数据库技术第四次上机作业

张适可 515021910306

1. 在数据库的设计过程中，为了保证3NF，你做了哪些调整？3NF在本数据库中带来的好处是什么（如果有）？坏处是什么（如果有）？

答：把客户积分、等级和折扣率表拆开，消除传递依赖。好处是降低了表内信息的冗余度，坏处是增加了查表的复杂程度。

2.内容1（写出下列三步操作对应的sql语句)

(1) 随机插入10名供应商的信息 + (3) 供应商发布四类商品的情况

为了方便我的供应商插入和四类商品各四种插入是一起的。结果是供应商(store)10行，商品(commodity)和库存(reservation)各160行。其中库存是随机的，商品价值设置在20到30之间。

代码如下：(procedure)

```
use m3DBS;

DELIMITER $$

create procedure insert_store()
begin
declare num char(10);
declare od int;
declare cnt int;
declare c_id char(10);
declare g_num int;
declare genre_id int;
declare genre_num int;
declare cnt_2 int;
declare cnt_3 int;
declare price int;
    set od = (select count(1) from store);
    set genre_num = (select count(1) from commodity);
    set num = od;
    set cnt = 1;
    set cnt_2 = 1;
    set cnt_3 = 1;
    set genre_num = (select count(1) from commodity);

    while   cnt <= 10 do
        insert into store value(concat('s',num));
            while cnt_3 <= 4 do
                while cnt_2 <=4 do
```

```
                    insert into commodity value(concat('g',genre_num),
concat(concat('s',num), '_', cnt_3, '_', cnt_2 ), 20+genre_num%10, rand(),
cnt_3, null,null,genre_num+rand()*20,'1');
                    insert into reservation value(concat('s',num),
concat('g',genre_num));
                    set genre_num = genre_num + 1,
                            cnt_2 = cnt_2 + 1;
                end while;
                set cnt_3 = cnt_3 +1,cnt_2 = 1;

            end while;

            set c_id = c_id + 1 ,
                    num = num+1,
                    cnt = cnt+1,
                    cnt_3 = 1;
    end while;
 end$$

 show create procedure insert_store;


DELIMITER ;
```

(2)随机插入100名客户的信息

这里和客户信息相关的表有三个：customer，CA，account。

其中的CA可以理解为一个“注册”，所以依赖关系是customer中的“phone_number”决定CA中的“phone_number”，然后系统分配一个账号，因此CA中的账号决定account中的账号表。这三张表是相关的，所以update是一起进行的。

代码如下：(procedure)

```
drop procedure insert_customer;

DELIMITER $$

create procedure insert_customer()
begin
declare num char(10);
declare od int;
declare base int;
declare cnt int;
declare age int;
declare id char;
set base = 80000000;
set od = (select count(1) from store);
set id = od;
set num = base+od;
set cnt = 1;
set age = 20+59*rand();
    while   cnt <= 100 do
       insert into customer value(num, 'john', 'm', age);
       insert into CA value(num, concat('c', id));
       insert into account value(concat('c', id), '0', '1');
       set od = od+1;
       set id = od;
       set num = base + od;
       set cnt = cnt + 1;
    end while;
 end$$
 show create procedure insert_customer;
DELIMITER ;
```

3. 内容2

（1）给出以下操作对应的sql语句和查询结果：“查询所有客户中年龄最大和最小的用户的名字（记为customer1和customer2）和享受的额外折扣率”。

sql语句：

```
   select t1.name, t2.disCnt from customer as t1 left join (CA as t3
natural join cDiscount as t2)
    on t1.phone_number = t3.phone_number and t3.ID = t2.ID
    where age = (select min(age) from customer) or age = (select max(age)
from customer);
```

查询结果：

```
mysql>    select t1.name, t2.disCnt from customer as t1 left join (CA as t3 natu
ral join cDiscount as t2)
    -> on t1.phone_number = t3.phone_number and t3.ID = t2.ID
    -> where age = (select min(age) from customer) or age = (select max(age) fro
m customer);
+---------+--------+
| name    | disCnt |
+---------+--------+
| john_68 |    0.9 |
| john_69 |    0.9 |
+---------+--------+
2 rows in set (0.00 sec)
```

