2022年数据库系统原理实践任务书

# 课程介绍

“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为:<https://www.educoder.net/paths/x7rzkcvp>。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。

在实践进行过程中，大部分时间同学们会面对着这样的界面：

同学们可以尽情在实践环境中大展身手，以提升个人能力，当然也要尽最大努力去完成本次实践的所有关卡。如果学习过程中遇到问题，首先学会自己查资料解决，然后求助老师。

本课程相关资料网站：

MYSQL手册：<https://dev.mysql.com/doc/>

JAVA手册：<https://docs.oracle.com/javase/8/docs/api/index.html>

课程开放资源：<https://gitee.com/kylin8575543/db2022-spring>

实践中遇到平台故障或其他问题，请及时联系：

杨老师：

助教王同学：QQ 2193586366 邮箱 [m202173575@hust.edu.cn](mailto:m202173575@hust.edu.cn)

助教韩同学：QQ 841774960

助教备注：由于头歌OJ平台的使用，本次实践任务大部分都可以完全按照网站上要求进行。尽管如此，本次实践中仍希望同学们在实践第3部分数据查询环节中，翻阅本任务书中提供的背景介绍中的表结构来更容易地完成编程工作。另外遇到任务要求介绍不明确地部分，请同学们首先查看任务书中是否有更清晰的任务要求介绍。如果仍存在题目要求不明的情况，请及时联系老师或助教反应。

# 实践环节

**跳关说明：总共有62关卡，全部完成则获得头哥平台总分118分，不要求所有关卡都完成，最终程序检查满分只计头哥平台中的100分，即有些关卡可以自行选择跳过不做，但是实践环节的2级标题子任务的前4个（2.1至2.4子任务）不能整体跳过（至少要完成其中每个二级子任务中的一个关卡）。**

**2级标题子任务中的第2.12子任务最多只能跳过1个关卡。**

## 数据库、表、完整性约束的创建与修改

### 数据库、表与完整性约束的定义(Create)

**说明：本关卡内测评环境与命令行为同一环境，这也意味着测试环境不会在每次提交时进行重置。因此同学们可以在命令行或自测运行中完成任务要求，然后以一个空的代码文件提交评测，也是可以通过的。**

#### 创建数据库

创建用于2022年北京冬奥会信息系统的数据库:beijing2022。

#### 创建表及表的主码约束

创建数据库TestDb，在TestDb下创建表t\_emp：

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| id | INT | 员工编号，主码 |
| name | VARCHAR(32) | 员工名称 |
| deptId | INT | 所在部门标号 |
| salary | FLOAT | 工资 |

#### 创建外码约束(foreign key)

设有以下两张表：

dept(部门)

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| deptNo | INT | 部门号，主键 |
| deptName | VARCHAR(32) | 部门名称 |

staff(职工)

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| staffNo | INT | 职工号，主键 |
| staffName | VARCHAR(32) | 职工姓名 |
| gender | CHAR(1) | 性别，F-女，M-男 |
| dob | date | 出生日期 |
| salary | numeric(8,2) | 工资 |
| deptNo | INT | 部门号,外键 |

请创建上述两个表，为表定义主键，并给表staff创建外键，这个外键约束的名称为FK\_staff\_deptNo。在创建表之前可能需要先创建数据库：MyDb，并且将两张表创建在MyDb数据库中。不需考虑关于性别的约束。

#### CHECK约束

表products的结构如下：

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| pid | char(10) | 产品户ID,主码 |
| name | varchar(32) | 产品名称 |
| brand | char(10) | 品牌，只能是('A','B')中的某一个 |
| price | int | 价格，必须>0 |

请在数据库MyDb中创建表products，并分别实现对品牌和价格的约束，两个CHECK约束的名称分别为CK\_products\_brand和CK\_products\_price，主码约束不要显示命名。

#### DEFAULT约束

表hr的结构如下：

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| id | char(10) | 工号,主码 |
| name | varchar(32) | 姓名,不允许为空值 |
| mz | char(16) | 民族, 缺省值为“汉族” |

请在数据库MyDb中创建表hr，并实现name列的NOT NULL约束和mz列的Default约束。

#### UNIQUE约束

表s的结构如下：

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| sno | char(10) | 学号,主码 |
| name | varchar(32) | 姓名,不允许为空值 |
| ID | char(18) | 身份证号, 不允许有两个相同的身份证号 |

请在数据库MyDb中创建表s，并实现相关约束。

### 表结构与完整性约束的修改(ALTER)

**注意：从本关卡开始，评测环境与命令行环境成为两个独立的环境，评测环境在每次评测时会进行重置。因此，同学们需要在每个关卡中在代码文件中完整给出代码。在代码文件中，最好不要使用use指定数据库，以免评测出现问题。**

#### 修改表名

数据库TestDb1中有表your\_table，请根据提示，在右侧代码文件编辑窗中添加恰当的语句，将表名your\_table更改为my\_table。

#### 添加与删除字段

假设数据库MyDb中有表order(订单)和orderDetail(订单明细) 等表，两表的结构分别如下：

order表

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| orderNo | char(12) | 订单号，主码 |
| orderDate | date | 订购日期 |
| customerNo | char(12) | 客户编号，外码，与customer.customerNo对应 |
| employeeNo | char(12) | 雇员工号，外码，与employee.employeeNo对应 |

orderDetail表

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| orderNo | char(12) | 订单号，主属性，外码，与order.orderNo对应 |
| productNo | char(12) | 产品编号，主属性，外码，与product.productNo对应 |
| quantityOrdered | int | 订购数量 |
| orderDate | date | 订购日期 |

