的具体环境。

对于开发方而言，工程师最重要的责任是保证开发产品的完善，能够满足开发方的各个需求并能够正确处理开发方的各方面问题。在保证了开发的完整性和正确性的基础上，工程师也应当考虑所开发的开销成本，尽可能使用高效便捷的开发方法，实现开发成本的降低以及开发效率的提升。

对于用户而言，工程师所开发的产品应当是高效、简明、易上手的，应该有利于各个年龄、种族、学历等的用户快速了解并熟悉。除此之外，也应当是健康、安全、遵守法律法规、尊重各民族文化的，工程师所开发的产品应当尽可能排除对用户有害的信息，保护用户个人隐私信息，尊重用户的文化，对于潜在的风险，工程师应当进行排查并修护。

2.14 数据库应用开发(JAVA 篇)

本小节共 7 个任务，要求基于 JDBC 应用实现数据库应用的开发，包含简单的查询、用户登录、新用户添加、银行卡销户、用户修改密码、事务与转账操作和把稀疏表格转换为键值对存储等。

2.16.1 JDBC 体系结构和简单的查询

本关任务：查询 client 表中邮箱非空的客户信息，列出客户姓名，邮箱和电话。

根据指引，需要构建 JDBC 应用。首先导入 JDBC 包，使用 Class.forName 方法注册驱动程序，如代码第 7 行所示；然后使用 DriverManager.getConnection

方法进行数据库连接，实现 URL、用户名和密码的传递，如代码第 11 行所示；其次需要进行 SQL 操作，使用 connection.createStatement 方法创建接口并构造相应的操作字符串，之后使用 statement.executeQuery 方法进行 SQL 语句的执行，如代码 12-14 行所示；将查询所得结果进行 next 方法遍历，如代码第 16 行所示；操作全部完成之后使用 close 方法将创建的对象全部关闭。

具体代码如下。

```
1. public class Client {
2.     public static void main(String[] args) {
3.         Connection connection = null;
4.         Statement statement = null;
5.         ResultSet resultSet = null;
6.         try {
7.             Class.forName("com.mysql.cj.jdbc.Driver");
8.             String URL = "jdbc:mysql://127.0.0.1:3306/finance?use
   Unicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=U
   TC";
9.             String USER = "root";
10.            String PASS = "123123";
11.            connection = DriverManager.getConnection(URL, USER, P
   ASS);
12.            statement = connection.createStatement();
13.            String SQL = "select * from client where c_mail is no
   t null;";
14.            resultSet = statement.executeQuery(SQL);
15.            System.out.println("姓名\t 邮箱\t\t\t\t 电话");
16.            while (resultSet.next()) {
17.                System.out.print(resultSet.getString("c_name")+'\
   t');
18.                System.out.print(resultSet.getString("c_mail")+"\
   t\t");
19.                System.out.println(resultSet.getString("c_phone"))
   ;
20.            }
21.         } catch (ClassNotFoundException e) {
22.            System.out.println("Sorry,can`t find the JDBC Driver!
   ");
23.            e.printStackTrace();
24.        } catch (SQLException throwables) {
25.            throwables.printStackTrace();
26.        } finally {
27.            try {
28.                if (resultSet != null) {resultSet.close();}
```

```
                if (statement != null) {statement.close();}
                if (connection != null) {connection.close();}
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
    }
}
```

2.16.2 用户登录

本关任务：编写客户登录程序，提示用户输入邮箱和密码，并判断正确性，给出适当的提示信息。

与 2.16.1 类似，提示用户输入用户名和密码，将用户名用于查询的 SQL 字符串中，将查询获得的 resultSet 的密码与输入的密码进行对比，若相同则可以成功登录，否则提示错误，具体代码如下。

```
public class Login {
    public static void main(String[] args) {
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
        Scanner input = new Scanner(System.in);
        System.out.print("请输入用户名：");
        String loginName = input.nextLine();
        System.out.print("请输入密码：");
        String loginPass = input.nextLine();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String userName = "root";
            String passWord = "123123";
            String url = "jdbc:mysql://127.0.0.1:3306/finance?use
Unicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=U
TC";
            connection = DriverManager.getConnection(url, userNam
e, passWord);
            statement = connection.createStatement();
            String SQL = "select * from client where c_mail = '"+
loginName+"' ;";
            resultSet = statement.executeQuery(SQL);
            if(resultSet.next()){
                if(resultSet.getString("c_password").equals(login
Pass))
                    System.out.println("登录成功。");
```

```
                else System.out.println("用户名或密码错误！");
            }
            else System.out.println("用户名或密码错误！");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            try {
                if (resultSet != null) {resultSet.close();}
                if (statement != null) {statement.close();}
                if (connection != null) {connection.close();}
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
    }
}
```