（2）给出customer1下单对应的sql语句 （若分步进行，依次给出sql语句和中间结果）

sql语句：

sql语句1(在procedure里面，用cursor遍历选择的结果)：

```
/*找出最便宜的商品g17,g138,g31,g68*/
DECLARE cur1 cursor for select c_id from commodity where price*discnt in
(select min(price*discnt) from commodity group by genre);
```

sql语句2：

```
/*建立订单+买家-订单联系*/
insert into purchase*
    VALUES('oc10', 'c68');
insert into ord
    VALUES('oc10', 0, null, null, '0', null, '000000', '0');
```

sql语句3:

```
/*遍历,把所有商品插入ord_c表*/
open cur1;
read_loop_1: loop

fetch cur1 into cid_c1;

if done=1 then
    leave read_loop_1;
end if;

set num = 3+rand()*10;

insert into ord_c
VALUES('oc10', cid_c1, num);

end loop;
```

结果：(和客户2下单后结果一起)

1. ord表

```
mysql> select*from ord;
+-------+---------+------+--------+-------+------+-----------+-------+
| od_id | money   | dT   | out_dT | state | addr | mail_addr | score |
+-------+---------+------+--------+-------+------+-----------+-------+
| oc10  | 18.2862 | NULL |   NULL | 4     | NULL | 000000    |     0 |
| oc11  | 4850.86 | NULL |   NULL | 3     | NULL | 000000    |     0 |
+-------+---------+------+--------+-------+------+-----------+-------+
2 rows in set (0.00 sec)
```

2.ord_c表

```
mysql> select*from ord_c;
+-------+------+--------+
| od_id | c_id | number |
+-------+------+--------+
| oc10  | g138 |     12 |
| oc10  | g17  |      9 |
| oc10  | g31  |      8 |
| oc10  | g68  |     10 |
| oc11  | g147 |    165 |
| oc11  | g150 |    168 |
| oc11  | g152 |    169 |
| oc11  | g157 |    168 |
| oc11  | g158 |    168 |
+-------+------+--------+
9 rows in set (0.00 sec)
```

3.purchase 表

```
+-------+-----+
| od_id | ID  |
+-------+-----+
| oc10  | c68 |
| oc11  | c69 |
+-------+-----+
2 rows in set (0.00 sec)
```

（3）给出customer2下单对应的sql语句（若分步进行，依次给出sql语句和中间结果）

sql语句1:

```
/*找出库存最多的商品—g157,g158,g152,g147,g150 (p.s: 其中g157和g158的库存一样多)*/
DECLARE cur2 cursor for select c_id,remain from commodity where remain in
(select max(remain) from commodity group by genre);
```

sql语句2:

```
/*建立订单+买家-订单*/
insert into purchase
        VALUES('oc11', 'c69');
insert into ord
        VALUES('oc11', 0, null, null, '0', null, '000000', '0');
```

sql语句3:

```
/*遍历,把所有商品插入ord_c表*/
open cur2;
read_loop_2: loop

fetch cur2 into cid_c2, rm;

if done=1 then
    leave read_loop_2;
end if;

insert into ord_c
VALUES('oc11', cid_c2, rm);
end loop;
```

结果：

1.ord表（money是自动的，通过trigger实现的）

```
mysql> select*from ord;
+-------+---------+------+--------+-------+------+-----------+-------+
| od_id | money   | dT   | out_dT | state | addr | mail_addr | score |
+-------+---------+------+--------+-------+------+-----------+-------+
| oc10  | 18.2862 | NULL |   NULL | 4     | NULL | 000000    |     0 |
| oc11  | 4850.86 | NULL |   NULL | 3     | NULL | 000000    |     0 |
+-------+---------+------+--------+-------+------+-----------+-------+
2 rows in set (0.00 sec)
```

2.ord_c表

```
mysql> select*from purchase;
+-------+-----+
| od_id | ID  |
+-------+-----+
| oc10  | c68 |
| oc11  | c69 |
+-------+-----+
2 rows in set (0.00 sec)
```

3.purchase表

```
mysql> select*from purchase;
+-------+-----+
| od_id | ID  |
+-------+-----+
| oc10  | c68 |
| oc11  | c69 |
+-------+-----+
2 rows in set (0.00 sec)
```

