

**学校教务系统的设计与实现**

|  |  |
| --- | --- |
| **二级学院** | **医药信息工程学院** |
| **专 业** | **计算机科学与技术** |
| **班 级** | **201** |
| **学生姓名** | **李文智、张铁锋、张泽锴** |
| **学 号** | **27、60、61** |
| **组 号** | **8** |
| **指导教师** | **李闵** |
|  |  |
| 2022 年 12 月 | |

目录

[1、需求分析 1](#_Toc122312744)

[1.1设计背景 1](#_Toc122312745)

[1.1.1系统开发背景 1](#_Toc122312746)

[1.1.2系统分析 1](#_Toc122312747)

[1.1.3用户需求分析 2](#_Toc122312748)

[2、概念结构设计 3](#_Toc122312749)

[2.1 实体的设计 3](#_Toc122312750)

[2.2 联系的设计 3](#_Toc122312751)

[2.3 概念模型（ER图） 4](#_Toc122312752)

[3、数据库物理设计 4](#_Toc122312753)

[3.1 表的设计 4](#_Toc122312754)

[3.1.1 Student(学生)表： 5](#_Toc122312755)

[3.1.2 Class（班级）表 5](#_Toc122312756)

[3.1.3 Course（课程）表 6](#_Toc122312757)

[3.1.4 Teacher（教师）表 6](#_Toc122312758)

[3.1.5 Deparment（系别）表 6](#_Toc122312759)

[3.1.6 User（用户）表 7](#_Toc122312760)

[3.2 索引的设计 7](#_Toc122312761)

[3.2.1索引表 7](#_Toc122312762)

[3.2.2索引目录结构 7](#_Toc122312763)

[3.2.3索引生成代码 8](#_Toc122312764)

[3.3 视图的设计 9](#_Toc122312765)

[3.3.1视图表 9](#_Toc122312766)

[3.3.2视图目录结构 10](#_Toc122312767)

[3.3.3视图生成代码 10](#_Toc122312768)

[3.4 触发器的设计 12](#_Toc122312769)

[3.4.1触发器表 12](#_Toc122312770)

[3.4.2触发器目录结构 13](#_Toc122312771)

[3.4.3触发器生成代码 13](#_Toc122312772)

[4、数据库实施 15](#_Toc122312773)

[4.1 代码的生成 15](#_Toc122312774)

[4.2 相关内容的改进 16](#_Toc122312775)

[4.3 数据测试 16](#_Toc122312776)

[4.3.1视图的测试 16](#_Toc122312777)

[4.3.2约束的测试 20](#_Toc122312778)

[4.3.3触发器测试 20](#_Toc122312779)

[附录 24](#_Toc122312780)

# 1、需求分析

## 1.1设计背景

### 1.1.1系统开发背景

伴随着席卷全球的信息化、数字化浪潮，我国已经步入了信息社会的大门，计算机的应用在我国信息化的过程中起到了至关重要的作用。在飞速发展的信息时代，各行各业中离不开信息处理，计算机被广泛应用于信息管理系统中。使用计算机进行信息控制，不仅提高了工作效率，而且大大的提高了其安全性。尤其对于复杂的信息管理，计算机能够充分发挥它的优越性。

中国教育发展迅速，随着高校的扩招，学校教学管理的任务量也在不断加大，教务人员负担很重。如何才能提高办学效率，更好地完成教学任务，跟上社会发展步伐，这是一个摆在教学工作者面前的一个迫切的问题。应用信息化来改造传统的教学管理模式是一个重要途径，为此开发了教务管理系统，以实现学校教务管理的自动化。

### 1.1.2系统分析

教务管理系统的目的是对学校的教学任务进行管理。建立该管理系统的主要目标是为该校提供全面的教学任务管理的解决方案，以解决传统的人工管理方式效率低，保密性差，不利于查找，更新和维护的问题，极大地提高学校教务管理的效率，节约教育经费，以适应学校信息化建设的发展趋势。

教务管理系统分为以下几个部分：学生管理、课程管理、班级管理、教师管理、系别管理等。

其中学生管理包括学号、姓名、性别、出生日期、手机号、住址、班级号；班级管理包括班级号、系别号、班主任的工号、班长的学号、班级名称；课程管理包括课程号、课程名称、课时、简介；教师管理包括工号、姓名、性别、出生日期、职位、工作系别、最低薪水；系别管理包括系别号、系别名称、联系方式。