2.16.3 添加新用户

本关任务：编程完成向 client(客户表)插入记录的方法。

自定义一个 insertClient 方法，将 SQL 字符串设置为 insert into 语句，将各个参数通过 setString 方法进行设置，从而实现新用户的添加，代码如下所示。

```
public static int insertClient(Connection connection, int c_id, String c_name, String c_mail,String c_id_card, String c_phone, String c_password){
        PreparedStatement pstmt = null;
        int n = 0;
        try {
            String sql = "insert into client values (?,?,?,?,?,?);";
            pstmt = connection.prepareStatement(sql);
            pstmt.setInt(1,c_id);
            pstmt.setString(2,c_name);
            pstmt.setString(3,c_mail);
            pstmt.setString(4,c_id_card);
            pstmt.setString(5,c_phone);
            pstmt.setString(6,c_password);
            n = pstmt.executeUpdate();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
```

```
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return n;
    }
```

2.16.4 银行卡销户

本关任务：编写一个能删除指定客户编号的指定银行卡号的方法。

自定义一个 removeBankCard 方法，将 SQL 字符串设置为 delete from 语句，将各个参数通过 setString 方法进行设置，从而实现银行卡的销户，代码如下所示。

```
public static int removeBankCard(Connection connection,int b_c_id,
 String b_number){
        PreparedStatement pstmt = null;
        int n = 0;
        try {
            String sql = "delete from bank_card where b_c_id = ?
and b_number = ?;";
            pstmt = connection.prepareStatement(sql);
            pstmt.setInt(1,b_c_id);
            pstmt.setString(2,b_number);
            n = pstmt.executeUpdate();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        return n;
    }
```

2.16.5 客户修改密码

该关卡任务已完成，实验情况本报告略过。

2.16.6 事务与转账操作

该关卡任务已完成，实验情况本报告略过。

2.16.7 把稀疏表格转为键值对存储

该关卡任务已完成，实验情况本报告略过。

2.15 数据库的索引 B+树实现

本小节共 5 个任务，要求实现数据库 B+数相关内容。

2.15.1 BPlusTreePage 的设计

该任务关卡跳过。

2.15.2 BPlusTreeInternalPage 的设计

该任务关卡跳过。

2.15.3 BPlusTreeLeafPage 的设计

该任务关卡跳过。

2.15.4 B+树索引：Insert

该任务关卡跳过。

2.15.5 B+树索引：Remove

该任务关卡跳过。

3 课程总结

本次数据库系统原理实践以理论课知识为基础，以 MySQL 为例，设计了一系列的数据库相关应用实训任务。完成了数据库、表与完整性约束的定义，表结构与完整性约束的修改，数据查询之一，数据的插入、修改与删除，视图，触发器，用户自定义函数，安全性控制以及数据库应用开发（JAVA 篇）的全部关卡。选择性地完成了数据查询之二，存储过程与事务，数据库设计与实现的部分关卡。最终的 educoder 平台得分为 110 分，达到了实验预期的分数要求。

完成了数据库、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部分，实现了对数据库、表、视图等的创建和增删改查等操作，实现了约束的增加与删除，实现了用户自定义函数和触发器的实现与应用。

完成了数据查询，数据插入、删除与修改等数据处理相关任务部分，基于实验所给出的金融应用场景，实现了按照各种不同的要求进行查询，运用多种查询方法进行数据的查询。

完成了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，实现了用户和角色的创建以及权限的授予和收回。

完成了数据库的设计与实现，实现了从概念模型到 MySQL 的实现，了解并使用了相关建模工具，并对制约因素分析与设计和工程师责任及其分析进行了相关的讨论。

完成了数据库应用系统的开发（JAVA 篇），实现了基于 JAVA 语言的数据库系统的操作，实现了 JDBC 应用的构建，并基于该应用使用了不同的 SQL 语言进行了多种具体数据库增删改查操作，使用 JAVA 语言和 SQL 语言结合自定义了方法并使用。

总体而言本次课程实践任务工作量较多、难度较为适中，不过老师给了充足且自由的时间来完成，且老师和主教也积极地为我们解答问题，因此能够顺利完成。本次实践除了理论课的知识的应用以外，也加入了许多课外的知识，例如使用 JAVA 进行数据库应用开发等，这些课外知识的应用向我们展示了数据库系统的广发应用以及强大的兼容性。

对于本次课程实践的建议是，可以适当对每一个实训任务进行合理的实践分配，而不是所有的实训任务的截止日期都相同，这样有利于促进学生的完成，避免学生将所有任务堆积到最后才一起做。