数据库综合设计实验报告

131220128　杨帆

一、实验环境

操作系统：windows7

软件版本：MySQL Server 5.7

MySQL　Workbench 6.3CE

二、实验过程：

代码分为几个部分：

1. 建表

```
create database Project_js character set gbk;
use Project_js;

create table course_info
(
      cname varchar(20) not null,
       cno smallint unique primary key
);
#该表表示定义的课程名和课程号的对应关系，起到记录作用，没有实际必要
#We set 1.yuwen 2.maths 3.english 4.physics 5.chemistry 6.history 7.geography
       8.politics 9.biology 10.computer science


create table course_combine
(
       c_choice char(2) not null primary key,
       elect1 smallint not null,
       elect2 smallint not null,
       other1 smallint not null,
       other2 smallint not null,
```

```
    other3 smallint not null,
    other4 smallint not null
);
```
该表定义了选修代码号与所选修课程、必修课程的对应关系
如（“45”对应选修为 4 物理 5 化学、必修为 6 历史 7 地理 8 政治 9 生物）

```
create table senior_school
(
   school_no char(6) unique not null primary key,
    school_name varchar(30) not null,
    tel_no varchar(11) unique not null
);
```
该表定义了中学的编号、名称、电话的对应关系，其中每个学中学编号是唯一的，设为主键

```
create table student
(
    sname varchar(30) not null,              #姓名
    gender char(1) not null,                 #性别
    birthdate date not null,                 #生日
    idno char(18) not null,                  #省份证号码
    ethnic varchar(10) not null,             #名族
    senior_school_no char(6),                #中学学校编号
    stu_type varchar(10) not null,           #学生类型
    stu_academic varchar(7) not null comment 'Art/Science',      #文科/理科
    course_choice char(2) not null,          #选课代码
    phone_no varchar(11) not null,           #电话
    deliver_address varchar(100) not null,       #收件地址
    post_no char(6) not null,                    #邮编
    post_receiver varchar(30) not null,          #收件人
    sno char(12) not null primary key,           #考籍号
    KSH char(14) not null,                       #考生号
    ZKH char(13) not null,                       #准考证号
    character_score smallint,                    #特征分
    plunge_score smallint,                       #投档分
   elect1_grade varchar(2),                      #选修 1 的等第
    elect2_grade varchar(2),                     #选修 2 的等第
    rank_in_province int,                        #投档分省内排名
    foreign key(senior_school_no) references senior_school(school_no),
    foreign key(course_choice) references course_combine(c_choice)
);
```
该表定义了学生信息，其中选课代码和中学编码都设为外键。
此表必须定义在选修对应表和中学表之后，否则会报错。

```
create table stu_grade
(
        sno char(12) not null,                          #考籍号
       score_yw smallint not null,                      #语文成绩
        score_sx smallint not null,                     #数学成绩
        score_eng smallint not null,                    #英语成绩
         addition smallint not null,                    #附加题成绩
         elect1_score smallint,                         #选课 1 分数成绩
         elect1_grade varchar(2),                       #选课 1 等第
         elect2_score smallint,                         #选课 2 分数成绩
         elect2_grade varchar(2),                       #选课 2 等第
         other1_score smallint not null,                #必修 1 分数成绩
         other1_grade char(1),                          #必修 2 等第    （后面类似）
         other2_score smallint not null,
         other2_grade char(1),
         other3_score smallint not null,
         other3_grade char(1),
         other4_score smallint not null,
         other4_grade char(1),
         cs_score smallint not null,                        #信息技术分数成绩
         cs_grade char(6),                          #信息技术是否合格设为“Pass/Failed”
         req_score smallint,                        #必修课程总分 0/1/2/3/5
         foreign key(sno) references student(sno)
);
```

