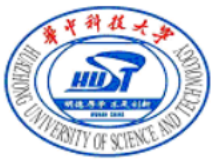

华中科技大学

数据库系统原理实践报告

专　　业：计算机科学与技术

班　　级：

学　　号：

姓　　名：

指导教师：

分数	
教师签名	

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配合数据库系统原理课程开展，旨在将理论运用于实践，加深对数据库原理的理解，提高个人处理数据库相关问题的能力，熟悉与掌握数据库编程思路以及数据库相关功能的使用。整体将 MYSQL 作为主要编程语句，设计了一系列系统实训任务，依托头歌实践教学平台开展实验教学。

配置整体实验环境为 Linux 操作系统下的 MySQL 8.0.28，部分实验关卡使用 8.0.22 版本。在数据库应用开发实验关卡，使用 JAVA 1.8 版本进行高级语言的数据库应用系统开发。

具体实验内容涉及以下任务：

（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）数据查询与数据插入、删除与修改等数据处理相关操作；

（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具体实现与应用；

（4）数据库的设计与实现；

（5）利用 JAVA 高级语言进行数据库应用系统的开发。

将上述任务分解为 13 个实训关卡，每个实训关卡实现相关任务的具体细节操作。具体分解为数据库、表、完整性约束的创建与修改，数据查询，数据的插入、修改与删除，与视图相关的操作，存储过程与事务，触发器，用户自定义函数，安全性控制，并发控制与事务的隔离级别，介质故障与数据库恢复，数据库设计与实现，JAVA 实现数据库应用开发。

2　任务实施过程与分析

2.1 数据库、表、完整性约束的创建与修改

本任务环节要求使用 MYSQL 提供的 DDL 数据定义语句进行数据库与表的创建，数据完整性约束条件的定义，以及表结构和约束条件的修改。具体使用 CREATE 语句、ALTER 语句、CONSTRAINT 完整性约束命名子句完成以上内容。

2.1.1 数据库、表与完整性约束的定义(Create)

（1）创建数据库

使用 create database <数据库名>进行数据库创建。完整代码为：

```
create database beijing2022;
```

（2）创建表及表的主码约束

使用 create table <表名> (<列名><数据类型>[列级完整性约束条件]...[,<表级约束条件>])进行表的创建和完整性约束定义。其中主码可采用列级完整性约束定义，即 id int PRIMARY KEY。完整代码为：

```
create database TeseDb;
use TestDb;
create table t_emp(
    id INT PRIMARY KEY,
    name VARCHAR(32),
    deptId INT,
    salary FLOAT
);
desc t_emp;
```

（3）创建外码约束(foreign key)

使用完整性约束命名子句对外码创建参照完整性约束，具体实现为 constraint <约束条件名> <完整性约束条件>。关键代码为：

```
constraint FK_staff_deptNo foreign key(deptNo) references dept(deptNo)
```

（4）CHECK 约束

使用完整性约束命名子句进行 CHECK 约束定义。关键代码为：

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

（5）DEFAULT 约束

使用列级完整性约束条件实现 DEFAULT 约束，且只能通过列级约束定义。当在表中添加元组时，若对指定列无具体值赋值，则自动为该列数据填写默认值。关键代码为：

```
mz char(16) default '汉族'
```

（6）UNIQUE 约束

使用列级完整性约束条件实现 UNIQUE 约束。创建 UNIQUE 约束用以保证字段取值的唯一性。关键代码为：

```
ID char(18) unique
```

2.1.2 表结构与完整性约束的修改(ALTER)

（1）修改表名

使用 ALTER 语句修改表名，具体实现为 alter table <待修改表名> rename [TO|AS] <新表名>。关键代码为：

```
alter table your_table rename AS my_table;
```

（2）添加与删除字段

使用 ALTER 语句为表添加新的字段，具体实现为 alter table <表名> add [column] <列名> <数据类型> [列级完整性约束条件] [FIRST|AFTER 指定列名]，其中 FIRST 指示添加到表首，AFTER 指示添加到指定列后。使用 ALTER 语句为表删除字段，具体实现为 alter table <表名> drop [column] <列名>。关键代码为：

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

（3）修改字段

使用 ALTER 语句修改字段。若只修改表中列名，具体实现为 alter table <表名> rename column <待修改列名> to <新列名>；若只修改字段的数据类型和约束，具体实现为 alter table <表名> modify [column] <列名><数据类型> [列级完整性约束条件] [FIRST|AFTER 指定列名]；若同时修改，则使用 alter table <表名> change <修改内容>进行修改。关键代码为：

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

（4）添加、删除与修改约束

对于主码约束的修改，使用 alter table <表名> drop PRIMARY KEY;进行主码

的删除，使用 alter table <表名> add PRIMARY KEY(主码列 1[,主码列 2,...]);进行主码的添加；对于外码约束和用户自定义约束的修改，则使用 alter table <表名> (add|drop) <完整性约束子句>;进行约束的添加与删除。关键代码为：

```
alter table Staff add primary key(staffNo);
alter table Staff
add constraint FK_Staff_dept foreign key(dept) references Dept(deptNo);
```

2.2 数据查询(Select)

数据查询作为数据库的核心操作之一，需要熟练掌握。本任务环节需要通过使用一条 SQL 语句进行 SELECT 数据查询操作。需要掌握各子句的组成和使用，掌握单表查询、连接查询、嵌套查询、集合查询和基于派生表查询的方法，掌握聚集函数、日期函数等的使用。

2.2.1 查询客户主要信息

进行单表查询，并使用 ORDER BY 子句进行查询结果的排序。完整代码为：

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

2.2.2 邮箱为 null 的客户

在 WHERE 子句中，使用 is null 判断属性值是否为空。完整代码为：

```
select c_id,c_name,c_id_card,c_phone
from client
where c_mail is null;
```

