**《数据库系统原理》实验报告**

实验题目：函数与事件

姓名：刘俊傲 实验日期：2018年 12月 25日

**实验内容及完成情况：**

1. 首先建立各数据表
2. Sotrebook 表： 用于网站书籍库存

CREATE TABLE `storebook` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`num` int(11) NOT NULL COMMENT '书号',

`name` varchar(255) NOT NULL COMMENT '书名',

`author` varchar(255) NOT NULL COMMENT '作者，最多4个，有序',

`publish` varchar(255) NOT NULL COMMENT '出版社',

`money` int(11) NOT NULL COMMENT '价格',

`keyword` varchar(255) DEFAULT NULL COMMENT '关键字',

`dir` text COMMENT '书籍目录',

`cover` blob COMMENT '封面',

`count` int(11) NOT NULL COMMENT '存货量',

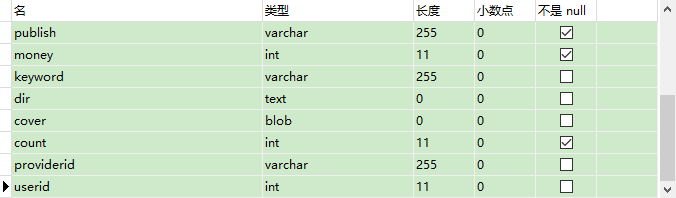
`providerid` varchar(255) DEFAULT NULL COMMENT '供应商ID，可有多个',

`userid` int(11) DEFAULT NULL

COMMENT '对应 storeuser 中的 id，用于记录当前或最近订书的用户',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='对应于网站书籍库存';

****

1. Missbook 表： 用于记录书籍存货量低于一定数量的书籍信息

CREATE TABLE `missbook` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`num` int(11) NOT NULL COMMENT '书号',

`name` varchar(255) NOT NULL COMMENT '书名',

`publish` varchar(255) NOT NULL COMMENT '出版社',

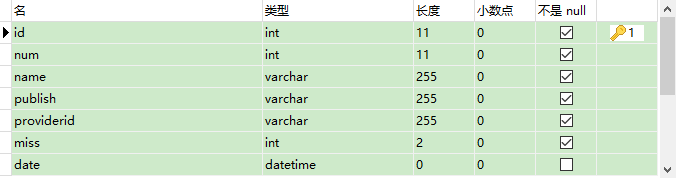
`providerid` varchar(255) NOT NULL COMMENT '供应商ID',

`miss` int(2) NOT NULL COMMENT '是否缺失：\r\n0： 不缺失\r\n1： 缺失',

`date` datetime DEFAULT NULL COMMENT '登记日期',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用于记录书籍存货量低于一定数量的书籍信息';

****

1. Missuser 表： 记录顾客的缺书请求以便答复

CREATE TABLE `missuser` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`userid` int(11) DEFAULT NULL COMMENT '缺货用户ID，对应 storeuser 表',

`username` varchar(255) DEFAULT NULL COMMENT '缺货用户名',

`userbooknum` int(11) DEFAULT NULL COMMENT '缺货书籍的书号',

`count` int(11) DEFAULT NULL COMMENT '缺书的数量',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='记录顾客的缺书请求以便答复';

****

1. Storeuser 表： 用于保存用户的相关信息

CREATE TABLE `storeuser` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '网上 id，主键',

