



**课 程 设 计**

课程名称\_\_\_\_\_\_数据库\_\_\_\_\_\_\_\_\_\_

题目名称\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生学院\_\_\_\_\_\_自动化学院\_\_\_\_\_\_

专业班级\_\_\_18级大数据2班\_\_\_\_

学 号\_\_\_3118001551\_\_\_\_\_\_\_\_\_

学生姓名\_\_\_\_\_李炜柯\_\_\_\_\_\_\_\_\_\_\_

指导教师\_\_\_\_\_黄永慧\_\_\_\_\_\_\_\_\_\_\_

20 年 月 日

## 1 本次课设所设计的数据库总步骤概述

###### 规划阶段

1. 系统调查（2）可行性分析

（3）确定数据库系统的总目标

###### 1.2 需求分析阶段

（1）分析用户活动，产生业务流程图（2）确定系统范围，产生系统关联图

（3）分析用户活动范围涉及的数据，产生数据流图（4）分析系统数据，参数数据字典

###### 1.3 概念设计阶段 （目标：产生反映用户需求的数据库概念结构，即概念模型）

（1）进行数据抽象，设计局部概念模型（2）将局部概念模型综合成全局概念模型（消除冲突）

（3）评审（用户评审and 应用开发人员评审）

方法：实体联系法（ER模型）（与DBMS无关的概念模型）

###### 1.4 逻辑设计阶段（目的：将设计好的概念模型转换成与DBMS所支持的数据模型相符合的逻辑）

结构（包括数据库逻辑模型和外模型）

步骤：

（1）把概念模型转化为逻辑模型（2）设计外模型（3）设计应用程序与数据库的接口

（4）评价模型（5）修正模型

###### 1.5 物理设计阶段

物理设计：对于给定的基本数据模型选取一个最合适应用环境的物理结构的过程。

步骤：

（1）存储记录结构设计（2）确定数据存放的位置（3）存取方法的设计（以上三个为物理结构设计）

（4）完整性和安全性考虑（5）程序设计

###### 1.6 数据库的实现

（1）用DDL定义数据库结构（2）组织数据入库

（3）编制与调试应用程序（4）数据库试运行