2.2.3 既买了保险又买了基金的客户

使用带 IN 谓词的子查询，先查询 property 表中既含保险（pro_type = '2'）又含基金（pro_type = '3'），通过 property 自身连接查询得到，再通过 client 表嵌套搜索 c_id 在子查询内的元组，组成最后的查询结果。完整代码为：

```
select c_name,c_mail,c_phone
from client
where c_id in (
    select p1.pro_c_id
    from property p1,property p2
    where p1.pro_c_id = p2.pro_c_id AND p1.pro_type = '2' and p2.pro_type = '3');
```

2.2.4 办理了储蓄卡的客户信息

将 client 表和 bank_card 表进行用户 id 的等值连接，找出银行卡类型为储蓄卡的元组，查询出后通过 order by 子句对其排序。完整代码为：

```
select c.c_name,c.c_phone,b.b_number
from client c,bank_card b
where c.c_id = b.b_c_id and b.b_type = '储蓄卡'
order by c.c_id;
```

2.2.5 每份金额在 30000～50000 之间的理财产品

对 finances_product 表单表查询。在 where 子句中，对金额使用 between-and 查找满足 30000~50000 范围内的元组，随后按要求进行升降序排序，降序使用关键字 DESC。完整代码为：

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

2.2.6 商品收益的众数

对 property 表进行分组统计，使用 group by 子句，对收入 pro_income 进行分组，并使用 having 子句，选择组内元组个数大于等于所有分组元组个数的分组，即使用关键字 ALL 和子查询完成上述选择，子查询内统计按 pro_income 分组的组内元组个数。完整代码为：

```
select pro_income,COUNT(*) presence
from property
group by pro_income
having COUNT(*) >= ALL(
    select COUNT(*)
    from property
    group by pro_income);
```

2.2.7 未购买任何理财产品的武汉居民

对 client 查询，首先设定属于武汉的条件，使用 LIKE 搜索以“4201”开头的身份证号，使用“4201%”完成上述条件表达式。其次子查询在 property 表中购买理财产品（pro_type = 1）的用户内没有出现的用户 id，即使用 not exists 谓

词产生条件表达式。具体代码省略。

2.2.8 持有两张信用卡的用户

首先在子查询中，对 bank_card 表查找信用卡的元组，按用户 id 分组后，选择元组数大于等于 2 的组别，投影用户 id。随后在 client 表中查找用户 id 为子查询用户 id 即可。完整代码为：

```
select c_name,c_id_card,c_phone
from client
where c_id in (
    select b_c_id
    from bank_card
    where b_type = '信用卡'
    group by b_c_id
    having count(*) >= 2)
order by c_id;
```

2.2.9 购买了货币型基金的客户信息

此查询为多层嵌套查询，自底向上查询依次为：首先对 fund 表查询出 f_type 为货币型的基金 id（f_id）。其次对 property 表查询元组内类型属性为基金（pro_type = 3），且基金 id 为上述查询的基金 id 的元组，投影用户 id。最后查询 client 表内用户 id 为上述投影 id 的元组，然后根据需求排序与投影。具体代码省略。

2.2.10 投资总收益前三名的客户

对 client 表和 property 进行用户 id 的等值连接，且选取投资状态为“可用”的元组。按用户 id 进行分组统计，使用聚集函数 SUM 统计组内的收益和。按收益和倒序排序，使用 limit 函数获取前 3 个元组，即总收益前 3 的客户。完整代码为：

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

2.2.11 黄姓客户持卡数量

此查询关键点在于有的黄姓客户没有办理银行卡，即在银行卡中没有该用户信息，若使用等值连接，则 client 里的悬浮元组无法统计。因此使用左外连接保存 client 的悬浮元组，便于统计银行卡数目为 0 的黄姓用户。具体的左外连接为 from client left outer join bank_card on(b_c_id = c_id)。具体代码省略。

2.2.12 客户理财、保险与基金投资总额

此次查询涉及多表，因此先生成分表查询合并的派生表，再进行后续分组统计投资总额。首先进行派生表的生成，对 property 表和 finances_product 表进行理财产品 id 的等值连接，以用户 id（pro_c_id）分组，统计理财产品投资总额，sum(pro_quantity*p_amount)；对 property 表和 insurance 表进行保险 id 的等值连接，以用户 id 分组，统计保险投资总额，sum(pro_quantity*i_amount)；对 property 表和 fund 表进行基金 id 的等值连接，以用户 id 分组，统计基金投资总额，sum(pro_quantity*f_amount)，将 3 个查询结果按保留重复元组 UNION ALL 的方式合并为派生表。

然后进行投资总额的统计。将 client 表与上述派生表以用户 id 等值连接，且以用户 id 分组统计，计算该用户 3 类投资产品的投资金额总和。具体代码省略。

2.2.13 客户总资产

采用多表连接的方式统计客户总资产。客户总资产包括用户储蓄卡余额、投资总收益和投资总额，减去信用卡透支金额。将上述的涉及金额都生成相应的派生表，最后将 client 表与上述派生表左外连接后，对于每个元组计算总资产即可。

对于储蓄卡派生表的生成。对 bank_card 单表查询满足卡类型为储蓄卡的元组，以用户 id（b_c_id）为分组统计余额总和；对 bank_card 单表查询满足卡类型为信用卡的元组，以用户 id 为分组统计透支总金额；对 property 单表查询，以用户 id（pro_c_id）为分组统计投资总收益；投资总额的派生表与上一题一致，在此不赘述。

将 client 表与上述产生的派生表进行多表左外连接，对于连接产生的每个元组，进行储蓄卡余额+投资总收益+投资总额-信用卡透支金额的计算，对用户 id 和计算总资产进行排序、投影即可。具体代码省略。