`name` varchar(255) NOT NULL COMMENT '客户名',

`password` varchar(255) NOT NULL COMMENT '登陆密码',

`address` varchar(255) NOT NULL COMMENT '地址',

`balance` int(11) NOT NULL COMMENT '余额',

`debt` int(11) DEFAULT NULL COMMENT '欠款：等级三四有限额，等级五无限额',

`grade` int(2) NOT NULL

COMMENT '信用等级：

一级：10% 折扣，不能透支

二级：15% 折扣，不能透支

三级：15% 折扣，透支限额，可先发书再付款

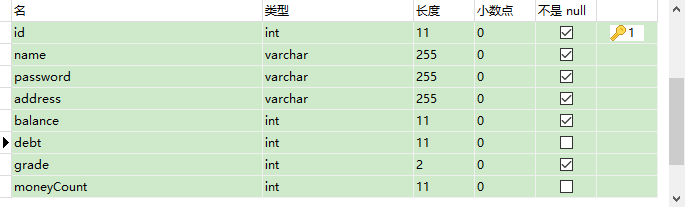
四级：20% 折扣，透支限额，可先发书再付款

五级：25% 折扣，透支无限额，可先发书再付款',

`moneyCount` int(11) DEFAULT NULL COMMENT '累计充值',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用于保存用户的相关信息';

****

1. Order 表： 记录用户订单信息

CREATE TABLE `order` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '订单号，主键',

`date` datetime NOT NULL COMMENT '订货日期',

`userid` int(11) DEFAULT NULL COMMENT '客户ID',

`orderbyid` int(255) DEFAULT NULL COMMENT '对应 orderbyid 表中的 ID',

`money` int(11) DEFAULT NULL COMMENT '金额',

`address` varchar(255) DEFAULT NULL COMMENT '发货地址',

`condition` int(2) DEFAULT NULL COMMENT '发货情况：

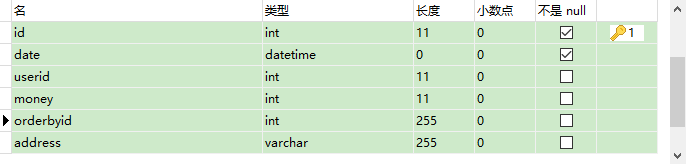
0： 未发货

1： 已发货，但客户还未收到

2： 已发货，且客户已收到',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户订单信息';

****

1. Allbook 表： 记录供应商的相关信息

CREATE TABLE `provider` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`name` varchar(255) DEFAULT NULL COMMENT '供应商姓名',

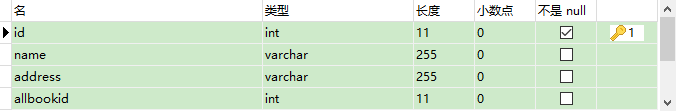
`address` varchar(255) DEFAULT NULL COMMENT '供应商地址',

`allbookid` int(11) DEFAULT NULL

COMMENT '供应商所拥有的书籍对应的allbook 表信息',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='供应商的相关信息';

****

1. Allbook 表： 记录所有书籍相关信息，对应供应商提供的书籍信息

CREATE TABLE `allbook` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`num` int(11) NOT NULL COMMENT '书号',

`name` varchar(255) DEFAULT NULL COMMENT '书名',

`author` varchar(255) DEFAULT NULL COMMENT '作者，最多四个，有序',

`publish` varchar(255) DEFAULT NULL COMMENT '出版社',

`money` int(11) DEFAULT NULL COMMENT '价格',

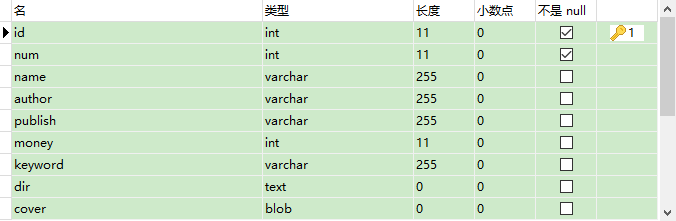
`keyword` varchar(255) DEFAULT NULL COMMENT '关键字',

`dir` text COMMENT '书记目录',

`cover` blob COMMENT '书记封面',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='所有书籍相关信息，对应供应商提供的书籍信息';

****

1. orderbyid 表： 用于存储同一用户的订单信息，可同时预订多本书籍

CREATE TABLE `orderbyid` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`bookid` int(11) DEFAULT NULL COMMENT '书号',

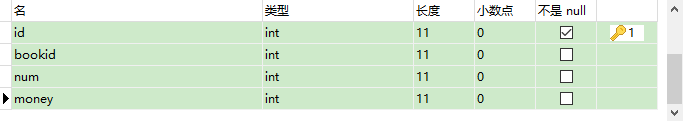
`num` int(11) DEFAULT NULL COMMENT '该本书预定的数量',