本表记录了学生及其所有成绩的对应关系

查询一个唯一的考籍号就可查询到其所有成绩

本来想把学生、课程建成一张表查分的，后来觉得太过麻烦，就将所有成绩放在了一张表中

2. 创建两个学生信息视图（分别面向文/理科考生）

信息技术科目合格，且六门选测/必测科目没有 D 的考生

视图中的属性包括：学籍号，考生号，准考证号，姓名，投档分，特征分，选测科目一的等级（理科是‘物理’，文科是‘历史’），选测科目二的等级

```
create view Pass_Student_for_Art as
(
    select student.sno, KSH, ZKH, sname, character_score, plunge_score, student.elect1_grade, student.elect2_grade from student, stu_grade
     where stu_academic = '文科'
          and stu_grade.sno = student.sno
            and 60 <= stu_grade.cs_score
            and 'D'<> stu_grade.other1_grade
            and 'D'<> stu_grade.other2_grade
          and 'D'<> stu_grade.other3_grade
            and 'D'<> stu_grade.other4_grade
);
```

建立文科合格学生视图

```
#view of qualified science student
create view Pass_Student_for_Science as
(
    select student.sno, KSH, ZKH, sname, character_score, plunge_score, student.elect1_grade, student.elect2_grade from student, stu_grade
     where stu_academic = '理科'
          and stu_grade.sno = student.sno
            and 60 <= stu_grade.cs_score
            and 'D'<> stu_grade.other1_grade
            and 'D'<> stu_grade.other2_grade
          and 'D'<> stu_grade.other3_grade
            and 'D'<> stu_grade.other4_grade
);
```