2.2.14 第 N 高问题

采用子查询先搜索第 4 高的保险金额，再进行特定的查询。首先子查询中，对 insurance 表进行单表查询，先按 i_amount 降序排序，采用 distinct 排除重复金额的元组，使用 limit 选择第 4 高的金额数。在外层查询中，对 insurance 表查询 i_amount 数值等于子查询的第 4 高金额数的元组，将查询到满足的所有元组按保险 id 升序排序。完整代码为：

```
select i_id,i_amount
from insurance
where i_amount = (
    select distinct i_amount
    from insurance
    order by i_amount desc
    limit 3,1)
order by i_id;
```

2.2.15 基金收益两种方式排名

本次查询需要使用用户变量进行排名序号的管理。首先生成基金总收益的派生表，对 property 表选择投资类型为基金（pro_type = 3）的元组，按用户 id 分组统计用户的总收益，按总收益降序排序。

对于名次不连续的查询，在 FROM 子句中设置 3 个变量，@rankcount 记录查询元组计数，@currank 记录当前元组排名，@rank_revenue 记录上一元组的总收益。在元组查询时，对 rankcount 进行自增操作，当上一元组的收益 rank_revenue 值与本元组的收益值相等时，将 currank 作为排名属性值，否则将 rankcount 赋给 currank 并作为排名属性值。

对于名次连续的查询，在 FROM 子句中设置 2 个变量，@currank 记录当前元组排名，@rank_revenue 记录上一元组的总收益。在元组查询时，若当前元组的收益等于上一元组的收益 rank_revenue 值时，将 currank 作为排名属性值，否则将 currank 自增 1 并作为排名属性值。

2.2.16 持有完全相同基金组合的客户

首先生成两张相同的客户基金组合派生表，再进行等值连接，进行特定元组的选取。

对于客户基金组合派生表的生成，具体操作为对 property 表查询资金类型为基金（pro_type = 3）的元组，按用户 id 分组，使用 group_concat 函数对组内的基金号合并，生成一个用户基金的集合，集合内按照基金号排序。对于该两个生成表定义为 p1 和 p2。

对上述产生的 p1 和 p2 进行基金组合集合的等值连接，产生基金组合相同的用户对，并且为了减少数据冗余，仅选取前用户 id 小于后用户 id 的元组，按前用户 id 升序排序。具体代码省略。

2.2.17 购买基金的高峰期

关键点在于连续三个交易日的选取。首先对 property 表和 fund 表进行等值连接，选取交易日期在 2022 年 2 月且在交易日的元组组成派生表。然后进行枚举查找 3 个连续的交易日内金额大于 100 万的元组，由于 2 月连续 3 个交易日的组合较少，故该方法具有可行性。对与查找到满足要求的元组，进行 UNION 操作，排除重复元组。具体代码省略。

2.2.18 至少有一张信用卡余额超过 5000 元的客户信用卡总余额

首先生成至少有一张信用卡余额超过 5000 元的用户 id 的派生表，对 bank_card 表查询类型为信用卡且余额大于 5000 的元组，投影该用户 id。

其次将 bank_card 表与该派生表进行用户 id 的等值连接，选取信用卡元组，按用户 id 分组统计信用卡金额的总和，并将其按用户 id 升序排序。具体代码省略。

2.2.19 以日历表格式显示每日基金购买总金额

此查询的关键点在于将列数据筛选变成行数据，即进行行列数据转换。

查询过程如下，首先将 property 表和 fund 表进行基金 id 的等值连接，计算基金购买金额。再将该派生表中选取 2022 年 2 月交易日的元组按购买日期分组统计，由此产生周号、星期号和当日基金金额的派生表，该表为列数据型表。

对此进行行列数据转换，将同周的数据组成一个元组。具体做法则是将该派生表按周号分组统计，在周号相同的一组中，统计星期号为周一的总金额，周二的总金额，...，以此产生每周的元组数据。具体代码省略。

2.3 数据的插入、修改与删除(Insert,Update,Delete)

本任务环节要求使用INSERT、DELETE和UPDATE语句对表进行数据更新。需要掌握完整信息的插入、不完整信息的插入、批量插入、特定元组数据的更新与删除、连接更新等操作。

2.3.1 插入多条完整的客户信息

使用 INSERT 语句，插入多条完整元组数据。完整代码为：

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

2.3.2 插入不完整的客户信息

使用 INSERT 语句，插入一条不完整元组数据。完整代码为：

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

2.3.3 批量插入数据

在 INSERT 语句中，使用 SELECT 批量查询数据并插入。完整代码为：

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

2.3.4 删除没有银行卡的客户信息

使用 DELETE 语句，在 WHERE 子句中，使用 not exists 谓词判断没有银行卡用户的子查询。完整代码为：

```
delete from client
where not exists (
    select *
    from bank_card
    where client.c_id = bank_card.b_c_id);
```

2.3.5 冻结客户资产

使用 UPDATE 语句，WHERE 子句采用嵌套子查询。完整代码为：

```
update property
set pro_status = '冻结'
```

```
where pro_c_id in (
    select c_id
    from client
    where c_phone = '13686431238');
```

2.3.6 连接更新

使用 UPDATE 语句，在 SET 子句中，将更新目标 property 表与 client 表连接，更新用户 id 相同的元组的身份证 id。完整代码为：

```
update property
set pro_id_card = (
    select c_id_card
    from client
    where pro_c_id = c_id);
```