系统的数据输入信息包括系别、教师、班级、课程、学生、成绩。系统的数据输出信息包括计算机汇总输出信息，如各类查询结果、各类视图等。

本系统主要能实现以下功能：

1、能够用户的所有信息。

2、能够对教师、学生、班级、系别、课程的信息注册登记管理。

3、 能够通过学生的某一项信息找到其班级、课程、成绩等内容的相关信息。

4、能够通过教师的某一项信息找到其任课课程、任课班级等内容的相关信息。

5、能够通过系别的某一项信息找到其系别号、系名、联系方式等内容的相关信息。

6、能够通过班级的某一项信息找到其班主任、班长、所属系别等内容的相关信息。

7、能够通过课程的某一项信息找到其任课老师、课程名、课时等内容的相关信息。

8、管理人员能对教师、学生的全部信息进行管理。

9、能对不同用户（管理员、教师、学生）设置管理权限。

10、能够进行学生成绩添加、删除和修改。

### 1.1.3用户需求分析

#### 1.1 用户类型

1、管理员类型：管理员可以成功登陆系统，系统管理员有权限进行如下操作：

（1）增加或删除删除用户（2）修改已有用户信息（3）增加或删除已有实体的信息（4）修改已有实体的信息等

2、教师类型：教师可以成功登录系统，教师有权限进行如下操作：

（1）录入学生课程成绩（2）查询学生的成绩（3）修改个人密码（4）查看自己的薪水情况等

3、学生类型：学生可以成功登录系统，学生有权限进行如下操作：

（1）查询成绩（2）修改个人密码（3）修改个人基本信息。

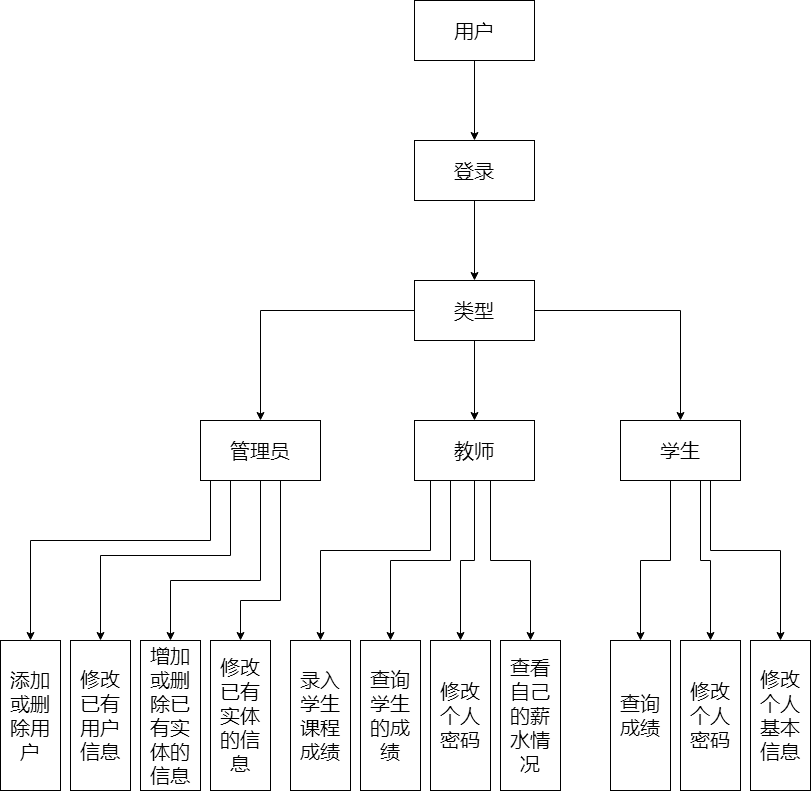


图1-1 整体设计流程

#### 1.2 数据流图

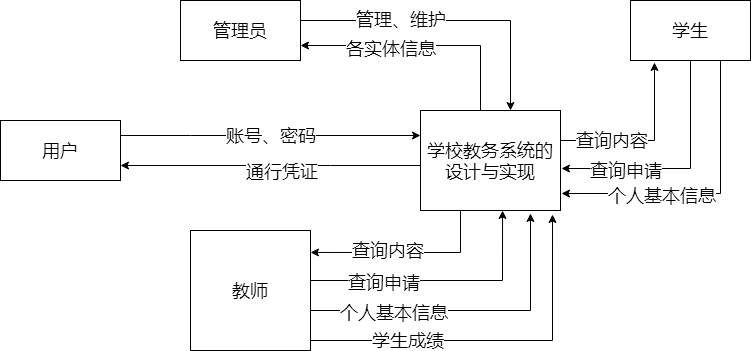


图1-2 数据流程顶层图

# 2、概念结构设计

## 2.1 实体的设计

1、系别实体，其属性有系编号、系名称、电话号码，系别人数。

2、班级实体，其属性有班级编号、班级名称、班主任、班长、所属系别编号，班级人数。

3、教师实体，其属性有教师编号、姓名、性别、出生日期、职称、所属系别编号，最低薪水，联系方式。

4、学生实体，其属性有学号、姓名、性别、出生日期、联系电话、家庭住址、所在班级，平均分。

5、课程实体，其属性有课程编号、课程名称、课时数、课程介绍。

6、学生成绩资料实体，其属性有学生编号、课程编号、成绩。

7、课程教师资料实体，其属性有课程编号、班级编号、任课老师编号。

8、用户实体，其属性有账号、密码、权限信息

## 2.2 联系的设计

一对一联系：

1. 班级与教师之间的管理联系：一个班级有且只有一个班主任。
2. 班级与学生之间的管理联系：一个班级有且只有一个班长。

一对多联系：

1、系别和班级之间的属于联系：一个系别有多个班级，一个班级只能属于一个系别。

2、班级和学生之间的属于联系：一个班级有多个学生，一个学生只能属于一个班级。

3、系别和教师之间的属于联系：一个系别有多个老师，一个老师只能属于一个系别。