编程的任务是对orderDetail表进行修改：

1 orderDetail表的orderDate列明显多余，因为同一订单中的每一笔交易都发生在同一天，这个日期在订单主体表order中已有记录，请删除列orderDate。

2 产品的单价是订单明细需要记录的内容，请在orderDetail中添加列unitPrice以记录产品的单价:

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| unitPrice | numeric(10,2) | 产品的成交单价 |

#### 修改字段

数据库MyDb中有表addressBook(通信录)，结构如下：

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| serialNo | int | 自动编号，主码 |
| name | char(32) | 姓名 |
| company | char(32) | 工作单位 |
| position | char(10) | 职位 |
| workPhone | char(16) | 办公电话 |
| mobile | char(11) | 手机 |
| QQ | int | QQ号 |
| weixin | char(12) | 微信号 |

本关编程任务是对表addressBook作以下修改：

1 将QQ号的数据类型改为char(12);

2 将列名weixin改为wechat。

#### 添加、删除与修改约束

数据库MyDb中有以下两表：

Dept(部门)

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| deptNo | INT | 部门号，主键 |
| deptName | VARCHAR(32) | 部门名称，不同部门不允许重名 |
| tel | char(11) | 部门电话 |
| mgrStaffNo | int | 部门经理的工号，外码 |

Staff(职工)

| **字段名称** | **数据类型** | **备注** |
| --- | --- | --- |
| staffNo | INT | 工号，主键 |
| staffName | VARCHAR(32) | 职工姓名 |
| gender | CHAR(1) | 性别，取值范围：F-女，M-男 |
| dob | date | 出生日期 |
| Salary | numeric(8,2) | 工资 |
| dept | INT | 部门号,外键 |

本关编程任务是：

1 为表Staff添加主码;

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;

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;

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;

5 为表Dept添加unique约束：deptName不允许重复。约束名为UN\_Dept\_deptName.

## 数据查询(Select)

#### 背景介绍

本实训采用的是某银行的一个金融场景应用的模拟数据库，数据库中表，表结构以及所有字段的说明如下：

表1 client(客户表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| c\_id | INTEGER | PRIMARY KEY | 客户编号 |
| c\_name | VARCHAR(100) | NOT NULL | 客户名称 |
| c\_mail | CHAR(30) | UNIQUE | 客户邮箱 |
| c\_id\_card | CHAR(20) | UNIQUE NOT NULL | 客户身份证 |
| c\_phone | CHAR(20) | UNIQUE NOT NULL | 客户手机号 |
| c\_password | CHAR(20) | NOT NULL | 客户登录密码 |

表2 bank\_card(银行卡)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| b\_number | CHAR(30) | PRIMARY KEY | 银行卡号 |
| b\_type | CHAR(20) | 无 | 银行卡类型(储蓄卡/信用卡) |
| b\_c\_id | INTEGER | NOT NULL FOREIGN KEY | 所属客户编号,引用自client表的c\_id字段。 |
| b\_balance | NUMERIC(10,2) | NOT NULL | 余额,信用卡余额系指已透支的金额 |

说明：银行卡类型只有”储蓄卡”或”信用卡”两种取值。对于b\_balance列，如果b\_type, b\_balance取值为(”储蓄卡”,10000)，表示这张储蓄卡内有10000的储蓄余额；而b\_type, b\_balance取值为(”信用卡”,10000)，表示这张信用卡已经透支10000元。

表3 finances\_product(理财产品表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| p\_name | VARCHAR(100) | NOT NULL | 产品名称 |
| p\_id | INTEGER | PRIMARY KEY | 产品编号 |
| p\_description | VARCHAR(4000) | 无 | 产品描述 |
| p\_amount | INTEGER | 无 | 购买金额 |
| p\_year | INTEGER | 无 | 理财年限 |

表4 insurance(保险表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| i\_name | VARCHAR(100) | NOT NULL | 保险名称 |
| i\_id | INTEGER | PRIMARY KEY | 保险编号 |
| i\_amount | INTEGER | 无 | 保险金额 |
| i\_person | CHAR(20) | 无 | 适用人群 |
| i\_year | INTEGER | 无 | 保险年限 |
| i\_project | VARCHAR(200) | 无 | 保障项目 |

表5 fund(基金表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| f\_name | VARCHAR(100) | NOT NULL | 基金名称 |
| f\_id | INTEGER | PRIMARY KEY | 基金编号 |
| f\_type | CHAR(20) | 无 | 基金类型 |
| f\_amount | INTEGER | 无 | 基金金额 |
| risk\_level | CHAR(20) | NOT NULL | 风险等级 |
| f\_manager | INTEGER | NOT NULL | 基金管理者 |