2.4 视图

本任务环节要求使用 CREATE 语句创建视图，且对视图进行查询等的相关操作。要求掌握 CREATE 的视图创建、SELECT 的基于视图的查询等操作。

2.4.1 创建所有保险资产的详细记录视图

使用 CREATE 语句创建所需视图，具体实现方法为 create view <视图名>[(<列名>[,<列名>,...])] as <子查询> [WITH CHECK OPTION];。创建所有保险资产视图将 client 表、property 表和 insurance 表进行多表等值连接，投影所需属性作为视图子查询。完整代码如下：

```
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,property,insurance
where c_id = pro_c_id and pro_type = '2' and pro_pif_id = i_id;
```

2.4.2 基于视图的查询

基于上述创建的视图，进行与查询基本表无异的查询操作。完整代码为：

```
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_id_card
```

```
order by insurance_total_amount desc;
```

2.5 存储过程与事务

本任务环节要求掌握存储过程这一种可编程数据对象的构造与使用，用以增强 SQL 的功能和灵活性，体现模块化编程思想。具体通过三类控制结构进行存储过程的构造，掌握基于流程控制语句的存储过程、基于游标的存储过程、基于事务的存储过程。

2.5.1 使用流程控制语句的存储过程

流程控制语言的编程过程类似过程式的编程思路，提供复合体 begin...end、if控制语句 if then...elseif then... else... end if 和 while 循环语句 while do...end while。首先使用 declare 语句定义斐波那契变量 fibn，数组序列号 n，以及辅助递推变量 a、b。a 初始化为 0，b 初始化为 1。对于初始情况，当输入为 1 或 2 时，直接插入斐波那契数列的对应数值。当输入大于 2 时，进行迭代循环，初始化 n 为 2，置 fibn 为 a+b，并将元组数据(n,fibn)插入结果表。随后置 a 为 b，b 为 fibn，n 自增，重复上述操作，直到 n 为要求输出行数为止。

2.5.2 使用游标的存储过程

该任务难点在于游标循环开启关闭时机，以及主任的排班问题。定义两个游标，其中一个为护士列表游标，另一个为医生（包含主任）列表游标。开启两个游标，在医生列表游标中获取一个医生的姓名和具体类别，因为有主任与医生的区别，在护士列表中获取两行护士姓名数据，分别存储到对应的变量中。随后进行循环迭代，直到所有日期排版为止。

每次迭代的关键在于判断该排班日期是否为周六周日，且排班人员是否为主任。产生 3 种可能情况：（1）当前排班医师为主任且为周末时，将该主任名字记录在暂存变量中并记录在周末出现主任的 flag，接着再取医生列表游标。将满足要求的医生名和两个护士名加入排班表，并再取下一次迭代的值班人员；（2）当前排班星期为周一时且周末出现主任（判断 flag）时，将暂存的主任姓名作为此次值班医生，填表后，只需获取排班护士即可；（3）上述情况之外时，直接插入获取的医生和护士姓名。

值得注意的是，当定义多个游标时，游标遍历结束时不会产生特定游标结束标志，而需要在每次获取游标时判断该处是否遍历结束，若遍历结束，则需关闭

游标后重新开启该游标。

2.5.3 使用事务的存储过程

事务具有原子性，根据该特性可以指定一系列操作，判断这些操作是否满足要求，若满足则 COMMIT 事务提交，否则 ROLLBACK 事务回滚。在进行自定义事务操作前，应将自动事务模式关闭，使用 START TRANSACTION 开启事务。

首先进行相关更新，即完成账户转账的一系列操作。随后进行 if 判断，当在 bank_card 表中没有转款人转出卡号码的储蓄卡时，返回参数置 0 并进行事务回滚，或者当收款人的卡号在 bank_card 表中不存在时，返回参数置 0 并进行事务回滚，若不属于上述情况，则返回参数置 1 并事务提交。

2.6 触发器

本任务环节要求创建触发器进行合法性检查。具体要求在进行 INSERT、DELETE 或 UPDATE 操作时，激活触发器，进行数据完整性的检查，以判断该操作是否合法。要求掌握触发器的构造、old 表与 new 表的使用、出错信息的设置。

2.6.1 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

首先创建触发器结构，需要在 insert 插入数据前判断插入数据的合法性，具体实现为 CREATE TRIGGER before_property_inserted BEFORE INSERT ON property FOR EACH ROW。设置 4 种错误的 if 判断：（1）若添加元组的类型不在 3 类投资类型内，具体判断 new.pro_type 属性，若不合法，则设置错误信息，使用 concat 函数构造指定报错信息 msg，并使用通用异常处理进行报错 signal sqlstate '45000' set message_text = msg;。（2）若添加元组的类型为理财产品，且 finances_product 表中不含添加的产品 id，则进行报错。（3）若添加元组的类型为保险，且 insurance 表中不含添加的保险 id，则进行报错。（4）若添加元组的类型为基金，且 fund 表中不含添加的基金 id，则进行报错。

2.7 用户自定义函数

本任务环节要求创建用户自定义函数并对其进行调用。定义用户自定义函数能够将特定操作模块化，且方便后续的多次调用。要求掌握用户自定义函数的定

义、用户自定义函数的调用。

2.7.1 创建函数并在语句中使用它

使用创建自定义函数的语句进行函数名、参数名以及返回值的定义，具体定义为 create function get_deposit(client_id int) returns numeric(10,2)。构造函数体，可使用 select <单值> into <参数>的方式将查询的单值传给指定参数，此实验中通过查找指定用户的储蓄卡余额进行求和统计，将该单值传入特定参数，并在最后使用 return <参数>;的方式进行函数返回。

对于自定义函数的调用，则可直接将其视为函数调用方式，具体代码为：

```
select c_id_card,c_name,get_deposit(c_id) as total_deposit
from client
where get_deposit(c_id) >= 1000000
order by get_deposit(c_id) desc;
```

2.8 安全性控制

本任务环节要求掌握安全性措施，重点掌握自主存取控制方法。数据库安全性作为防止数据泄露、更改或破坏的特性是主要技术指标之一。要求掌握用户和角色的创建、权限的授予与回收。