2.3 概念模型（ER图）

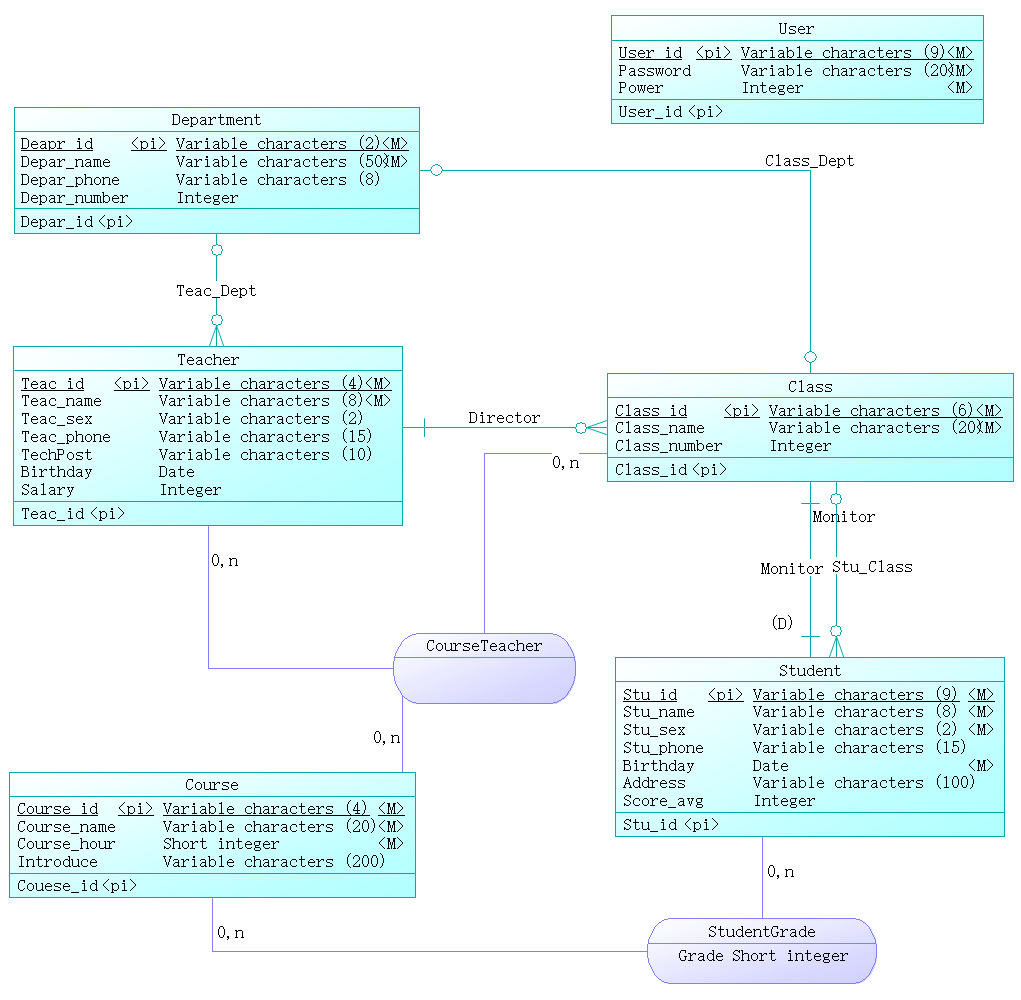


图2-1 学校教务管理系统E-R图

图四 是根据需求分析和数据流图设计出的E-R图，该概念模型有8个实体.6个视图。

# 3、数据库物理设计

3.1 表的设计

如下是所设计表的各个属性

表3-1 实体属性一览表

|  |  |  |
| --- | --- | --- |
| 序号 | 表格 | 属性 |
|  | Student(学生) | Stu\_id,Stu\_name,Stu\_sex,Birthday,Phone,Address,Class\_id,  Score\_avg |
|  | Teacher(教师) | Teac\_id,Teac\_name,Teac\_Sex,Birthday,TechPost,Depar\_id,  Teac\_phone,Salary |
|  | Deparment(系别) | Depar\_id,Depar\_name,Depar\_phone,Depar\_number |
|  | Class(班级) | Class\_id,Director,Depar\_id,Monitor,Class\_name,Class\_number |
|  | Course(课程) | Course\_id,Course\_name,Course\_hour,Intorduce |
|  | User(用户) | User\_id,Password,Power |

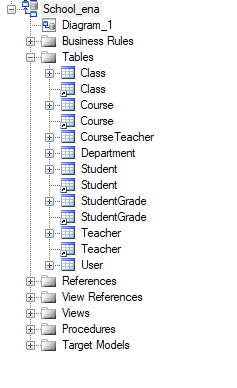
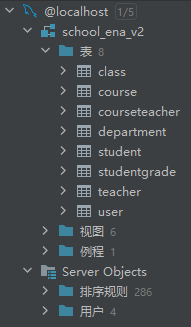


图3-1 MySQL、PowerDesigner中的表目录结构

### 3.1.1 Student(学生)表：

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，学生表中没有出现重复项，满足第一范式.

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，学号是主码，非主属性都完全依赖于学号，故满足第二范式。

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该学生表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、类完整性规则：

实体完整性：主码学号不为空。

参照完整性：学生表有外键班级编号，且班级编号不为空。

用户自定义完整性：设计属性性别为默认值为“男”，姓名、生日属性不能为空。

### 3.1.2 Class（班级）表

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，班级表中没有出现重复项，满足第一范式；

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，班级编号是主码，非主属性都完全依赖于班级编号，故满足第二范式；

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该班级表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、满足三类完整性规则：

实体完整性：主码班级编号不为空。

参照完整性：有外键系编号，且系编号不为空。

用户自定义完整性：班级名称、班长学号不能为空。

### 3.1.3 Course（课程）表

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，课程表中没有出现重复项，满足第一范式；

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，课程号是主码，非主属性都完全依赖于课程号，故满足第二范式；

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该班级表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、满足两类完整性规则：

实体完整性：主码课程号不为空。

用户自定义完整性：课程名称不能为空。

### 3.1.4 Teacher（教师）表

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，教师表中没有出现重复项，满足第一范式；

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，教师号是主码，非主属性都完全依赖于教师工号，故满足第二范式；

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该班级表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、满足三类完整性规则：

实体完整性：主码教师编号不为空。

参照完整性：有外键系别编号，且系别编号不为空。

用户自定义完整性：教师姓名不能为空。

### 3.1.5 Deparment（系别）表

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，系别表中没有出现重复项，满足第一范式；

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，系别号是主码，非主属性都完全依赖于系别号，故满足第二范式；

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该班级表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、满足两类完整性规则：