###### 1.7 数据库的运行与维护

（1）由DBA完成（2）数据库的转储和恢复（3）数据库的安全性、完整性控制

（4）数据库性能的监督、分析和改进（5）数据库的重组织和重构造

## 2 数据库的设计

###### 2.1规划阶段

本课设的移动电子商务以移动电子书城（提供浏览图书信息，购买图书等活动的电子商务平台）。所以通过分析淘宝，天猫等移动商务系统，我总结出了以下需求分析。

###### 2.2 需求分析阶段

该系统分为书城前台子系统（面向购书者用户）和书城后台管理子系统（面向购书系统管理员）





###### 2.3概念设计阶段

**2.3.1主体**

（1）用户：

基本的昵称，真名，性别，**手机号**，电子邮箱，密码，头像，身份证，生日， 居住地，余额。

（2）购物车：

用户手机号（主要就是体现一下购物车的存在，其实也可以用用户 来替代 这一主体）

（3）收藏夹：

用户的手机号（主要就是体现一下购物车的存在，其实也可以用用户来替代 这一主体）

（4）订单：

订单号，用户的手机号，管理员的证书号，商家的证书号。

（5）管理员：

管理员的证书、基本的昵称，真名，性别，手机号，电子邮箱，密码，头像， 身份证，生日，居住地。

（6）商家：

制造产地，证书号，

（7）商品：

商品号，名称，价格，出版社，作者，商家的证书号，管理员的证书号，出 版时间，简介，图片，商品所在的网址，类型。

**2.3.2联系**

（1）用户、购物车、商品之间的联系。

用户拥有购物车：用户:购物车=1:1  
商品加入购物车：商品:购物车=n:m

（2）用户、订单、商品之间的联系。

用户下单订单：用户:订单=1:n【UML模型(1和0…\*)】  
订单订购商品：订单:商品=n:m【UML模型(0…\*和0…\*)】

（3）用户、收藏夹、商品之间的联系

用户拥有收藏夹：用户:收藏夹=1:n【UML模型(1和0…\*)】  
收藏夹收藏商品：收藏夹:商品=n:m【UML模型(0…\*和0…\*)】

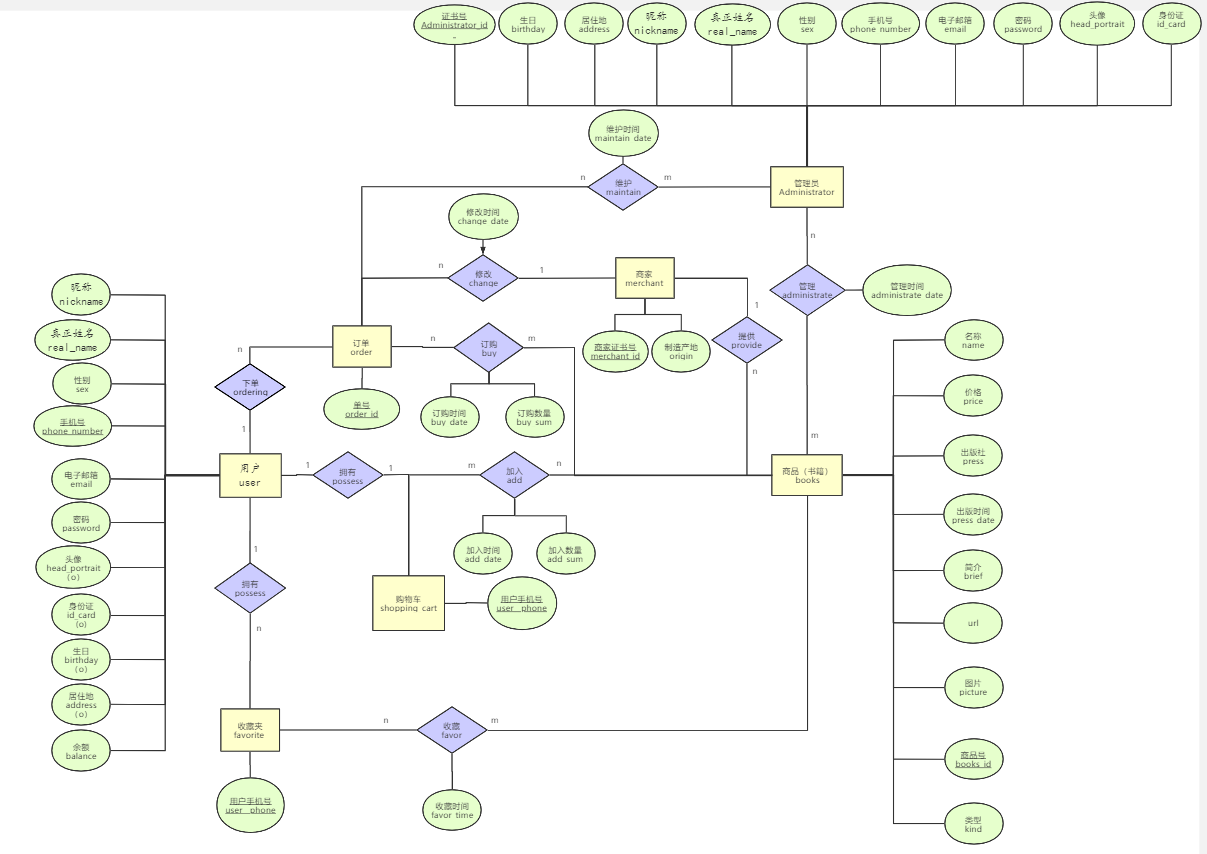
（4）管理员、订单的联系  
管理员维护订单：管理员:订单=n:m【UML模型(0…\*和0…\*)】