说明：以上3张表中的金额都指每购入一份所要花费的金额。

表6 property(资产表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| pro\_id | INTEGER | PRIMARY KEY | 资产编号 |
| pro\_c\_id | INTEGER | NOT NULL | 客户编号  FOREIGN KEY |
| pro\_pif\_id | INTEGER | NOT NULL | 业务约束 |
| pro\_type | INTEGER | NOT NULL | 商品类型:1表示理财产品;2表示保险;3表示基金 |
| pro\_status | CHAR(20) | 无 | 商品状态 |
| pro\_quantity | INTEGER | 无 | 商品数量 |
| pro\_income | INTEGER | 无 | 商品收益 |
| pro\_purchase\_time | DATE | 无 | 购买时间 |

说明：1.商品状态只有”可用”或”冻结”两种取值。2.商品收益指的是本条资产记录所记录商品的总收益，例如f\_id, f\_amount为(1031, 10000), pro\_pif\_id, pro\_type, pro\_quantity, pro\_income为(1031,3,10,27000)表示这条资产记录购入10份1037号基金，总花费10\*10000=100000元，总收益为27000元。

#### 查询客户主要信息

查询所有客户的名称、手机号和邮箱信息。查询结果按照客户编号排序。

输出：c\_name,c\_phone,c\_mail

#### 邮箱为null的客户

查询客户表(client)中没有填写邮箱信息(邮箱字段值为null)的客户的编号、名称、身份证号、手机号。

输出：c\_id,c\_name,c\_id\_card,c\_phone

#### 既买了保险又买了基金的客户

查询既买了保险又买了基金的客户的名称、邮箱和电话,结果依客户编号排序。

输出：c\_name,c\_mail,c\_phone

注意：MYSQL中不支持intersect

#### 办理了储蓄卡的客户信息

查询办理了储蓄卡的客户名称、手机号和银行卡号。注意一个客户在本行可能不止一张储蓄卡，应全列出。查询结果结果依客户编号排序。

输出：c\_name,c\_phone,b\_number

#### 每份金额在30000～50000之间的理财产品

查询理财产品中每份金额在30000～50000之间的理财产品的编号,每份金额，理财年限，并按金额升序排序，金额相同的按照理财年限降序排序。

输出：p\_id,p\_amount,p\_year

#### 商品收益的众数

众数是一组数据中出现次数最多的数值，有时众数在一组数中会有好几个。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，出现的次数命名为presence。

输出：pro\_income, presence

#### 未购买任何理财产品的武汉居民

已知身份证前6位表示居民地区，其中4201开头表示湖北省武汉市。查询身份证隶属武汉市没有买过任何理财产品的客户的名称、电话号、邮箱。依客户编号排序。

输出：c\_name,c\_phone,c\_mail

#### 持有两张信用卡的用户

查询持有两张(含）以上信用卡的用户的名称、身份证号、手机号。查询结果依客户编号排序。

输出：c\_name,c\_id\_card,c\_phone

#### 购买了货币型基金的客户信息

查询购买了货币型(f\_type='货币型')基金的用户的名称、电话号、邮箱。依客户编号排序。

输出：c\_name,c\_phone,c\_mail

#### 投资总收益前三名的客户

查询当前总的可用资产收益(被冻结的资产除外)前三名的客户的名称、身份证号及其总收益，按收益降序输出，总收益命名为total\_income。不需要考虑并列排名情形。

输出：c\_id,c\_name,c\_id\_card,total\_income

注意：1. 本题目不需要考虑并列排名情形，这意味着前5名为10000，9900，9900，8000，8000时，也只需要给出10000，9900，9900即可。2. 背景介绍中已给出，pro\_status有”可用”或”冻结”两种取值，pro\_status=”冻结”表示这条资产记录处于冻结状态。

#### 黄姓客户持卡数量

给出黄姓用户的编号、名称、办理的银行卡的数量(没有办卡的卡数量计为0)。银行卡数量命名为number\_of\_cards。 按办理银行卡数量降序输出,持卡数量相同的,依客户编号排序。

输出：c\_id,c\_name, number\_of\_cards

#### 客户理财、保险与基金投资总额

综合客户表(client)、资产表(property)、理财产品表(finances\_product)、保险表(insurance)和基金表(fund)，列出客户的名称、身份证号以及投资总金额（即投资本金，每笔投资金额=商品数量\*该产品每份金额)，投资总金额命名为total\_amount。查询结果按总金额降序排序。

输出：c\_name, c\_id\_card, total\_amount

注意：1.投资金额按类型需查询不同的表，投资总金额是客户购买的各类(理财,保险,基金)资产投资金额的总和。2. 不考虑资产是否可用，可用的和冻结的都考虑在内。3.需要列出所有客户的投资总金额，如果他/她没有购买过任何资产，那么他/她的投资总金额为0。

#### 客户总资产

综合客户表(client)、资产表(property)、理财产品表(finances\_product)、保险表(insurance)、基金表(fund)，列出所有客户的编号、名称和总资产，总资产命名为total\_property。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(信用卡余额即为透支金额)。客户总资产包括被冻结的资产。

输出：c\_id,c\_name,total\_property

注意：与12)相同，客户总资产为0也需要列出。

#### 第N高问题

查询每份保险金额第4高保险产品的编号和保险金额。在数字序列8000,8000,7000,7000,6000中，两个8000均为第1高，两个7000均为第2高,6000为第3高。