实体完整性：主码系别编号不为空。

用户自定义完整性：系别名称、系别联系方式、系别人数不能为空。

### 3.1.6 User（用户）表

1、满足三范式：

第一范式满足的条件是元组中的每一个分量都必须是不可分割的数据项，用户表中没有出现重复项，满足第一范式；

第二范式要求在第一范式的基础上，要求所有的非主属性完全依赖于其主码，用户号是主码，非主属性都完全依赖于系别号，故满足第二范式；

第三范式在第二范式的基础上，任何一个非主属性都不传递依赖于任何主关键字，该班级表中的非主属性都不传递依赖于主关键字，故满足第三范式。

2、满足两类完整性规则：

实体完整性：主码课程号不为空。

用户自定义完整性：用户密码、权限信息不能为空。

## 3.2 索引的设计

### 3.2.1索引表

表3-2 索引表

|  |  |
| --- | --- |
| 序号 | 索引 |
|  | FK\_Class\_Dept2 |
|  | FK\_Director2 |
|  | FK\_Monitor |
|  | FK\_CourseTeacher2 |
|  | FK\_CourseTeacher3 |
|  | FK\_CourseTeacher\_course\_id |
|  | FK\_Stu\_Class |
|  | FK\_StudentGrade2 |
|  | FK\_StudentGrade\_stu\_id |
|  | FK\_Teac\_Dept |

### 3.2.2索引目录结构

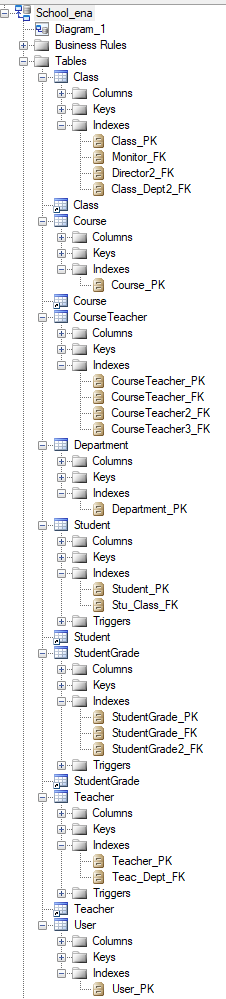
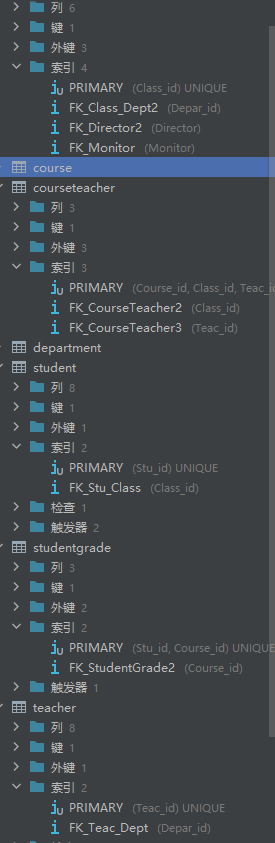


图3-2 MySQL中显示的索引

### 3.2.3索引生成代码

alter table Class add constraint FK\_Class\_Dept2 foreign key (Depar\_id)

references Department (Depar\_id) on delete restrict on update restrict;

alter table Class add constraint FK\_Director2 foreign key (Director)

references Teacher (Teac\_id) on delete restrict on update restrict;

alter table Class add constraint FK\_Monitor foreign key (Monitor)

references Student (Stu\_id) on delete restrict on update restrict;

alter table CourseTeacher add constraint FK\_CourseTeacher2 foreign key (Class\_id)

references Class (Class\_id) on delete restrict on update restrict;

alter table CourseTeacher add constraint FK\_CourseTeacher3 foreign key (Teac\_id)

references Teacher (Teac\_id) on delete restrict on update restrict;

alter table CourseTeacher add constraint FK\_CourseTeacher\_course\_id foreign key (Course\_id)

references Course (Course\_id) on delete restrict on update restrict;

alter table Student add constraint FK\_Stu\_Class foreign key (Class\_id)

references Class (Class\_id) on delete restrict on update restrict;

alter table StudentGrade add constraint FK\_StudentGrade2 foreign key (Course\_id)

references Course (Course\_id) on delete restrict on update restrict;

alter table StudentGrade add constraint FK\_StudentGrade\_stu\_id foreign key (Stu\_id)

references Student (Stu\_id) on delete restrict on update restrict;

alter table Teacher add constraint FK\_Teac\_Dept foreign key (Depar\_id)

references Department (Depar\_id) on delete restrict on update restrict;

## 3.3 视图的设计

### 3.3.1视图表

表3-3 视图的设计

|  |  |  |
| --- | --- | --- |
| 序号 | 视图名称 | 视图功能 |
|  | Course\_select\_num | 显示全部课程的学生人数 |
|  | Class\_stu\_num | 显示全部班级的学生人数 |
|  | Student\_full\_info | 显示学生的具体信息以减少连接次数 |
|  | Teacher\_full\_info | 显示老师的具体信息以减少连接次数 |
|  | Course\_top1 | 显示各个课程最高分 |
|  | Class\_top1 | 显示各个班级最高分 |