`money` int(11) DEFAULT NULL COMMENT '该类型书的总价格',

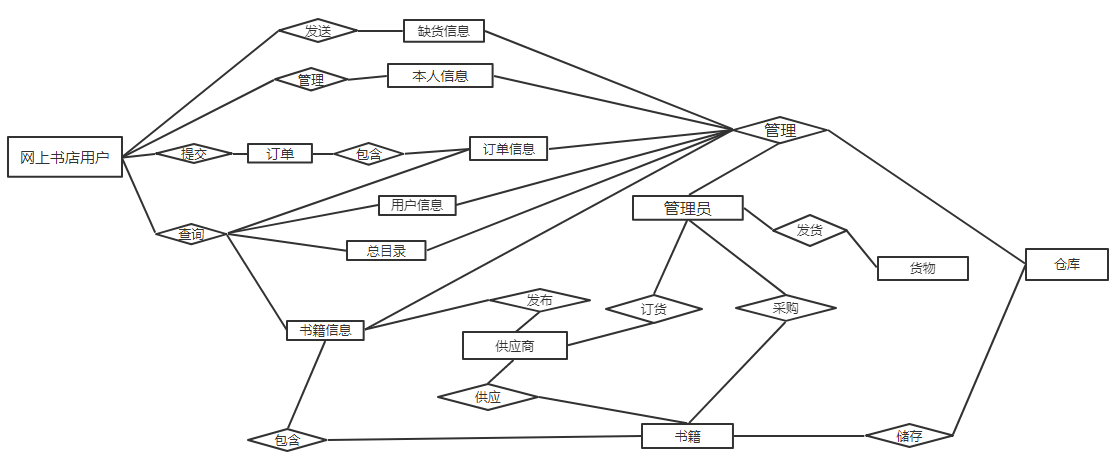
PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

COMMENT='用于存储同一用户的订单信息，可同时预订多本书籍';

****

因此给出总的 E-R 图：

****

1. 供书目录及库存管理

① 定义 insertNewBook 触发器： 用于实现当向 storebook 表中插入新的数据时，同时 也向 missbook 表中插入对应数据，方便对于 missbook 表的管理

DELIMITER $

CREATE TRIGGER insertNewBook

AFTER INSERT ON storebook

FOR EACH ROW

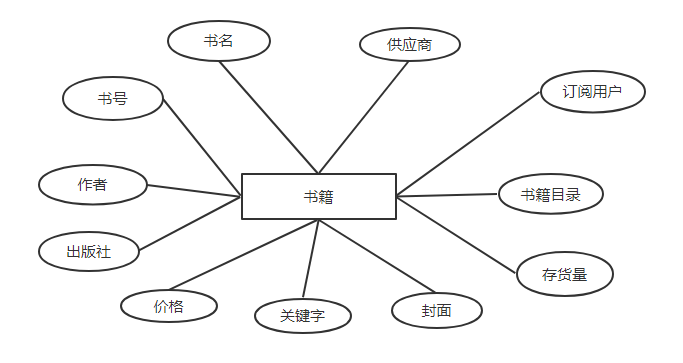
BEGIN

INSERT INTO missbook(num,name,publish,providerid,miss,date)

VALUES(NEW.num,NEW.name,NEW.publish,NEW.providerid,0,NOW());

END $

DELIMITER ;



1. 采购管理
2. 当用户购买书籍时，可在程序中调用 sql 语句对 storebook 表进行实时查询，若书籍 存量不足，则进行缺失登记

UPDATE missbook SET miss=1,date=NOW()

WHERE 5 > (

SELECT count FROM storebook WHERE id=@id; #@id 为用户购买书籍时传入

);

② 定义 misscount 触发器： 用于实现当 storebook 中的书籍存量低于 5 本时，在 missbook 记录对应缺失书籍信息

DELIMITER $

CREATE TRIGGER misscount

AFTER UPDATE ON storebook

for EACH ROW

BEGIN

IF(NEW.count < 5)