输出：i\_id,i\_amount

#### 基金收益两种方式排名

查询资产表中客户编号，客户基金投资总收益,基金投资总收益的排名(从高到低排名)。总收益相同时名次亦相同(即并列名次)。总收益命名为total\_revenue, 名次命名为rank。第一条SQL语句实现全局名次不连续的排名，第二条SQL语句实现全局名次连续的排名。不管哪种方式排名，收益相同时,客户编号小的排在前。

输出：pro\_c\_id,total\_revenue,rank

注意：1.排名说明详见平台相关知识部分。2.实践前面介绍过，当数据库/表/列与SQL保留字同名时，需要用`（大键盘数字键1前一个按键，英文输入模式下）框起加以区分，例如sql关键字select，如果需要建立一个名为select的表，语句应该为 create table `select`;

#### 持有完全相同基金组合的客户

查询持有相同基金组合的客户对，如编号为A的客户持有的基金，编号为B的客户也持有，反过来，编号为B的客户持有的基金，编号为A的客户也持有，则(A,B)即为持有相同基金组合的二元组，请列出这样的客户对。为避免过多的重复，如果(1,2)为满足条件的元组，则不必显示(2,1)，即只显示 编号小者在前的那一对，这一组客户编号分别命名为c\_id1,c\_id2。

输出：c\_id1, c\_id2

#### 购买基金的高峰期

查询2022年2月购买基金的高峰期。至少连续三个交易日，所有投资者购买基金的总金额每天超过100万(含)，则称这段连续交易日为投资者购买基金的高峰期。只有交易日才能购买基金,但不能保证每个交易日都有投资者购买基金。2022年春节假期之后的第1个交易日为2月7日,周六和周日是非交易日，其余均为交易日。请列出高峰时段的日期和当日基金的总购买金额，按日期顺序排序。总购买金额命名为total\_amount。

输出：pro\_purchase\_time,total\_amount

说明：1.高峰期内每一天的金额都超过100万；2.注意高峰期指连续的交易日，而不是连续的日子，这意味着周四、周五、周一有可能构成高峰日。举例：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 周一 | 周二 | 周三 | 周四 | 周五 | 周六 | 周日 |
| 800000 | 1100000 | 900000 | 1080000 | 1008000 | 闭市 | 闭市 |
| 1200000 | 1300000 | 990000 | 1000000 | 1900000 | 闭市 | 闭市 |

#### 至少有一张信用卡余额超过5000元的客户信用卡总余额

查询至少有一张信用卡余额超过5000元的客户编号，以及该客户持有的信用卡总余额，总余额命名为credit\_card\_amount。查询结果依客户编号排序。

输出：b\_c\_id, credit\_card\_amount

注意：这里实际上说的是信用卡的已透支金额。

#### 以日历表格式显示每日基金购买总金额

以日历表格式列出2022年2月每周每个交易日基金购买总金额，输出格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| week\_of\_trading | Monday | Tuesday | Wednesday | Thursday | Friday |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

列表中第1列为周次，2022年2月7日(星期一)为当月的第1个交易日，这一周记为第1周次。注意显示结果并不需要画表格线，只需按这个格式输出结果即可。

输出：如上

注意：2月7日星期一做为日历表的开始，2月28日星期一是2月所有交易日的结束。之后的日期既不应该用3月日期填补，也不应该为这些日期写0，因为它们是不存在的日期，在这些位置输出NULL(思考SQL查询输出中什么情况下会出现空值)

## 数据的插入、修改与删除(Insert,Update,Delete)

#### 插入多条完整的客户信息

向客户表插入以下3条数据:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **c\_id** | **c\_name** | **c\_mail** | **c\_id\_card** | **c\_phone** | **c\_password** |
| 1 | 林惠雯 | [960323053@qq.com](mailto:960323053@qq.com) | 411014196712130323 | 15609032348 | Mop5UPkl |
| 2 | 吴婉瑜 | [1613230826@gmail.com](mailto:1613230826@gmail.com) | 420152196802131323 | 17605132307 | QUTPhxgVNlXtMxN |
| 3 | 蔡贞仪 | [252323341@foxmail.com](mailto:252323341@foxmail.com) | 160347199005222323 | 17763232321 | Bwe3gyhEErJ7 |

用一条insert语句，或用三条insert语句完成，都可以。

#### 插入不完整的客户信息

已知33号客户部分信息如下:

c\_id(编号):33

c\_name(名称):蔡依婷

c\_phone(电话):18820762130

c\_id\_card(身份证号):350972199204227621

c\_password(密码):MKwEuc1sc6

请用一条SQL语句将这名客户的信息插入到客户表(client)。

#### 批量插入数据

已知表new\_client保存了一批新客户信息，该表与client表结构完全相同。请用一条SQL语句将new\_client表的全部客户信息插入到客户表(client)。

#### 删除没有银行卡的客户信息

请用一条SQL语句删除client表中没有银行卡的客户信息。

注意：MySQL的delete语句中from关键词不能省略。

#### 冻结客户资产

请用一条update语句将手机号码为“13686431238”这位客户的投资资产(理财、保险与基金)的状态置为”冻结”。

#### 连接更新

在金融应用场景数据库中，已在表property(资产表)中添加了客户身份证列，列名为pro\_id\_card，类型为char(18)，该列目前全部留空(null)。

请用一条update语句，根据client表中提供的身份证号(c\_id\_card)，填写property表中对应的身份证号信息(pro\_id\_card)。