### 3.3.2视图目录结构

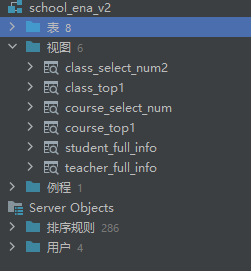
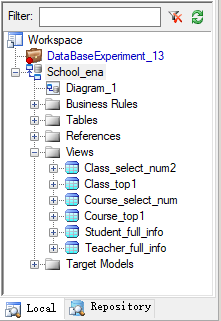


图3-3 PowerDesigner、MySQL中显示的视图

### 3.3.3视图生成代码

/\*==============================================================\*/

/\* View: Class\_select\_num2 \*/

/\*==============================================================\*/

CREATE VIEW Class\_select\_num2 AS select Student.Class\_id, Class\_name,COUNT(Student.Stu\_id) as Stu\_num

from Student, Class

where Student.Class\_id = Class.Class\_id

group by Student.Class\_id,Class\_name;

/\*==============================================================\*/

/\* View: Class\_top1 \*/

/\*==============================================================\*/

CREATE VIEW Class\_top1 AS select Class.Class\_id, Class.Class\_name, MAX(avg\_grade.avg\_grade) as max\_avg\_grade

from StudentGrade, Class, Student,

(select Student.Stu\_id, avg(Grade) as avg\_grade

from Student, StudentGrade

where StudentGrade.Stu\_id = Student.Stu\_id

group by Student.Stu\_id) as avg\_grade

where StudentGrade.Stu\_id = Student.Stu\_id

and Student.Stu\_id = avg\_grade.Stu\_id

and Class.Class\_id = Student.Class\_id

group by Class.Class\_id, Class.Class\_name;

/\*==============================================================\*/

/\* View: Course\_select\_num \*/

/\*==============================================================\*/

CREATE VIEW Course\_select\_num AS select

StudentGrade.Course\_id,

Course\_name,

COUNT(Student.Stu\_id) as Stu\_num

from

Student,

StudentGrade,

Course

where

StudentGrade.Course\_id = Course.Course\_id

and Student.Stu\_id = StudentGrade.Stu\_id

group by

StudentGrade.Course\_id,

Course\_name;

/\*==============================================================\*/

/\* View: Course\_top1 \*/

/\*==============================================================\*/

CREATE VIEW Course\_top1 AS select Course.Course\_id, Course.Course\_name, max(Grade) as max\_grade

from StudentGrade, Course

group by Course.Course\_id, Course.Course\_name;

/\*==============================================================\*/

/\* View: Student\_full\_info \*/

/\*==============================================================\*/

CREATE VIEW Student\_full\_info AS select

Student.Stu\_id,

Student.Stu\_name,

AVG(StudentGrade.Grade) as Score\_avg,

Class.Class\_name,

Teacher.Teac\_name as Director\_name,

Department.Depar\_name

from

Student,

Class,

Teacher,

Department,

StudentGrade

where

Student.Class\_id = Class.Class\_id

and Class.Director = Teacher.Teac\_id

and Student.Stu\_id = StudentGrade.Stu\_id

and Class.Depar\_id = Department.Depar\_id

group by

Student.Stu\_id,

Student.Stu\_name,

Class.Class\_name,

Teacher.Teac\_name,

Department.Depar\_name;

/\*==============================================================\*/

/\* View: Teacher\_full\_info \*/

/\*==============================================================\*/

CREATE VIEW Teacher\_full\_info AS select Teac\_id, Teac\_name, Salary, Teac\_sex, Birthday, TechPost, Depar\_name

from Teacher, Department

where Teacher.Depar\_id = Department.Depar\_id;

## 3.4 触发器的设计

### 3.4.1触发器表

表3-4 触发器表

|  |  |  |
| --- | --- | --- |
| 序号 | 触发器 | 功能 |
|  | Auto\_score\_avg | 插入StudentGrade，同步修改学生平均分 |
|  | Set\_salary\_min | Salary限制最低值（5000(当地公务员平均工资)+课时\*30) |
|  | Auto\_create\_Student\_user | 插入学生，自动在用户表中插入新用户，用户id为学生id，用户密码默认为123456，权限为2 |
|  | Auto\_create\_Teacher\_user | 插入老师，自动在用户表中插入新用户，用户id为学生id，用户密码默认为123456，权限为1 |
|  | Auto\_change\_num | 插入Student，同步修改Class人数，Deparment人数。 |