2.8.1 用户和权限

首先创建用户，并初始化密码。具体实现为：

```
create user 'tom' identified by '123456';
```

使用 GRANT 语句，对用户授权以及授予转授权限。具体实现为：

```
grant select(c_name,c_phone,c_mail)
on client
to tom
with grant option;
```

使用 REVOKE 语句，回收角色的已授予权限。具体实现为：

```
revoke select
on bank_card
from Cindy;
```

2.8.2 用户、角色与权限

首先创建角色，具体实现为：

```
create role client_manager;
```

使用 GRANT 语句，将权限授予角色，具体实现为：

```
grant select,insert,update
on client
to client_manager;
```

使用 GRANT 语句，将角色权限授予用户，具体实现为：

```
grant client_manager
to tom,jerry;
```

2.9 并发控制与事务的隔离级别

本任务环节中需要掌握事务并发访问数据时产生的不一致性问题。该问题可以通过数据库管理系统的并发控制子系统解决，从而保证事务的隔离性和一致性。具体掌握各隔离级别的隔离特性，不一致性问题的产生与解决，如丢失修改、读脏、不可重复读与幻读，以及共享锁和写锁的添加。

2.9.1 并发控制与事务的隔离级别

MYSQL 对事务的隔离级别由低到高以此为 READ UNCOMITTED、READ COMMITTED、REPEATABLE READ、SERIALIZABLE，依次扩展解决读脏问题、不可重复读问题与幻读问题。具体设置隔离级别代码如下：

```
set session transaction isolation level read uncommitted;
```

2.9.2 读脏

读脏是指当某一事务事务回滚前的修改被其他事务错误读取。为了重现读脏错误，需将事务隔离级别设置为最低级别 READ UNCOMITTED。事务 1 读取读取余票前需设置 sleep，等待事务 2 修改；事务 2 回滚需等待事务 1 读取余额数据，故在事务 2 的最终回滚前设置 sleep。

2.9.3 不可重复读

不可重复读是指某一事务多次读取数据期间，该数据被其他事务修改，导致读取数据不一致的问题。设置最低隔离级别，且设计事务 2 为发生问题的事务。在事务 1 设置 sleep，使得事务 2 在事务 1 修改数据前读取；接着事务 2 设置 sleep，使得事务 1 修改数据；事务 1 修改完数据后，事务 2 继续读取该数据，并进行修改，于是发生不可重复读问题。

2.9.4 幻读

幻读是指某一事务多次读取数据期间，另一事务进行了 insert 或 delete 操作导致表内数据增减，导致多次读取数据的元组数不同。在事务 2 设置好后，进行事务 1 的设计。首先事务 1 先查询超过 300 张余票的航班信息，使用 sleep 等待事务 1 对航班信息进行 insert 添加；随后事务 1 再次进行查询，发现两次查询的元组数不一致，产生幻读问题。

2.9.5 主动加锁保证可重复读

共享锁保证同一事务重复读取某数据时，其他事务将无法修改该数据，从而保证了可重复读。故在事务 1 中，第一次查询余票时，添加共享锁，由此其他事务在加锁期间将无法对该数据进行修改，具体添加共享锁代码为：

```
select tickets from ticket where flight_no = 'MU2455' for share;
```

2.9.6 可串行化

多个事务并发执行是正确的，当且仅当其结果与按某一次序串行地执行这些事务时的结果相同。在事务 2 修改数据时，添加 X 锁，且在读数据时添加 S 锁。为了得到按 T2→T1 的调度执行结果，可在事务 1 执行时先 sleep，让事务 2 先执行，即先加 X 锁。最终保证了事务的可串行化。

2.10 备份+日志：介质故障与数据库恢复

本任务环节要求掌握介质故障与数据库恢复机制。在具体生产环境中，难免遇到数据库破坏、存储介质故障等问题，而利用数据库备份与日志文件恢复数据库可以解决上述问题，从而提高数据安全性和可靠性。具体掌握基于备份的数据库恢复、带日志文件的数据库恢复。

2.10.1 备份与恢复

进行数据库备份需要使用 MYSQL 提供的备份工具 mysqldump，由此备份服务器上的全部数据库或指定数据库。具体命令代码为：

```
mysqldump -h127.0.0.1 -uroot --databases residents > residents_bak.sql
```

随后当对某数据库进行误删（drop）时，可以利用上述的备份文件进行恢复。同样使用 MYSQL 提供的恢复工具 mysql，具体命令代码为：

```
mysql -h127.0.0.1 -uroot < residents_bak.sql
```

2.10.2 备份+日志：介质故障的发生与数据库的恢复

在进行数据库备份时，在 mysqldump 中选择可选参数--flush-logs，从而刷新日志，重新开启新的日志文件。具体命令代码为：

```
mysqldump -h127.0.0.1 -uroot --flush-logs --databases train > train_bak.sql
```

当发生介质故障时，首先通过还原工具 mysql 通过备份数据库 train_bak.sql 恢复数据库。接着使用日志工具，通过故障发生前的日志文件继续恢复数据库。具体命令代码为：

```
mysql -h127.0.0.1 -uroot < train_bak.sql
mysqlbinlog --no-defaults --stop-position=200000 log/binlog.000018 | mysql -uroot
```

2.11 数据库设计与实现

本任务环节要求掌握数据库设计方法与实现过程。数据库设计具体过程概括为在精准需求分析的基础上，设计概率模型，再到逻辑模型，最后进行物理模型的设计，以此完成数据库的建模工作。具体掌握概念模型的设计、逻辑模型的构建、建模工具的使用。