## 视图

#### 创建所有保险资产的详细记录视图

创建包含所有保险资产记录的详细信息的视图v\_insurance\_detail，包括购买客户的名称、客户的身份证号、保险名称、保障项目、商品状态、商品数量、保险金额、保险年限、商品收益和购买时间。

视图名：v\_insurance\_detail

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

#### 基于视图的查询

基于上一关创建的视图v\_insurance\_detail进行分组统计查询，列出每位客户的姓名，身份证号，保险投资总额(insurance\_total\_amount)和保险投资总收益(insurance\_total\_revenue),结果依保险投资总额降序排列。

输出：c\_name,c\_id\_card,insurance\_total\_amount,insurance\_total\_revenue

注意：对于视图中每条资产记录，i\_amount记录的是每一份保险的购买金额，而pro\_income记录的是总的收益，注意区分。

## 存储过程与事务

#### 使用流程控制语句的存储过程

数据库中有表fibonacci，用来储存斐波拉契数列的前n项：

| **列名** | **类型** | **说明** |
| --- | --- | --- |
| n | int | 斐波拉契数列的第n项,主码 |
| fibn | bigint | 第n项的值 |

创建存储过程sp\_fibonacci(in m int)，向表fibonacci插入斐波拉契数列的前m项，及其对应的斐波拉契数。fibonacci表初始值为一张空表。请保证存储过程可以多次运行而不出错。

注意：测试使用m范围为1<=m<=92

#### 使用游标的存储过程

医院的某科室有科室主任1名(亦为医生)，医生若干(至少2名，不含主任)，护士若干(至少4人)，现在需要编写一存储过程，自动安排某个连续期间的大夜班(即每天00:00-8:00时间段)的值班表，排班规则为：

1. 每个夜班安排1名医生，2名护士；
2. 值班顺序依工号顺序循环轮流安排(即排至最后1名后再从第1名接着排)；
3. 科室主任参与轮值夜班，但不安排周末(星期六和星期天)的夜班，当周末轮至科主任时，主任的夜班调至周一，由排在主任后面的医生依次递补值周末的夜班。

存储过程的名字为sp\_night\_shift\_arrange,它带两个输入参数：start\_date, end\_date，分别指排班的起始时间和结束时间。排班结果直接写入表night\_shift\_schedule，其结构如下：

表night\_shift\_schedule(夜班值班安排表)

| **列** | **类型** | **说明** |
| --- | --- | --- |
| n\_date | date | 日期, primary key |
| n\_doctor\_name | char(30) | 医生姓名 |
| n\_nurse1\_name | char(30) | 护士1姓名 |
| n\_nurse2\_name | char(30) | 护士2姓名 |

假定该科室没有同名的医生和同名的护士。

科室参与值班的医护人员存储在表employee中，其结构为：

表employee(医护人员表)

| **列** | **类型** | **说明** |
| --- | --- | --- |
| e\_id | int | 编号, primary key |
| e\_name | char(30) | 姓名 |
| e\_type | int | 类别：1-主任,医生;2-医生;3-护士 |

不用考虑其它信息(比如科室之类的)，在生产环境中，只需在where短语中施加条件限制即可明确选出所需科室的医护人员。这里，且把表中全部人员视为该科室人员。

#### 使用事务的存储过程

在金融应用场景数据库中，编程实现一个转账操作的存储过程sp\_transfer，实现从一个帐户向另一个帐户转账。该过程有5个输入参数：

applicant\_id 付款人编号

source\_card\_id 付款卡号

receiver\_card\_id 收款人编号

dest\_card\_id 收款卡号

amount 转账金额

还有1个整型输出参数：

return\_code 1：正常转账；0:转账不成功

转账操作涉及对表bank\_card的操作(在生产环境中，至少还要记录转账操作本身相关的信息至转账表，在实验环境中没有设计这样的表，从略；另外，生产环境中，当银行卡被冻结，或被卡主挂失后，都不能进行转账，在实验环境中，没有设计相应的字段 ，故也从略)。

注意事项：

仅当转款人是转出卡的持有人时，才可转出；

仅当收款人是收款卡的持有人时，才可转入；

储蓄卡之间可以相互转账；

允许储蓄卡向信用卡转账，称为信用卡还款(允许替它人还款)，还款可以超过信用卡余额，此时，信用卡余额为负数；

信用卡不能向储蓄卡转账；

转账金额不能超过储蓄卡余额；

## 触发器

#### 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

补充代码，实现本任务所要求的完整性业务规则。当插入的数据不符合要求时，拒绝数据的插入，并反馈出错信息：

(1) pro\_type数据不合法时，显示:

type x is illegal!

这里，x系指试图插入的pro\_type值。

(2) pro\_type = 1,但pro\_pif\_id不是finances\_product表中的某个主码值，显示:

finances product #x not found!

这里,x系指试图插入的pro\_pif\_id的值。

(3) pro\_type = 2,但pro\_pif\_id不是insurance表中的某个主码值，显示:

insurance #x not found!

这里,x系指试图插入的pro\_pif\_id的值。

(3) pro\_type = 3,但pro\_pif\_id不是fund表中的某个主码值，显示:

fund #x not found!