### 3.4.2触发器目录结构

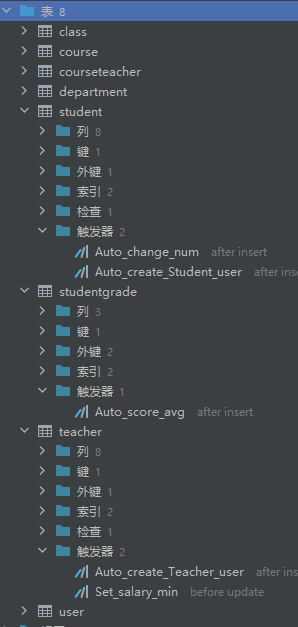
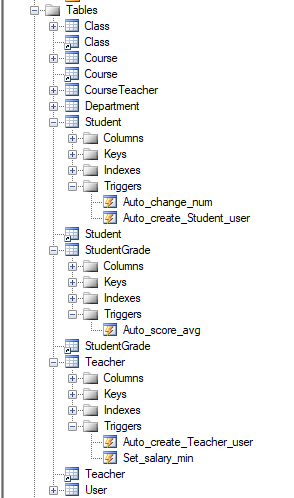


图3-4 PowerDesigner、MySQL中显示的触发器

### 3.4.3触发器生成代码

create trigger Auto\_change\_num

AFTER INSERT

on Student

for EACH ROW

begin

update Department set Depar\_number =

(select count(\*)

from Student, Class

where Student.Class\_id = Class.Class\_id

and Depar\_id = Department.Depar\_id)

+

(select count(\*)

from Teacher

where Depar\_id = Department.Depar\_id);

end;

create trigger Auto\_create\_Student\_user

AFTER INSERT

on Student

for EACH ROW

begin

insert into User

(User\_id, Password, Power)

VALUES

(new.Stu\_id, '123456', 2);

end;

create trigger Auto\_score\_avg

AFTER INSERT

on StudentGrade

for EACH ROW

begin

update Student set Score\_avg =

(select avg(Grade)

from StudentGrade

where StudentGrade.Stu\_id = new.Stu\_id)

where Stu\_id = new.Stu\_id;

end;

create trigger Auto\_create\_Teacher\_user

AFTER INSERT

on Teacher

for EACH ROW

begin

insert into User

(User\_id, Password, Power)

VALUES

(new.Teac\_id, '123456', 1);

end;

create trigger Set\_salary\_min

BEFORE Update

on Teacher

for EACH ROW

begin

DECLARE Total\_course\_hour INT;

Select SUM(Course\_hour) INTO Total\_course\_hour

from Course, CourseTeacher

where Course.Course\_id = CourseTeacher.Course\_id and Teac\_id = new.Teac\_id;

IF new.Salary < 5000 + Total\_course\_hour\*30/12 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'ERROR: Unreasonable Salary!';

END IF;

end;

# 4、数据库实施

## 4.1 代码的生成

见附录

## 4.2 相关内容的改进

1、添加Department的Phone CHECK

2、添加Department主键自增 01->99

3、添加StudentGrade的Grade CHECK

4、Deparment增加总人数Depar\_number

5、Teacher增加工资Salary

6、Class增加总人数Class\_number

7、Student增加平均分Score\_avg

8、添加实体User，权限power，0--管理员，1--老师，2--学生

9、增添视图：每个课程选课人数

10、增添视图：每个课程第一的学生成绩：Course\_id, Course\_name, max\_grade

11、增添视图：每个班平均分第一的学生：Class\_id, Class\_name, max\_grade

12、增添视图：学生全信息视图：Stu\_id, Stu\_name, Score\_avg,Class\_name,Director\_name,Depar\_name

13、增添视图：教师全信息视图：Teac\_id, Teac\_name, Salary,Teac\_sex, Birthday, TechPost, Depar\_name

14、Auto\_score\_avg：插入StudentGrade，同步修改学生平均分。

15、Set\_salary\_min：Salary限制最低值（5000(当地公务员平均工资)+课时\*30)