THEN

UPDATE missbook SET miss=1,date=NOW() WHERE num=OLD.num;

ELSE UPDATE missbook SET miss=0,date=NOW() WHERE num=OLD.num;

END IF;

END $

DELIMITER ;

③ 定义 insertNewBook 存储过程： 用于实现当顾客在网上进行缺书登记或订货数量超出库 存时，生成缺书记录（此处因触发器不支持复杂的变量获取操作，因此改为存储过程，可 用于程序中的手动调用）

DELIMITER $

CREATE PROCEDURE insertNewBook(in uid INT(11),in bnum INT(11))

BEGIN

set @uname = (SELECT name from storeuser WHERE id=uid);

INSERT INTO missuser(userid,username,userbooknum,count)

VALUES(uid,@uname,bnum,1);

END $

DELIMITER ;

1. 客户管理

书店管理：

1. 用户充值：

update storeuser set balance = @balance;

1. 调整信用额度：

定义 checkBalance 存储过程： 用于每月月初调用该过程实现对用户等级的升迁

DELIMITER $

CREATE PROCEDURE checkBalance(in balance INT,in moneyCount INT,in uid INT)

BEGIN

SET @grade = (SELECT grade FROM storeuser WHERE id=uid);

IF(balance > 500)

THEN

UPDATE storeuser SET grade=@grade+1 WHERE @grade<5;

ELSE IF(moneyCount > 5000)

THEN

UPDATE storeuser SET grade=@grade+1 WHERE @grade<5;

END $

DELIMITER ;

客户网上管理：

① 注册： insert into storeuser

(name,password,address,balance,grade,monyCount)

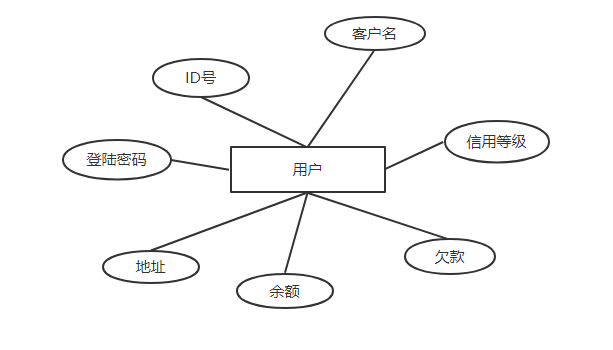
Values(@name,@password,@address,0,1,0);

② 修改用户信息：

修改用户名： update storeuser set name=@name;

修改密码：update storeuser set password=@password;

修改地址： update storeuser set address=@address;



1. 顾客订单管理和发货管理
2. 客户订单：

首先插入 orderbyid 表： insert into orderbyid(bookid,num,money)

Values(@bookid,@num,@money)

#@bookid: 为预定书记号

#@num: 为对应的预定书籍数量

#@money: 为相应的金额

接着插入 order 表： insert into order

(date,userid,money,orderbyid,address,condition)

values(NOW(),@uid,@money,@orderbyid,@address,@condition);

#@uid: 为用户 ID

#@money: 为该用户此次预定书籍的总金额

#@orderbyid: 为对应 orderbyid 表的 ID

#@address: 为用户地址信息

#@condition: 为用户情况描述

最后修改 storebook 表中的对应书籍数量信息

1. 当书籍数量不足时： 自动提示用户书籍数量不足，当新书籍到货时，是否提醒
2. 付款：

定义 buy 存储过程： 用于处理在付款过程中限制条件

DELIMITER $

CREATE PROCEDURE buy(in grade INT,in money INT,in uid INT,

in grade3 INT,in grade4 INT)

BEGIN

SET @balance=(SELECT balance FROM storeuser WHERE id=uid);

CASE

WHEN grade=1 THEN

UPDATE storeuser SET balance=@balance-money\*0.9;

WHEN grade=2 THEN

UPDATE storeuser SET balance=@balance-money\*0.85;