2.11.1 从概念模型到 MySQL 实现

根据建模完成的概念模型，如图 2.1 所示。其中包含实体：用户，旅客，机场，航空公司，民航飞机，航班常规调度表，航班表，机票。为其建表。同时，图中实体间联系为一对多的联系，故联系属性可添加到多端表中，且作为外码管

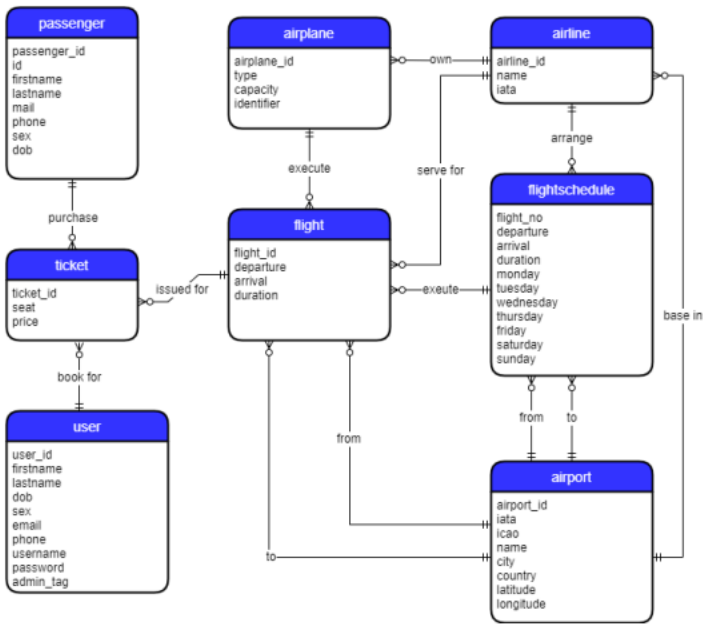

图 2.1 机票订票系统概念模型

理。除此之外，建表时还需要按需求建立完整性约束条件。特别需要注意的，若属性名称为 MYSQL 关键字时，需要使用`<特殊属性>`进行指定，使其作为属性名被使用。

2.11.2 从需求分析到逻辑模型

通过需求分析，可知所需主体为电影、顾客、放映厅、排场、电影。其中顾客与电影存在联系，电影票与排场存在联系，电影与排场存在联系，排场与放映厅存在联系，根据上述分析产生如图 2.2 所示 E-R 图。

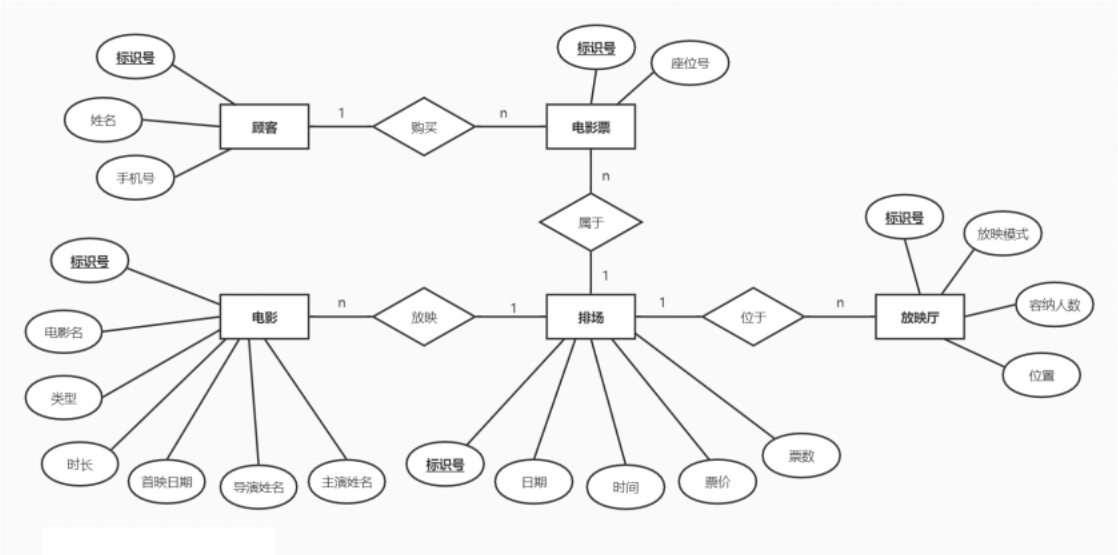

图 2.2 影院管理系统 E-R 图

根据以上 E-R 图，由于关系联系都是一对多联系，故将联系属性添加至多端实体中即可，构建关系模式为：

movie(movie_ID,title,type,runtime,release_date,director,starring,schedule_ID),
主码:(movie_ID);外码:(schedule_ID)
customer(c_ID,name,phone),主码:(c_ID)
hall(hall_ID,mode,capacity,location,schedule_ID),主码:(hall_ID);外码:(schedule_ID)
schedule(schedule_ID,date,time,price,number),主码:(schedule_ID)
ticket(ticket_ID,seat_num,c_ID,schedule_ID),
主码:(ticket_ID);外码:(c_ID,schedule_ID)