这里,x系指试图插入的pro\_pif\_id的值。

注意：出错信息中的#号是需要输出的部分，输出举例为：(该示例可以通过测试集查看)：

ERROR 1644 (45000) at line 4: type 21 is illegal!

ERROR 1644 (45000) at line 4: fund #40 not found!

## 用户自定义函数

#### 创建函数并在语句中使用它

补充代码，完成以下编程任务：

(1) 用create function语句创建符合以下要求的函数：

依据客户编号计算其所有储蓄卡余额的总和。

函数名为：get\_deposit

(2) 利用创建的函数，仅用一条SQL语句查询存款总额在100万(含)以上的客户身份证号，姓名和存款总额(total\_deposit)，结果依存储总额从高到低排序。

输出：c\_id\_card,c\_name, total\_deposit

## 安全性控制

#### 用户和权限

填写语句，完成以下创建用户和授权操作：

(1) 创建用户tom和jerry，初始密码均为'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；

(3) 授予用户jerry修改银行卡余额的权限；

(4) 收回用户Cindy查询银行卡信息的权限。

#### 用户、角色与权限

填写语句，完成以下创建用户和授权操作：

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

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

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

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

(5) 将client\_manager的权限授予用户tom和jerry；

(6) 将fund\_manager权限授予用户Cindy.

## 并发控制与事务的隔离级别

#### 背景介绍

有表ticket记录了航班余票数，其结构如下表所示：

| **列** | **类型** | **说明** |
| --- | --- | --- |
| flight\_no | char(6) | primary key |
| tickets | int | 余票数 |

有两个涉及该表的并发事务t1和t2，分别定义在t1.sql和t2.sql代码文件中。平台会让两个事务并发执行，请同学们通过修改代码文件来达到题目预期的并发执行效果。

#### 并发控制与事务的隔离级别

根据提示补充适当的代码，将事务的隔离级别设置为read committed; 并以rollback语句结束事务。

#### 读脏

请在两个代码文件适当的地方补充代码，构造“读脏”现象。t1是读脏的那个事务,而t2是那个修改数据后又撤销的事务。

提示：最简单的实现手段是通过延时来人为指定两个并行事务内部的操作顺序。延时如何实现？如果实在没有方法解决可以先预览本大关的第5关的部分代码。

#### 不可重复读

请在两个代码文件适当的地方补充代码，构造“读脏”现象。构造“不可重复读”现象。t2是发生不可重复读的那个事务,t1在t2的两次连续读之间修改了数据。

#### 幻读

本题中t2.sql已提前写好。在代码文件t1.sql适当的地方补充代码，要求如下:

两次查询余票超过300张的航班信息(第2次查询已提前写好)；

在第1次查询之后，事务t2插入了一条航班信息并提交；

第2次查询的记录数增多,发生“幻读”。

不得修改t1的事务隔离级别(保持默认的repeatable read)。

#### 主动加锁保证可重复读

请在两个代码文件适当的地方补充代码，实现：

两个事务的隔离级别都设置成read uncommitted;

事务t1连续查询两次航班MU2455的余票；

事务t2在t1的两次查询之间试图进行一次出票操作：将航班MU2455的余票减去1张。

保证事务t1的两次读结果是一样的(可重复读)

事务t1结束后，在等待t2正常提交(commit)后，再查询一次全部航班的余票,MU2455的余票应该减少1张。

#### 可串行化

在本关中事务t1两次查询航班MU2455的余票，事务t2修改航班MU2455的余票(减1)。请对两个代码文件进行修改，使得两个事务并发执行的结果与t2→t1串行执行的结果相同。

## 备份+日志：介质故障与数据库恢复

#### 备份与恢复

设有居民人口登记数据库residents,请为该数据库做一次静态的(个人独享服务器)海量逻辑备份，备份文件命名为residents\_bak.sql。 然后再用该逻辑备份文件恢复数据库。

注意：请仔细阅读平台上给出的注意部分，以免命令行操作中出现问题。

#### 备份+日志：介质故障的发生与数据库的恢复

摸拟介质故障的发生与数据库的恢复。依时间顺序发生下列事件：

时间点1：数据库train有业务数据产生；

时间点2：对数据库train作一次海量备份，同时新开日志；

时间点3：数据库train又有业务数据产生；

时间点4：故障发生；

时间点5：恢复数据库，保证两次发生的业务数据都不丢失。

补充完成时间点2和5的脚本文件：

对数据库train作一次海量逻辑备份同时开新日志：test2\_1.sh

利用备份文件和日志文件恢复数据库: test2\_2.sh

## 数据库设计与实现

#### 从概念模型到MySQL实现

任务要求详见平台编程要求，此处给出E-R图方便翻阅。



#### 从需求分析到逻辑模型

设计一个影院管理系统。影院对当前的放映厅和电影进行排片，顾客到来后，可以购买任一排场的电影票，进入对应放映厅观看。系统中有以下实体集：

电影(movie)：属性有标识号(movie\_ID)、电影名(title)、类型(type)、时长(runtime)、首映日期(release\_date)、导演姓名(director)、主演姓名(starring)。