4. commodity表

```
mysql> select*from commodity where c_id='g147'or c_id='g150' or c_id='g152' or c
_id='g157' or c_id='g158';
+------+--------+-------+-----------+-------+-------+-------+--------+-------+
| c_id | name   | price | discnt    | genre | dateT | presT | remain | score |
+------+--------+-------+-----------+-------+-------+-------+--------+-------+
| g147 | s9_1_4 |    27 |  0.278347 | 1     |  NULL |  NULL |      0 |     1 |
| g150 | s9_2_3 |    20 |  0.410816 | 2     |  NULL |  NULL |      0 |     1 |
| g152 | s9_3_1 |    22 | 0.0444492 | 3     |  NULL |  NULL |      0 |     1 |
| g157 | s9_4_2 |    27 |   0.12251 | 4     |  NULL |  NULL |      0 |     1 |
| g158 | s9_4_3 |    28 |  0.435482 | 4     |  NULL |  NULL |      0 |     1 |
+------+--------+-------+-----------+-------+-------+-------+--------+-------+
5 rows in set (0.00 sec)
```

（4）给出customer1退货对应的sql语句（若分步进行，依次给出sql语句和中间结果）

sql语句：

```
insert into back_od
VALUES('oc10', 'no reason');
```

结果：

1.ord表

```
mysql> select*from ord;
+-------+---------+------+--------+-------+------+-----------+-------+
| od_id | money   | dT   | out_dT | state | addr | mail_addr | score |
+-------+---------+------+--------+-------+------+-----------+-------+
| oc10  | 18.2862 | NULL |   NULL | 4     | NULL | 000000    |     0 |
| oc11  | 4850.86 | NULL |   NULL | 3     | NULL | 000000    |     0 |
+-------+---------+------+--------+-------+------+-----------+-------+
2 rows in set (0.00 sec)
```

2.back_od表

```
mysql> select*from back_od;
+-------+-----------+
| od_id | reasons   |
+-------+-----------+
| oc10  | no reason |
+-------+-----------+
1 row in set (0.01 sec)
```

（5）给出customer2接受订单对应的sql语句（若分步进行，依次给出sql语句和中间结果）

sql语句:

```
insert into cmt
VALUES('oc11', 'good', null, 5);*

update ord
set state='3' where od_id='oc11';
```

结果：

1.积分变化(c69分数+3)

```
mysql> select*from member where ID='c69' or ID='c68';
+-----+-------+
| ID  | score |
+-----+-------+
| c68 |    18 |
| c69 |    21 |
+-----+-------+
2 rows in set (0.00 sec)

mysql> select*from member where ID='c69' or ID='c68';
+-----+-------+
| ID  | score |
+-----+-------+
| c68 |    18 |
| c69 |    24 |
+-----+-------+
2 rows in set (0.00 sec)
```

2.cmt变化

```
mysql> select*from cmt;
+-------+---------+----------+------+
| od_id | comment | datetime | rank |
+-------+---------+----------+------+
| oc11  | good    |     NULL |    5 |
+-------+---------+----------+------+
1 row in set (0.01 sec)
```

Bonus

对每种完整性约束，给出对应的trigger代码截图

1.购买数量和库存量的约束

```
USE `m3DBS`$$
CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`ord_c_AFTER_INSERT_1` AFTER INSERT ON `ord_c` FOR EACH ROW
BEGIN

if (new.number >
(select remain from commodity
 where c_id=new.c_id))
 then
 SIGNAL SQLSTATE '45000'
    SET message_text = 'Do not have that amount of goods! ';
end if;
```

2.库存量减少机制

```
if(new.number <=
(select remain from commodity
 where c_id=new.c_id))
 then
   update commodity
   set remain = remain-new.number
   where c_id = new.c_id;

   update ord
   set state = '1'
   where od_id = new.od_id;
```

3. 库存量在订单失效后恢复机制

```
USE `m3DBS`$$
CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`back_od_AFTER_INSERT` AFTER INSERT ON `back_od` FOR EACH ROW
BEGIN

  declare num int;
  declare  cid char(16);
  declare done int default 0;
  declare cur_1 cursor for select c_id,number from ord_c where od_id=new.od_id ;

  DECLARE CONTINUE HANDLER FOR NOT FOUND set done=1;


  open cur_1;

read_loop_1: loop

      fetch cur_1 into cid, num;

      if done=1 then
         leave read_loop_1;
      end if;

    update commodity
    set remain=remain+num
    where c_id = cid;


end loop;


  update ord
  set state='4'
  where od_id=new.od_id;
  END$$
```

4.订单评价后的积分增加机制

```
USE `m3DBS`$$
CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`ord_AFTER_UPDATE` AFTER UPDATE ON `ord` FOR EACH ROW
BEGIN

if new.state='3'
then
update member
set score=score+3 where ID =
(select ID from purchase where od_id=new.od_id);
end if;

END$$
```