建立理科合格学生视图

3. 创建触发器：必测科目（小考高）的等级设置，以及一些错误的判断

```
#check students' grades
create trigger Check_Grade
before insert on stu_grade
for each row
begin
if new.score_yw > 160 or new.score_yw < 0 then set new.score_yw = null; end if;
if new.score_sx > 160 or new.score_sx < 0 then set new.score_sx = null; end if;
if new.score_eng > 120 or new.score_eng < 0 then set new.score_eng = null; end if;
if new.addition > 40 or new.addition < 0 then set new.addition = null; end if;
if new.elect1_score != null and new.elect1_score > 120 or new.elect1_score < 0 then set new.elect1_score = null; end if;
if new.elect2_score != null and new.elect2_score > 120 or new.elect2_score < 0 then set new.elect2_score = null; end if;
if new.other1_score > 100 or new.other1_score < 0 then set new.other1_score = null; end if;
if new.other2_score > 100 or new.other2_score < 0 then set new.other2_score = null; end if;
if new.other3_score > 100 or new.other3_score < 0 then set new.other3_score = null; end if;
if new.other4_score > 100 or new.other4_score < 0 then set new.other1_score = null; end if;
if new.cs_score > 100 or new.cs_score < 0 then set new.cs_score = null; end if;
end;
//
#利用触发器判断 insert 的分数数据是否有越界的错误情况，例如 0<语文数学<160,0<英语<120,等等 若输入数据越界，则将新插入的条目对应位置设置为 null,由于表中值为 not null,此时便会报错，提示输入数据有误。

#Calculate 4 required courses and computer science's score & grade
create trigger Cal_Req_Course
before insert on stu_grade
for each row
    begin
        declare cnt int;
        set cnt = 0;
        if new.other1_score >= 90 then set cnt = cnt + 1; end if;
        if new.other1_score >= 90 then set new.other1_grade = 'A'; end if;
        if new.other1_score <= 89 and new.other1_score >= 75 then set new.other1_grade = 'B'; end if;
    if new.other1_score <= 74 and new.other1_score >= 60 then set
```

```
new.other1_grade = 'C'; end if;
       if new.other1_score <= 59 then set new.other1_grade = 'D'; end if;
       if new.other2_score >= 90 then set cnt = cnt + 1; end if;
       if new.other2_score >= 90 then set new.other2_grade = 'A'; end if;
      if new.other2_score <= 89 and new.other2_score >= 75 then set new.other2_grade
= 'B'; end if;
     if new.other2_score <= 74 and new.other2_score >= 60 then set new.other2_grade
= 'C'; end if;
     if new.other2_score <= 59 then set new.other2_grade = 'D'; end if;
     if new.other3_score >= 90 then set cnt = cnt + 1; end if;
     if new.other3_score >= 90 then set new.other3_grade = 'A'; end if;
      if new.other3_score <= 89 and new.other3_score >= 75 then set new.other3_grade
= 'B'; end if;
     if new.other3_score <= 74 and new.other3_score >= 60 then set new.other3_grade
= 'C'; end if;
     if new.other3_score <= 59 then set new.other3_grade = 'D'; end if;
     if new.other4_score >= 90 then set cnt = cnt + 1; end if;
     if new.other4_score >= 90 then set new.other4_grade = 'A'; end if;
      if new.other4_score <= 89 and new.other4_score >= 75 then set new.other4_grade
= 'B'; end if;
     if new.other4_score <= 74 and new.other4_score >= 60 then set new.other4_grade
= 'C'; end if;
     if new.other4_score <= 59 then set new.other4_grade = 'D'; end if;
     if new.cs_score <= 59 then set new.cs_grade = 'Failed'; else set new.cs_grade =
'Pass';    end if;
     if cnt <= 3 then set new.req_score = cnt; else set new.req_score = 5;
                end if;
           end;
           //
    #利用触发器生成必修 4 门课的等级、必修课总分（1~5）信息技术合格与否
的信息

create trigger Check_Stu_Info
      before insert on student
       for each row
           begin
                declare tmp1 varchar(5);
                  declare tmp2 varchar(5);
                  declare tmp3 varchar(5);
                  declare tmp4 varchar(5);

                  set tmp1 = substring(new.course_choice, 1, 1);                    #?
               if new.stu_academic = '文科' and tmp1 = '4' then set new.stu_academic =
null; set new.course_choice = null; end if;
```

```
                if new.stu_academic = '理科' and tmp1 = '6' then set new.stu_academic
= null; set new.course_choice = null; end if;
                 set tmp1 = substring(new.sno, 3, 4);
                set tmp2 = substring(new.KSH, 5, 4);
                set tmp3 = substring(new.ZKH, 3, 4);
                set tmp4 = substring(new.senior_school_no, 1, 4);
                if tmp1 != tmp2 or tmp1 != tmp3 or tmp1 != tmp4 then
                   set new.sno = null; set new.KSH = null; set new.ZKH = null; set
new.senior_school_no = null; end if;

                set tmp1 = substring(new.KSH, 3, 2);
                if tmp1 != '32' then set new.KSH = null; end if;
                set tmp1 = new.course_choice;
                set tmp2 = substring(new.KSH, 9, 2);
                set tmp3 = substring(new.ZKH, 7, 2);
                if tmp1 != tmp2 or tmp1 != tmp3 then
                   set new.course_choice = null; set new.KSH = null; set new.ZKH =
null; end if;

                set tmp1 = substring(new.sno, 9, 4);
                set tmp2 = substring(new.KSH, 11, 4);
                if tmp1 != tmp2 then set new.sno = null; set new.KSH = null; end if;

                set tmp1 = substring(new.sno, 1, 2);
                set tmp2 = substring(new.KSH, 1, 2);
                set tmp3 = substring(new.ZKH, 1, 2);
                if tmp1 != tmp2 or tmp1 != tmp3 then
                   set new.sno = null; set new.KSH = null; set new.ZKH = null;    end
if;

                set tmp1 = substring(new.sno, 7, 2);
                set tmp2 = substring(new.senior_school_no, 5, 2);
                if tmp1 != tmp2 then set new.sno = null; set new.senior_school_no =
null; end if;
                set tmp1 = substring(new.KSH, 3, 2);
                if tmp1 != '32' then set new.KSH = null; end if;
end;
```

#利用触发器实现一些基本的信息的查错，比如考籍号、学生号、准考证号的前两位都为所在年份的末两位，必须相等。若发现这样的错误，则将一些值设置为null。在定义表时，由于值都为 not null,故会报错。