（5）管理员、商品的联系  
管理员管理商品：管理员:商品=n:m【UML模型(0…\*和0…\*)】

（6）商家、商品的联系  
商家提供商品：商家:商品=1:n【UML模型(1和0…\*)】

（7）商家、订单的联系  
商家修改订单：商家:订单=1:n【UML模型(1和0…\*)】

**2.3.3 ER图的设计**



###### 2.4逻辑设计阶段

这里我们需要联系类型转化为关系模型。

二元的转换的公式：

（1）若实体间的联系是1:1，可以再两个实体类型转换成的两个关系模式中任意的一个关系模式的属性加入另一个关系模式的键和联系类型的属性。

（2）若实体间的联系是1:n，则在n端的实体类型转换成的关系模式中加入1端实体类型的键和联系类型的属性。

（3）若实体间的联系是m:n,则将联系类型也转换成关系模式，其属性为两端实体类型的键加上联系类型的属性，而键为两端实体键的组合。

所以我们最后除了主体还能得到由n:m的概念模型转化关系模型生成的其他表：

订单&商品（下单）：订单号，商品号，商品下单的时间，下单商品的数量，

收藏夹&商品（收藏）：商品加入收藏夹的时间，商品号，用户手机号，

商品&购物车（加入购物车）：用户的手机号，商品号，商品加入购物车的时间，商品的数量。

管理员&订单（维护）：管理员的证书号，订单号，维护时间

管理员&商品（管理）：管理员的证书号，商品号，管理时间

###### 2.5 物理设计阶段

到了这里，就可以设计一些触发器还有储存过程来辅助我们的程序设计了。

###### 2.5.1 本次实验设计的触发器：

1. 日志表：

当用户进行增删改操作时，记录用户操作信息。

当商品进行增删改操作时，记录商品修改的信息。

当管理员进行操作时，记录了管理员的操作信息。

（2）行为联系：

当新建立一个用户时，建立对应的收藏夹，购物车，订单。

当删除书籍时，同时删除收藏夹，购物车，订单中存放的这类书籍的信息。

当用户购买书籍时，从购物车中删除商品，同时从余额中减去所花费的金额。

这里附上部分触发器的代码：

//创建insert型触发器，当用户购买后，扣除余额  
delimiter $  
create trigger user\_buy\_after  
after insert on `buy`  
for each row  
begin  
      update user,books,`order` set user.balance=user.balance-new.buy\_sum\*books.price where `order`.order\_id=new.order\_id and user.phone\_number=`order`.phone\_number and books.books\_id=new.books\_id;  
      delete `add` from books,`add`,shopping\_cart,`order` where `order`.order\_id=new.order\_id and `order`.phone\_number=`add`.cart\_user\_phone and `add`.books\_id=new.books\_id;  
    end $  
delimiter ;

//创建用户日志表  
create table user\_logs(  
    id int(11) not null auto\_increment,  
    operation varchar(20) not null comment "操作类型,insert/update/delete",  
    operate\_time datetime not null comment "操作时间",  
    operate\_phone char(11) not null comment "操作表的用户",  
    operate\_params varchar(500) comment "操作参数",  
    primary key(`id`)  
)engine=innodb default charset=utf8;

//创建 insert 型触发器，完成插入数据时的日志记录:  
delimiter $  
create trigger user\_logs\_insert\_trigger  
after insert on user  
for each row  
begin  
    insert into user\_logs(id,operation,operate\_time,operate\_phone,operate\_params)  
    values(null,"insert",now(),new.phone\_number,concat("插入后(nickname:",ifnull(new.`nickname`,""),",realname:",ifnull(new.`real\_name`,""),",sex:",ifnull(new.`sex`,""),",phone\_number:",ifnull(new.`phone\_number`,""),",email:",ifnull(new.`email`,""),",password:",ifnull(new.`password`,""),",head\_portrait:",ifnull(new.`head\_portrait`,""),",id\_card:",ifnull(new.`id\_card`,""),",birthday:",ifnull(new.`birthday`,""),",address:",ifnull(new.`birthday`,""),",balance:",ifnull(new.`balance`,""),",remember\_password:",ifnull(new.`remember\_password`,""),")"));  
end $  
delimiter ;