5. 会员积分与等级、折扣率的对应机制

- **update机制**

```
DELIMITER $$

USE `m3DBS`$$
CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`member_AFTER_UPDATE` AFTER UPDATE ON `member` FOR EACH ROW
BEGIN

if (new.score < 101)
then
update cDiscount
set disCnt = '0.9'
where ID = new.ID;

update rank
set rank = '1'
where ID = new.ID;

end if;



if (new.score >= 101 and new.score <= 500)
then
update cDiscount
set disCnt = '0.8'
where ID = new.ID;

update rank
set rank = '2'
where ID = new.ID;

end if;



if(new.score >= 501)
then
update cDiscount
set disCnt = '0.7'
where ID = new.ID;

update rank
set rank = '3'
where ID = new.ID;

end if;


END$$

DELIMITER ;
```

- **insert机制**

  会员insert机制

```
USE `m3DBS`$$
CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`account_AFTER_INSERT` AFTER INSERT ON `account` FOR EACH ROW
BEGIN
if(new.is_member = 1)
then
insert into member VALUES(new.ID, '0');

end if;
END$$
```

等级，折扣更新机制

```
353
354 •  USE `m3DBS`$$
355 •  CREATE DEFINER = CURRENT_USER TRIGGER `m3DBS`.`member_AFTER_INSERT` AFTER INSERT ON `member` FOR EACH ROW
356    BEGIN
357    if (new.score between 0 and 100)
358    then
359    insert into cDiscount
360    VALUES(new.ID, '0.9');
361
362    insert into rank
363    VALUES(new.ID, '1');
364
365    end if;
366
367    if (new.score between 101 and 500)
368    then
369    insert into cDiscount
370    VALUES(new.ID, '0.8');
371
372    insert into rank
373    VALUES(new.ID, '2');
374    end if;
375
376    if (new.score >  501)
377    then
378    insert into cDiscount
379    VALUES(new.ID, '0.7');
380
381    insert into rank
382    VALUES(new.ID, '3');
383    end if;
384
385
386    END$$
387
```

6. 其他各种trigger/procedure/插入数据代码 在附件中(左：文件名 右: 所含操作内容)

trigger_member_update	**对应会员在积分更新后 -> 更新等级和折扣率**
trigger_phone_num	**对应客户的电话号码在插入的时候不能超出8位整数的范畴的限制+客户不能大于80岁的限制**
trigger_phone_num_update	**对应客户电话号码在更新后不能超出8位整数的范畴的限制+客户不能大于80岁的限制**
trigger_Amt_alert_good_r-money_i-back_od_score_i	**(1). 订单的货物最大数量限制+订单的金额计算机制+订单的货物减少机制. (2). 订单的退货处理机制 (3). 会员的积分增加机制**
trigger_mail_addr	**对应订单的邮编的6位控制机制**
procedure_insert_ord_c	**作业4customer1和customer2下单插入代码**
procedure_insert_store	**作业4插入商店以及商店的4类商品各4种的代码**
procedure_insert_customer	**作业4插入100名客户的代码**
Hw4-2-4	**作业4顾客下单->退货->订单完成完成代码**
hw3构建脚本(trigger)	**(1).插入客户年龄大于80限制 (2). 插入客户折扣大于0小于1限制 3. 插入商品折扣率大于0小于1限制 (3). 插入评分小于5限制 (4). 会员-积分-等级-折扣升级机制**

Survey（不算分、选做）

关于上机课的这种形式（上机作业的次数、难度和时间安排等方面），大家是否有收获，欢迎同学们对不足的地方给出批评和改进的建议，感谢大家的配合！

收获可以说很大了～给助教打call...*(:3 」)*

如果不是上机我可能都不会用innoDB，不会深入了解各种隔离级的意义和应用场景，不会用workbench和dataGrip，没有动力自己设计一个数据库，不知道现实数据库中3NF和1NF比好处和坏处在哪里，h不知道数据库居然还可以lock，不会用JSDB，不会查资料用各种trigger,procedure和cursor...这些都是理论的课堂不能带来的好处。而且bonus的难度设计得刚刚好...不会太难也没有很简单，属于能做而且很好玩的范围～点赞...

唯一的建议是希望助教把查资料的工作量考虑进去吧*(:3 」)*，因为第二次作业做得蛮痛苦的.....（造福下一届—w—）

最后很感谢助教（超级即时）（不厌其烦）地回复我们（其实很蠢）的问题.....还有超级仔细地改我们的作业～*(:3 」)*再次打call...