WHEN grade=3 THEN

IF(@balance < @tmp=money\*0.85)

THEN

SET @tmp=money\*0.85 - @balance;

IF(grade3 > @tmp)

THEN

UPDATE storeuser SET balance=0;

UPDATE storeuser SET debt=@tmp;

END IF;

ELSE UPDATE storeuser SET balance=@balance-money\*0.85;

END IF;

WHEN grade=4 THEN

IF(@balance < @tmp=money\*0.8)

THEN

SET @tmp=money\*0.8 - @balance;

IF(grade3 > @tmp)

THEN

UPDATE storeuser SET balance=0;

UPDATE storeuser SET debt=@tmp;

END IF;

ELSE UPDATE storeuser SET balance=@balance-money\*0.8;

END IF;

WHEN grade=5 THEN

IF(@balance < @tmp=money\*0.75)

THEN

SET @tmp=money\*0.75 - @balance;

UPDATE storeuser SET balance=0;

UPDATE storeuser SET debt=@tmp;

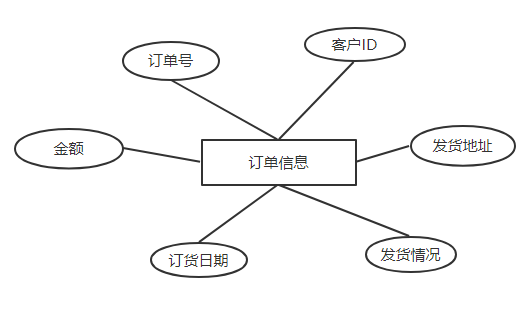
ELSE UPDATE storeuser SET balance=@balance-money\*0.75;

END IF;

END CASE;

END $

DELIMITER ;



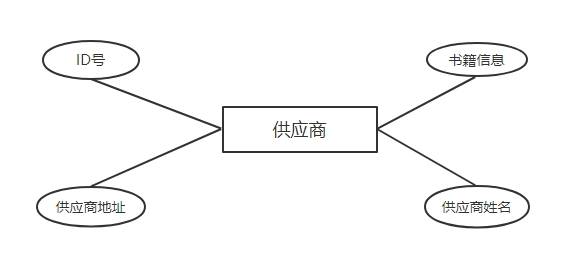
1. 供应商管理

供应商发布自己的相关信息：

Select \* from allbook where id=@pid;

#@pid: 对应供应商的 allbookid，表示只查询特定供应商的数目信息

给出此处的 E-R 图：



1. 网上浏览查询
2. 客户相关信息：

客户基本信息： select name,address,grade from storeuser;

客户历史订单信息：

先查询用户记录：select \* from order where userid=@uid;

再查询记录下的书籍信息： select \* from orderbyid where id=@oid;

#@oid: 为对应 storeuser 中的orderbyid

相关订单的发货信息： select condition from order where id=@oid

#@oid: 为对应的订单号

0： 未发货

1： 已发货，但客户还未收到

2： 已发货，且客户已收到

1. 书目信息：

书号： select \* from storebook where num=@num;

模糊查询： select \* from storebook where num like ’%xxx%’;

书名： select \* from storebook where name=@name;

模糊查询： select \* from storebook where name like ’%xxx%’;

出版社： select \* from storebook where publish=@publish;

模糊查询： select \* from storebook where publish like ’%xxx%’;

关键字： select \* from storebook where keyword like ‘%keyword %’;

作者： select \* from storebook where author like ‘%author %’;

#author 字段: 多个作者以 ‘,’ 分隔

1. 注意
2. 未防止出错导致数据库崩溃，对于数据表间的关联字段并未使用外键，其中的完整性 检查靠多余的 sql 代码来实现
3. 对于其中的存储过程方法，可手动调用，也可定义程序，定期调用
4. 对于多个作者按顺序查询，我还未想到可行且高效的方案

出现的问题及解决方案：

触发器中不可以使用复杂的操作，如： 不可以获取额外数据

教师评语及成绩：