顾客(customer)：属性有标识号(c\_ID)、姓名(name)、手机号(phone)。

放映厅(hall)：属性有标识号(hall\_ID)、放映模式(mode)、容纳人数(capacity)、位置(location)。

排场(schedule)：属性有标识号(schedule\_ID)、日期(date)、时间(time)、票价(price)、票数(number)。

电影票(ticket)：属性有标识号(ticket\_ID)、座位号(seat\_num)。

实体间的关系描述如下：

①. 顾客和电影票有一对多的购买关系。每位顾客可以买多张电影票，每张电影票被一位顾客购买。

②. 电影票和排场有多对一的属于关系。一张电影票只属于一个排场，一个排场有多张电影票。

③. 排场和电影有一对多的放映关系。每个排场放一部电影，每部电影可以在多个排场放映。

④. 排场和放映厅有一对多的位于关系。每个排场位于一个放映厅，每个放映厅可以安排多个排场。

请按照上述要求完成：

1.画出E-R图。

2.给出对应的关系模式。

在平台文本编辑器内给出E-R图url和关系模式。

强烈建议同学们在提交前检查一次自己提交的url是否可用，有些图片链接点击可用是由于在本地上有正确的cookie。同学们可以将链接发送到手机上，在手机上尝试是否可以查看图片来确认。

互联网上图片都以图链形式传送，在web中右键点击看到的图片，菜单中大部分时间都会有复制图片链接的选项，在本关中需要同学们来提供一个这样的url链接，在网络上定位到E-R图图片。

以下给出4种创建图链的方式，当然还有更多其他方式可以达到目的，同学们可以自己选择其一。

1. git操作示意

①登录gitee或注册一个gitee账号。

②如果账号下还没有仓库，按下边操作创建一个可用的仓库，如果已有跳转到3.



或在右上角



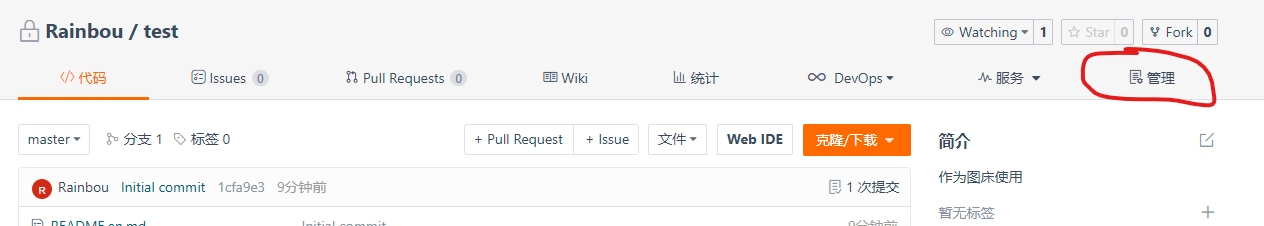
在创建仓库时可以不指定初始化仓库，勾选设置模板中的Readme文件即可。



如果没有勾选设置模板，面对一个原始的仓库，也可以点击这里来达到同样目的。



点击管理来把仓库设置为开源（只需要在第一个设置页下拉即可）。

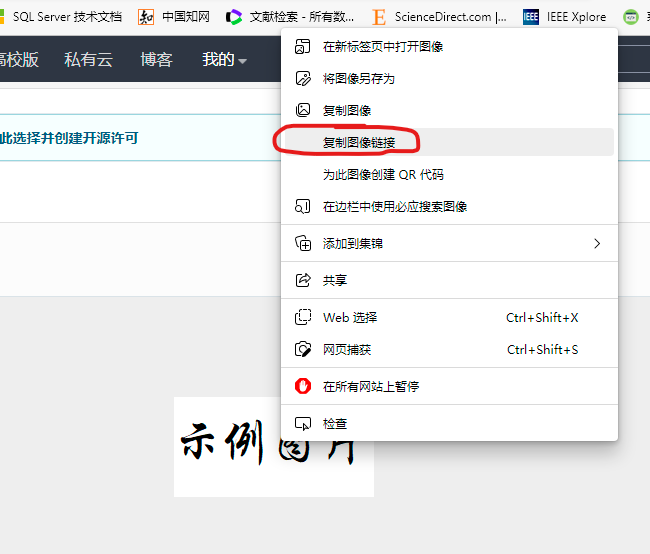




③回到仓库首页，来上传E-R图图形文件。



④点击图形文件查看，并在图片上右键选择复制链接，接下来就可以把url粘贴到平台上进行提交。



例如：<https://gitee.com/rainbou_1/test/raw/master/2.png>

1. 路过图床

①<https://imgtu.com/> 点击链接进入图床站。

②上传图片。

注：支持上传jpg、png、gif格式图片。

③下划到图片链接区域，点击复制url链接，在平台提交。



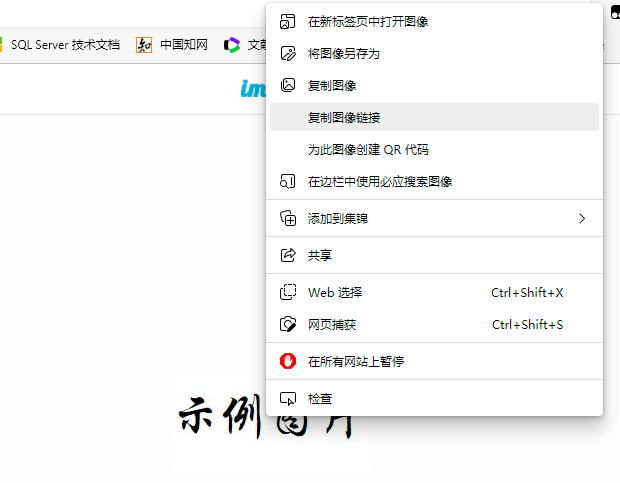
例如：<https://s1.ax1x.com/2022/03/29/qsLNDg.jpg>

1. imgbb图床

①<https://imgbb.com/> 点击链接进入图床站.