2.11.3 建模工具的使用

使用 MySQL Workbench 建模工具，将已建模的模型文件 rbac.mwb，如图 2.3

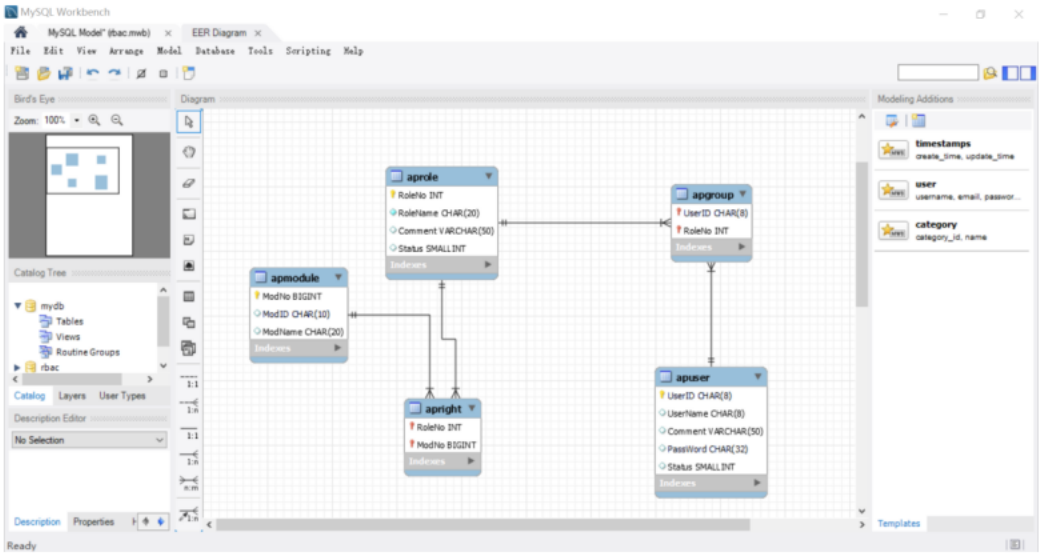

图 2.3 rbac.mwb 建模文件

所示，导出为 SQL 脚本。使用 Forward Engineering 功能，将建模的概念模型导出为 SQL 脚本，如图 2.4 所示。

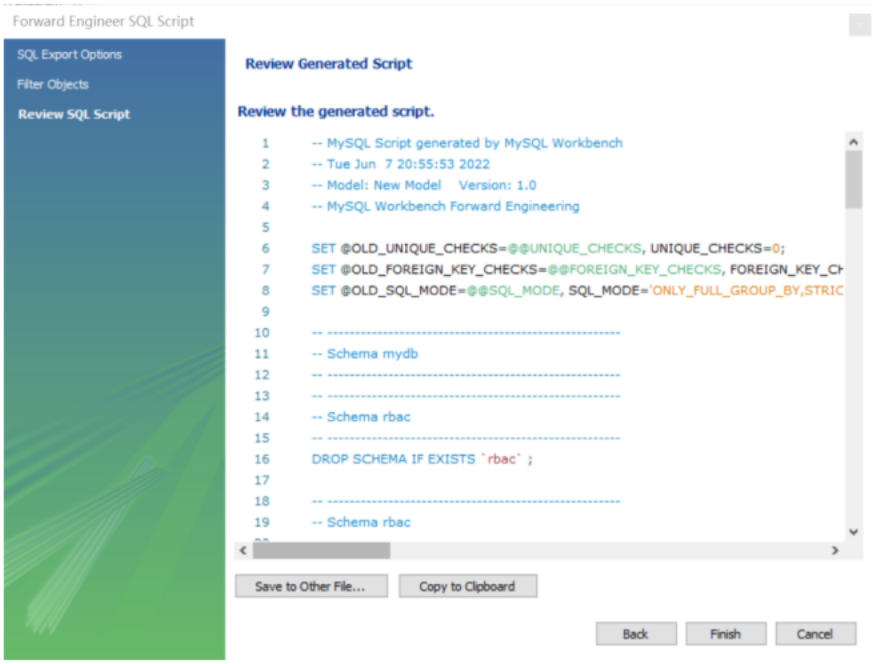

图 2.4 最终导出的 SQL 脚本

2.11.4 制约因素分析与设计

在进行需求分析到概念模型转化时，需要充分考虑到安全性问题、法律性问题等。例如在进行用户信息验证时，存储在数据库中的用户名与对应的密码是否采用密文方式存储。在网络安全作为国家安全之一的当下，保护用户隐私十分重要，且具有法律效力。故在处理用户隐私相关信息时，要考虑加密处理。可采用用户模块加密解密过程，而数据库模块中只允许使用密文存放隐私信息。

2.11.5 工程师责任及其分析

工程师除了进行数据结构设计、系统体系设计、系统模型设计等计算机相关工作要求外，还需要对该系统的应用场景、法律效力、社会安全风险等问题进行充分考虑与分析。不得建立违背法律规定、破坏社会安全的系统与应用，且构建的系统应充分符合使用场景，而不能成为空中楼阁。

2.12 数据库应用开发（JAVA 篇）

本任务环节要求使用 JAVA 高级语言进行数据库应用系统的开发。使用高级语言开发数据库应用系统能够丰富数据库的相关操作，例如与网络的连接等，为数据库提供外层的可靠性、安全性保障。具体掌握 JDBC 的使用，即一种用于执行 SQL 语句的 Java API。

2.12.1 JDBC 体系结构和简单的查询

构建 JDBC 应用包括六个步骤，导入包，注册 JDBC 驱动程序，打开连接，执行 SQL 操作，提取数据，释放资源。具体步骤如下：

（1）首先使用 import java.sql.*;导入 JDBC 包。

（2）其次使用 Class.forName() 方法注册驱动程序，具体实现为 Class.forName("com.mysql.cj.jdbc.Driver");。

（3）接着进行数据库连接，需要创建 Connection 对象，通过 DriverManager.getConnection()方法进行数据库连接，传递 URL 参数、用户名参数、密码参数。

（4）进行 SQL 操作环节。使用 connection.createStatement() 方法创建 Statement 接口，用于发送 SQL 命令并接收数据。编写 SQL 语句放置自定义的 String 对象中。使用 statement.executeQuery()方法进行 SELECT 语句的查询。

（5）提取数据。将（4）中执行返回值赋给一个 ResultSet 对象。接着对该对象进行 next()方法的查询，每次查询一行元组信息，当返回 false 时表示遍历结束。