其中的字符串处理用到了 substring(char* st, int a, int l)函数
即从字符串第 a 位截取 l 位。

4. 创建存储过程：对所有选修课程等第的计算（要用到单科省内排名的百分比）

首先定义一些局部变量记录信息。建立游标，有成绩由高到低取出学生与其成绩信息，逐条生成其名次与名次百分比。
这里 6 门可选的选修课生成方法类似，这里只贴出生成物理等第、名次的代码部分。

```
create procedure Cal_Grade_for_Physics()
    begin
        declare score smallint;    #temp score
          declare sno1 char(12);
          declare no_more_record smallint default 0;
          declare num int default 0;                    # #students
          declare cnt int default 1;              # #students who get pre_score
          declare rank int default 0;             #temp rank
          declare s_pre smallint default 121;           #previous score
          declare r_pre int default 0;            #previous rank
          declare p float;#percentage
          declare cursor_phy cursor for select student.sno, stu_grade.elect1_score from student, stu_grade where student.sno = stu_grade.sno and stu_academic = '理科' order by stu_grade.elect1_score desc;
        declare continue handler for not found set no_more_record = 1;
          select count(student.sno) from student where student.stu_academic = '理科' into num;
          open cursor_phy;
          fetch cursor_phy into sno1, score;
          select sno1,score;
          while no_more_record != 1 do
            if score = s_pre then set rank = r_pre; set cnt = cnt + 1; end if;
              if score < s_pre then
                set rank = r_pre + cnt;
                  set cnt = 1;
                  set r_pre = rank;
                  set s_pre = score;
            end if;
              set p = rank/num;
              if p <= 0.05 then
                update student set elect1_grade = 'A+' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'A+' where sno1 = stu_grade.sno; end if;
```

```
            if p > 0.05 and p <= 0.2 then
                update student set elect1_grade = 'A' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'A' where sno1 =
stu_grade.sno; end if;
            if p > 0.2 and p <= 0.3 then
                update student set elect1_grade = 'B+' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'B+' where sno1 =
stu_grade.sno; end if;
            if p > 0.3 and p <= 0.5 then
                update student set elect1_grade = 'B' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'B' where sno1 =
stu_grade.sno; end if;
            if p > 0.5 and p <= 0.9 then
                update student set elect1_grade = 'C' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'C' where sno1 =
stu_grade.sno; end if;
            if p > 0.9 then
                update student set elect1_grade = 'D' where sno1 = student.sno;
                  update stu_grade set elect1_grade = 'D' where sno1 =
stu_grade.sno; end if;
            fetch cursor_phy into sno1, score;
        end while;
        close cursor_phy;
    end;
```

5. 创建存储过程：分别计算文理科的省内排名（只贴出求理科排名的代码）

```
#Science students' ranking
create procedure Rank_in_Province_for_Sci()
    begin
        declare cnt smallint default 1;
        declare s_pre smallint default 500;
        declare r_pre smallint default 0;
        declare temp smallint;
        declare sno1 char(12);
        declare no_more_record smallint default 0;
        declare cursor_rank_province_sci cursor for select sno, plunge_score from
student where stu_academic = '理科' order by plunge_score desc;
        declare continue handler for not found set no_more_record = 1;
        open cursor_rank_province_sci;
        fetch cursor_rank_province_sci into sno1, temp;
        while no_more_record != 1 do
```

```
            if temp = s_pre then
                update student set rank_in_province = r_pre where sno1 =
student.sno;
                  set cnt = cnt + 1;
            else
                update student set rank_in_province = r_pre + cnt where sno1 =
student.sno;
                  set r_pre = r_pre + cnt;
                  set s_pre = temp;
                  set cnt = 1;
            end if;
              fetch cursor_rank_province_sci into sno1, temp;
        end while;
        close cursor_rank_province_sci;
         end;
        //
```