②上传图片，上传成功后点击图片进入查看大图界面。

③图片上右键选择复制链接，接下来就可以把url粘贴到平台上进行提交。

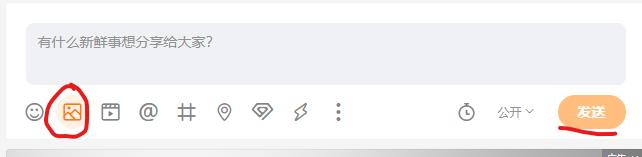


例如：<https://i.ibb.co/KjvsVC2/2.jpg>

1. 新浪微博

①登录微博 <https://weibo.com/>

②创建一条图片微博。



③点击发出的微博查看大图，并右键复制链接，粘贴到平台上提交。



例如：<https://wx4.sinaimg.cn/mw2000/0077zLsRly1h0qiguzgwqj305k02s3ye.jpg>

#### 建模工具的使用

下载资料包中的rbac.mwb，利用MySQL Workbench建模模块中的Forward engineering功能，自动导出SQL脚本，把脚本粘贴到代码文件中。

## 数据库应用开发(JAVA篇)（最多跳过一题）

#### JDBC体系结构和简单的查询

正确使用JDBC，查询金融应用场景数据库finance的client表(客户表)中邮箱不为空的客户信息，列出客户姓名，邮箱和电话.

输出格式要求：

姓名 邮箱 电话

夏雅惠 [57433144004@qq.com](mailto:57433144004@qq.com) 18962433158

钟庭玮 [59434199077@163.com](mailto:59434199077@163.com) 18110434192

其中，标题以及字段值之间用制表符隔开。第1列和第2列间用一个制表符，第2列和第3列间，标题用4个制表符，字段值用两个制表符隔开。

#### 用户登录

编程体验客户登录功能.程序先后提示客户输用户名和密码:

请输入用户名：

请输入密码：

客户的邮箱(c\_mail)充当用户名,而不是编号(c\_id).通常邮箱更容易记住.

根据客户的输入,输出以下两类信息之一:

登录成功。

用户名或密码错误！

#### 添加新客户

补充代码，实现向client表插入客户信息的方法，返回插入的行数。

其他要求详见平台。

#### 银行卡销户

补充代码，实现向银行卡销号的方法，只要客户编号和银行卡号匹配，即从bank\_card表中删除该银行卡，返回被删除的行数。

#### 客户修改密码

补充代码，实现修改密码的方法。客户修改密码通常需要确认客户身份，即客户需提供用户名(以邮箱为用户名)和密码，方法返回一个整数：1 - 密码修改成功；2 - 用户不存在；3 - 密码不正确；-1 - 程序异常(如没能连接到数据库等）。

#### 事务与转账操作

补充代码，实现一个银行卡转账的方法，方法返回boolean值，true表示转帐成功，false表示转账失败，并不需要细分或解释失败的原因。

下列任一情形都不可转账(转账失败的原因)：

转出或转入帐号不存在

转出账号是信用卡

转出帐号余额不足

#### 把稀疏表格转为键值对存储

一个表有很多列，但只有少数列存储有值，这样的表称为稀疏表。举个例子，大学里有几千名学生和几百门课程，为记录学生的成绩，设计这样一张表：每门课程占一列，每个学生占一行。但是，绝大多数学生只会选修这几百门课程中的一小部分，因此，整个表只会稀疏地存储了一些数据，这将产生大量的闲置空间。

一种更有效的方法是以(学号, 课程，成绩)这样的三元组形式存储。

设有高考成绩登记表entrance\_exam,其结构如下：

| **列名** | **类型** | **说明** |
| --- | --- | --- |
| sno | int | 学号，主码 |
| chinese | int | 语文 |
| math | int | 数学 |
| English | int | 英语 |
| physics | int | 物理 |
| chemistry | int | 化学 |
| biology | int | 生物 |
| history | int | 历史 |
| geography | int | 地理 |
| politics | int | 政治 |

转存表sc结构如下：

| **列名** | **类型** | **说明** |
| --- | --- | --- |
| sno | int | 学号 |
| col\_name | varchar(50) | 列名 |
| col\_value | varchar(50) | 列值 |

补充代码，完成转存任务。sc表初始为空表, 程序依前述规则将entrance\_exam表的值转写到sc表。对每一行，请从左至右依次考察每一列，转存非空列。

# 撰写课程实践报告

# 成绩占比说明

**实践课程序检查最终成绩**

**=中间进度检查成绩（占10%）+最终头哥检查成绩（占90%）**

**实践课综合成绩**

**=实践课程序检查最终成绩（占70%）+现场陈述与回答问题（占10%）+实践报告（占20%）**

中间进度成绩依据2次进度检查来评分，由指导教师依据完成情况评判，分别在第3周结束时和第5周结束时。