**2.5.2 本次实验设计的储存过程和储存函数：**

|  |  |
| --- | --- |
| 创建用户购物车和收藏夹和订单 | procedure create\_user\_cart\_favorite |
| 创建商品 | procedure create\_books |
| 创建商家 | procedure create\_merchant |
| 创建管理员 | procedure create\_administrator |
| 商品加入购物车 | procedure create\_add |
| 订单订购商品 | procedure create\_buy |
| 收藏夹收藏商品 | procedure create\_favor |
| 获取用户的剩余金额 | procedure user\_balance |
| 判断用户存不存在(储存过程实现) | procedure user\_exis |
| 判断用户存不存在(储存函数实现) | function user\_exist |
| 充钱 | procedure charge\_money |
| 花钱 | procedure cost\_money |
| 用户登陆判断密码是否正确(手机号或邮箱都可登陆) | procedure judge\_password |
| 管理员登陆判断密码是否正确(手机号或邮箱都可登陆) | procedure judge\_ad\_password |
| 管理员登陆认证 | procedure judge\_ad\_password |
| 根据手机号或邮箱获取用户名 | procedure obtain\_nickname |
| 根据手机号或邮箱获取用户信息 | procedure obtain\_user\_information |
| 根据手机号或邮箱获取管理员姓名 | procedure obtain\_ad\_nickname |
| 根据手机号或邮箱获取用户手机号 | procedure obtain\_user\_phone\_number |
| 根据书号获取书的信息 | procedure obtain\_books |
| 修改用户信息 | procedure update\_user |
| 修改管理员信息 | procedure update\_administrator |
| 删除用户收藏信息 | procedure delete\_favor |
| 删除购物车 | procedure delete\_cart |
| 购买操作 | procedure buy\_books |
| 获取管理员信息 | procedure `obtain\_admin\_information` |
| 修改书籍信息 | procedure update\_books |
| 删除书籍 | procedure delete\_books |
| 删除用户 | procedure delete\_user |

下面附上部分储存过程代码：

//用户登陆判断密码是否正确(手机号或邮箱都可登陆)  
delimiter $  
create procedure judge\_password(in phone\_or\_email varchar(20),in passwords char(20),out ok boolean)  
begin     
    declare ok1,ok2 int;  
    select count(\*) into ok1 from user where phone\_number=phone\_or\_email and `password`=passwords;    
    select count(\*) into ok2 from user where email=phone\_or\_email and `password`=passwords;  
    if ok1=1 or ok2=1 then set ok=true;  
    else set ok=false;  
    end if;  
end $  
delimiter ;

//购买操作  
delimiter $  
create procedure buy\_books(in phone\_numbers char(11),in books\_ids int)  
begin  
      declare order\_ids int;  
      declare buy\_sums int;  
      declare buy\_dates datetime;  
      set buy\_dates=now();  
      select order\_id into order\_ids from `order` where phone\_number=phone\_numbers;  
      select add\_sum into buy\_sums from `add` where cart\_user\_phone=phone\_numbers and books\_id=books\_ids;  
      insert into `buy`(order\_id,books\_id,buy\_date,buy\_sum) values(order\_ids,books\_ids,buy\_dates,buy\_sums);  
end $  
delimiter ;  
​

//删除用户（因为数据库的约束关系，一定要按这个顺序删除）  
delimiter $  
create procedure delete\_user(in phone\_numbers char(11))  
begin  
     declare order\_ids int;  
     select order\_id into order\_ids from `order` where phone\_number=phone\_numbers;  
     delete from `buy` where order\_id=order\_ids;  
     delete from `order` where phone\_number=phone\_numbers;  
     delete from `add` where cart\_user\_phone=phone\_numbers;  
     delete from `shopping\_cart` where cart\_user\_phone=phone\_numbers;  
     delete from `favor` where fav\_user\_phone=phone\_numbers;  
     delete from `favorite` where fav\_user\_phone=phone\_numbers;  
     delete from `user` where phone\_number=phone\_numbers;  
end$  
delimiter ;  
​