（6）当查询结束后，需要释放资源。将上述创建的 ResultSet 对象、Statement 对象以及 Connection 对象调用 close()方法关闭。

2.12.2 用户登录

首先获取用户输入的邮箱与密码。进行数据库连接以及 JDBC 对象的创建。查询表中邮箱为用户输入邮箱的元组，将结果存入 ResultSet 对象中。进行 next() 方法查询，若无数据，输出“用户名或密码错误！”；否则进行密码的比较，若与用户输入密码一致，则输出“登录成功”，否则输出“用户名或密码错误！”。处理完毕后，进行各 JDBC 对象的释放。

2.12.3 添加新客户

采用设置自定义方法进行新客户的添加。已知用户的 id、姓名、邮箱、身份证号、电话以及登录密码，将上述信息与已连接数据库的 Connection 对象传入方法中。可使用 PreparedStatement 接口进行 SQL 的传递。将上述信息加入 SQL 语句中，使用该接口的 executeUpdate()方法，该方法主要用于 INSERT、UPDATE 和 DELETE 的操作，并返回操作的元组数，将其作为添加客户方法的返回值。

2.12.4 银行卡销户

创建银行卡销户方法，传入已连接数据库的 Connection 对象、客户 id 以及银行卡号，返回销卡数目。同样使用 PreparedStatement 接口的 executeUpdate() 方法，编写 SQL 语句删除用户 id 为传入 id、银行卡号为传入卡号的元组，将执行后的返回值作为该方法的返回参数。

2.12.5 客户修改密码

创建客户修改密码方法，传入已连接数据库的 Connection 对象、用户邮箱、登录旧密码、登录新密码。使用 PreparedStatement 接口的 executeQuery()方法，查找邮箱为该用户邮箱的元组。若无要求元组，则返回 2 表示用户不存在。当存在时，查看该元组密码是否为用户旧密码，若不是，则返回 3 表示密码不正确；若用户邮箱与密码都匹配，则使用 executeUpdate()进行修改，并返回 1 表示密码修改正确。当出现程序异常时，返回-1。

2.12.6 事务与转账操作

创建转账操作方法，传入已连接数据库的 Connection 对象、转出账号、转入账号和转账金额。由于使用事务隔离控制，首先设置隔离级别为 REPEATABLE READ，具体实现为 connection.setTransactionIsolation(4);。于是先进行转出账号扣账，转入账号为信用卡时还账，为储蓄卡时入账。再进行操作合法性的判断，

若转出储蓄卡账号不存在，则事务回滚，具体实现为 connection.rollback();，且置返回值为 false。若存在转出储蓄卡账号，但转出余额不足，则事务回滚，置返回值为 false。然后查询转入账号是否存在，若不存在则事务回滚并置返回值为 false。不产生以上情况时，表明该转账操作合法，进行事务提交，具体实现为 connection.commit();，并置返回值为 true。

2.12.7 把稀疏表格转为键值对存储

整体设计两个环节分别对应两个方法，主方法 main 中进行稀疏表查询每个元组，对于每个元组，进行学生学号和各科成绩的提取。若某课成绩不存在，则视为成绩为 0，将不为 0 的成绩填入键值对表中，存储格式为（<Sno>，<科目名称>，<成绩>）。

创建存储键值对方法进行第二个环节。传入已连接数据库的 Connection 对象、学生学号 Sno、科目名称和该科成绩。使用 PreparedStatement 接口的 executeUpdate()方法进行该元组的插入。

当完成所有稀疏表中元组的转化后，进行各 JDBC 对象的释放。

3 课程总结

本次数据库系统原理实践整体完成了 13 个实训任务的所有子任务，在此实践过程中，完成了将数据库原理理论知识向实践能力的转换，掌握了以 MySQL 为主的数据库编程能力，掌握了使用 JAVA 高级语言进行实际应用开发的具体思路与操作，深入理解了数据库、数据库管理系统的特点以及与其相关的设计与实现，熟悉了相关建模工具的使用等。

对于各实训环节，具体主要工作如下。完成了数据库、表、完整性约束的创建与修改实训部分的所有子任务，掌握了 MySQL 的 DDL 数据定义语句以及完整性约束的使用；完成了数据查询、修改与删除实训部分的所有子任务，掌握了 DML 数据操纵语言的使用与联系；完成了视图实训部分的所有子任务，掌握了视图这一视图层抽象的构建与使用；完成了存储过程与事务、触发器、用户自定义函数实训部分的所有子任务，掌握了这些 MySQL 数据库对象的构建与使用；完成了安全性控制实训部分的所有子任务，掌握了如何进行自主存取控制提升系统安全性；完成了并发控制与事务的隔离级别实训部分的所有子任务，掌握了数据不一致问题的解决办法；完成了备份加日志的数据库恢复实训部分的所有子任务，掌握了使用日志文件与数据库备份的方法进行数据库恢复的技术；完成了数据库设计与实现实训部分的所有子任务，掌握了数据库系统结构中概念模型和逻辑模型设计与构建，以及对需求的分析；完成了 JAVA 高级语言开发数据库应用实训部分的所有子任务，掌握了 JDBC 体系架构和数据库应用的设计与实现。

通过完成上述实践，让我感受到了数据库在大数据管理与处理方面的强大，特别是进行高级语言的数据库应用开发时，了解并掌握了数据库在现实项目中的具体应用，并且在使用过程中十分便利。对于此次课程实践，建议增加高级语言开发数据库应用的规模，可以通过一个大项目，让同学们具体了解和掌握数据库在具体生产场景下的使用，并通过该大规模项目的开发增加更多的相关方法的介绍，而不是仅仅通过一个实训 13 的 7 个子任务进行简单应用开发介绍。希望能通过加大开发规模，让大家学习更加先进的数据库开发知识与技能，为以后的学习和发展提供帮助。