#这里同样用到了游标，取出理科学生的信息，由于事先生成了每个学生的特征分和投档分，将投档分由大到小处理即可。处理方法与计算选修课等第、名次时类似。

6. 创建存储过程：计算每个学生的特征分、投档分。

```
#Calculate character & plunge score
create procedure Cal_CP()
    begin
        declare no_more_record int default 0;
        declare sno1 char(12);
         declare yw smallint;
         declare sx smallint;
         declare eng smallint;
         declare addition1 smallint;
         declare req_4 smallint;
         declare cursor_sum cursor for select sno, score_yw, score_sx, score_eng,
addition, req_score from stu_grade;
        declare continue handler for not found set no_more_record = 1;
        open cursor_sum;
         fetch cursor_sum into sno1, yw, sx, eng, addition1, req_4;
         while no_more_record != 1 do
              update student set character_score = yw + sx + addition1
                                  where student.sno = sno1;
               update student set plunge_score = yw + sx + eng + req_4 + addition1
                                  where student.sno = sno1;
            fetch cursor_sum into sno1, yw, sx, eng, addition1, req_4;          #?
```

```
        end while;
         close cursor_sum;
    end;
    //    #利用游标取出每个学生的语文、数学、英语、附加分的成绩，into 到局部变量中，经过计算 update 到学生信息、成绩表中即可。
```

7. 插入一些必须信息，并插入两位学生信息及他们的成绩。（此处均为合法信息，若不合法，则程序会报错，无法建立数据库）

```
#插入课程信息，这里起到了记录信息的作用，并无实际意义。
insert into course_info value("语文", 1);
insert into course_info value("数学", 2);
insert into course_info value("英语", 3);
insert into course_info value("物理", 4);
insert into course_info value("化学", 5);
insert into course_info value("历史", 6);
insert into course_info value("地理", 7);
insert into course_info value("政治", 8);
insert into course_info value("生物", 9);
insert into course_info value("信息技术", 10);
insert into course_combine value("45", 4, 5, 6, 7, 8, 9);
insert into course_combine value("47", 4, 7, 5, 6, 8, 9);
insert into course_combine value("48", 4, 8, 5, 6, 7, 9);
insert into course_combine value("49", 4, 9, 5, 6, 7, 8);
insert into course_combine value("65", 6, 5, 4, 7, 8, 9);
insert into course_combine value("67", 6, 7, 4, 5, 8, 9);
insert into course_combine value("68", 6, 8, 4, 5, 7, 9);
insert into course_combine value("69", 6, 9, 4, 5, 7, 8);
#插入学校信息
insert into senior_school value(100101, "S_A", 12306);
insert into senior_school value(100102, "S_B", 55555);
insert into senior_school value(100103, "S_C", 45678);
#插入学生信息、成绩
insert into student value("叶尾鱼", "M", "1995-7-28", "320103199507282517", "汉",
"100102", "应届", "理科", "45", "18217417434", "南通市", "226001", "叶尾鱼",
"151001023457", "15321001453457", "1510014534502", null, null, null, null, null);
insert into stu_grade value("151001023457", 140, 150, 88, 35, 110, null, 90, null, 90,
null, 90, null, 90, null, 90, null, 90, null, null);
insert into student value("杨小帆", "M", "1994-11-3", "320103199411032518", "汉",
"100101", "应届", "理科", "45", "18001591168", "南京市", "210007", "杨小帆",
"151001013456", "15321001453456", "1510014534501", null, null, null, null, null);
```

```
insert into stu_grade value("151001013456", 109, 131, 103, 40, 105, null, 100, null, 88, null, 95, null, 95, null, 95, null, 100, null, null);
#计算选修课等第
call Cal_Grade_for_Physics();
call Cal_Grade_for_History();
call Cal_Grade_for_Chemistry();
call Cal_Grade_for_Geography();
call Cal_Grade_for_Politics();
call Cal_Grade_for_Biology();
#计算特征分投档分
call Cal_CP();
#计算文理科省内排名
call Rank_in_Province_for_Art();
call Rank_in_Province_for_Sci();

#因为 course_combine、senior_school_no 都是 student 表中的 foreign key，所以在插入学生信息之前，必须将选课号表、学校信息填充且包含学生所对应的信息。
#插入学生成绩时，学生的信息必须填充，由于 sno 是 stu_grade 中的 foreign key，理由同上。
```