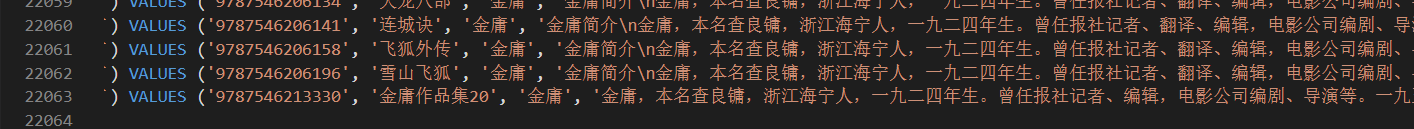
//获取用户的剩余金额  
delimiter $  
create procedure user\_balance(in phone\_numbers char(11),out ans decimal(30,2))  
begin  
      declare cnt int;  
      select count(\*) into cnt from user where phone\_number=phone\_numbers;  
      if cnt=0 then   
      set ans=-1.00;  
      else select balance into ans from user where user.phone\_number=phone\_numbers;  
      end if;  
end $  
delimiter ;  
​

​

###### 2.6 数据库的实现

将以上实验的代码运行到服务器的数据库中，之后便开始处理书籍的数据了。

这里我是去网上找了豆瓣的书籍的数据集，



一共20000多条数据，将此数据按我们所需要的信息导入数据。

就此我们基本实现了数据库的构建。

###### 1.7 数据库的运行与维护

对于将数据库用于真正实践即和高级编程语言合作时，会出现很多bug，得对数据库进行细微的调整。比如数据范围太小，类型java获取不到等等，都得进行细微的调整。

同时，为了信息的安全我还建立了三个购物车，订单，收藏夹的视图。下面是这三个视图的代码：

//创建购物车视图  
create view shopping\_cart\_view(`name`,`sum`,single\_price,user\_phone,add\_time,books\_id)  
as  
select books.name,`add`.add\_sum,books.price,`add`.cart\_user\_phone,`add`.add\_date,books.books\_id from   
books,`add`,shopping\_cart where `add`.books\_id=books.books\_id and `add`.cart\_user\_phone=shopping\_cart.cart\_user\_phone;  
​  
//创建订单视图  
create view order\_view(order\_id,`name`,`sum`,single\_price,user\_phone,add\_time,books\_id)  
as  
select `buy`.order\_id,books.name,`buy`.buy\_sum,books.price,`order`.phone\_number,`buy`.buy\_date,books.books\_id from  
books,`order`,`buy` where `buy`.books\_id=books.books\_id and `order`.order\_id=`buy`.order\_id;

//创建收藏夹视图  
create view favor\_view(books\_id,`name`,single\_price,user\_phone,favor\_time,Author)  
as  
select books.books\_id,books.name,books.price,`favor`.fav\_user\_phone,`favor`.favor\_date,books.Author from  
books,`favor`,`favorite` where `favor`.books\_id=books.books\_id and `favor`.fav\_user\_phone=`favorite`.fav\_user\_phone;

至此，在该项目本人所负责的数据库部分已经基本上完成了，代码量大概为1000行左右。

###### 3 心得体会：

全部从零开始设计加上实现数据库的过程，是很艰辛坎坷的。中间画了很多个ER图都不是很满意，只能一直百度其他人的ER图来分析自己的缺漏，慢慢的才能设计出一个略为满意的ER图。同时，经过这次课设，更加固了我对于数据库的代码能力以及对于问题的搜索处理能力。同时，也让我对数据库的整体结构有了更全面的了解。现在及以后，数据库对于我们都是极其重要的存在，在以后我也会继续对数据库的学习，让自己在这方面的能力能够更加成熟。