三、实验结果：

1.由于数据较为繁琐，只手动插入了两条学生信息及其成绩：

学生信息：

sname	gender	birthdate	idno	ethnic	senior_school_no	stu_type	stu_academic	course_choice	phone_no	deliver_addre
杨小帆	M	1994-11-03	320103199411032518	汉	100101	应届	理科	45	18001591168	南京市
叶尾鱼	M	1995-07-28	320103199507282517	汉	100102	应届	理科	45	18217417434	南通市

post_no	post_receiver	sno	KSH	ZKH	character_score	plunge_score	elect1_grade	elect2_grade	rank_in_prov
210007	杨小帆	151001013456	15321001453456	1510014534501	280	386	D	B	2
226001	叶尾鱼	151001023457	15321001453457	1510014534502	325	418	B	D	1

成绩：

sno	score_yw	score_sx	score_eng	addition	elect1_score	elect1_grade	elect2_score	elect2_grade
151001023457	140	150	88	35	110	B	90	D
151001013456	109	131	103	40	105	D	100	B

other1_score	other1_grade	other2_score	other2_grade	other3_score	other3_grade	other4_score	other4_grade	cs_score	cs_grade	req_score
90	A	90	A	90	A	90	A	90	Pass	5
88	B	95	A	95	A	95	A	100	Pass	3

2.视图：

sno	KSH	ZKH	sname	character_score	plunge_score	elect1_grade	elect2_grade
151001023457	15321001453457	1510014534502	叶尾鱼	325	418	B	D
151001013456	15321001453456	1510014534501	杨小帆	280	386	D	B

四、实验总结收获和遇到的困难：

1.学习到了一些 MySQL 的基本操作方法、调试方法，和一些具体的语法。

2.若表 A 有 foreign key K 是 B 中的主键，则表 B 必须定义在表 A 前。
在 insert 具体信息时，必须先把表 B 中主键 K 的信息填完再填写表 A，且表 A 对应信息必须在 B 中出现过。

3.学习了游标 cursor 的具体使用方法

```
定义游标
declare cursor_name cursor for select ... from ...;
定义循环结束标志
declare no_more_record int default 0;
declare continue handler for not found set no_more_record = 1;
打开游标
open cursor_name
取游标
fetch cursor_name into  局部变量
游标循环
while no_more_record != 1 do
    ...
    fetch cursor_name into  局部变量
end while;
释放游标
close cursor_name;
```

4.我在处理错误信息的过程中，将错误地点值设置为 null,比如学生语文成绩本来是 130 分，被错误登为 161 分，则将其语文成绩设置为 null。学生学号对应为与中学信息不符，则学生学号设置为 null，等等。不知道这样的处理方式是否得当、可取。

5.学习到了一些调试的技巧，比如在程序运行处插入 select 语句，第一次执行到该处时会自动停止，并显示出 select 出的变量实时值。

6.在写触发器和存储过程的时候，不论怎样修改代码，程序都一直在报错，后来在网络中搜索发现，一般情况下，MySQL 默认是以“;”作为结束执行语句，但在创建触发器过程中需要用到“;”，所以程序需一直会将;解析为结束执行语句，所以无法正确编译。

解决方案：为了解决这个问题，可以用 DELIMITER 语句。如“DELIMITER //”，可以将结束符号变成“//”。当触发器、过程创建完成后，可以用命令“DELIMITER ;”来将结束符号变成“;”。

7.程序中可能仍有一些 bug 没有被发现，由于测试数据还不够多，还未发现。请老师提出指正！