16、Auto\_create\_Student\_user：插入学生，自动在用户表中插入新用户，用户id为学生id，用户密码默认为123456，权限为2

17、Auto\_create\_Teacher\_user：插入老师，自动在用户表中插入新用户，用户id为学生id，用户密码默认为123456，权限为1

18、Auto\_change\_num：插入Student，同步修改Class人数，Deparment人数。

## 4.3 数据测试

### 4.3.1视图的测试

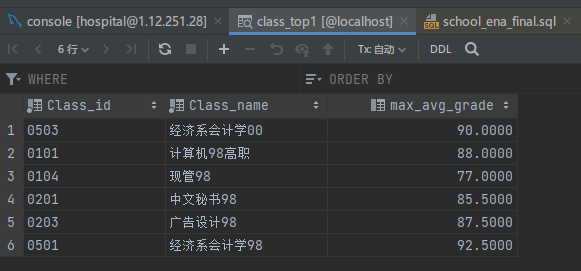


图4-1 各班级最高成绩情况视图

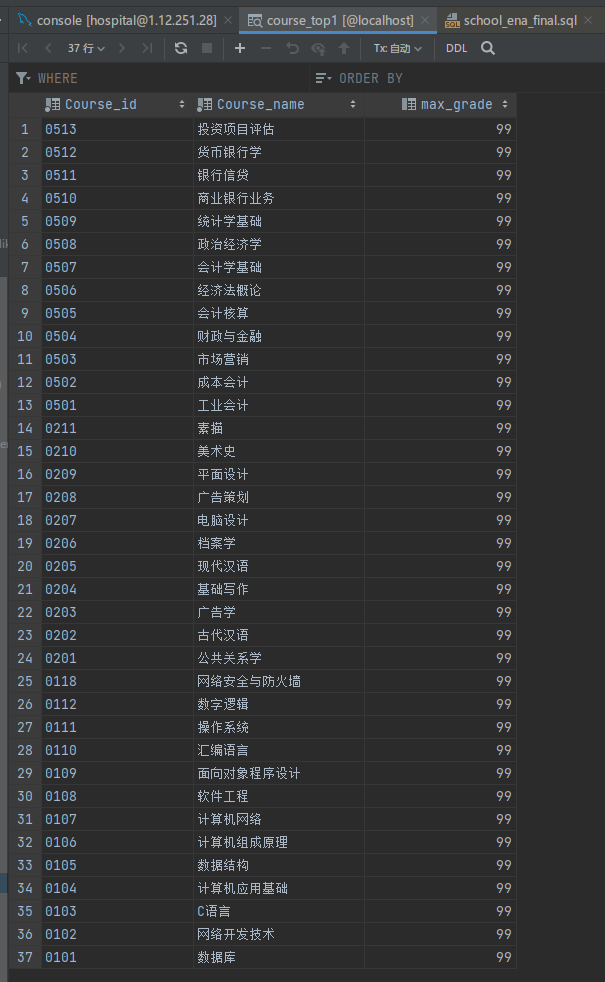


图4-2各课程最高成绩情况视图



图4-3各班人数情况视图

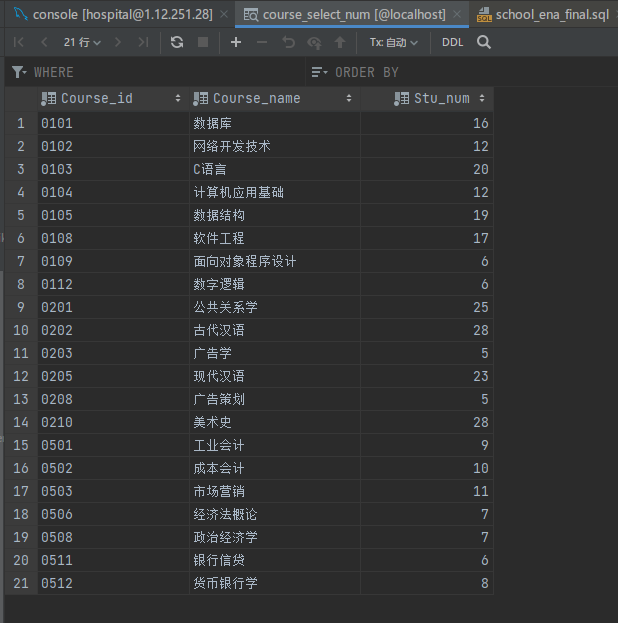


图4-4各课程人数情况视图

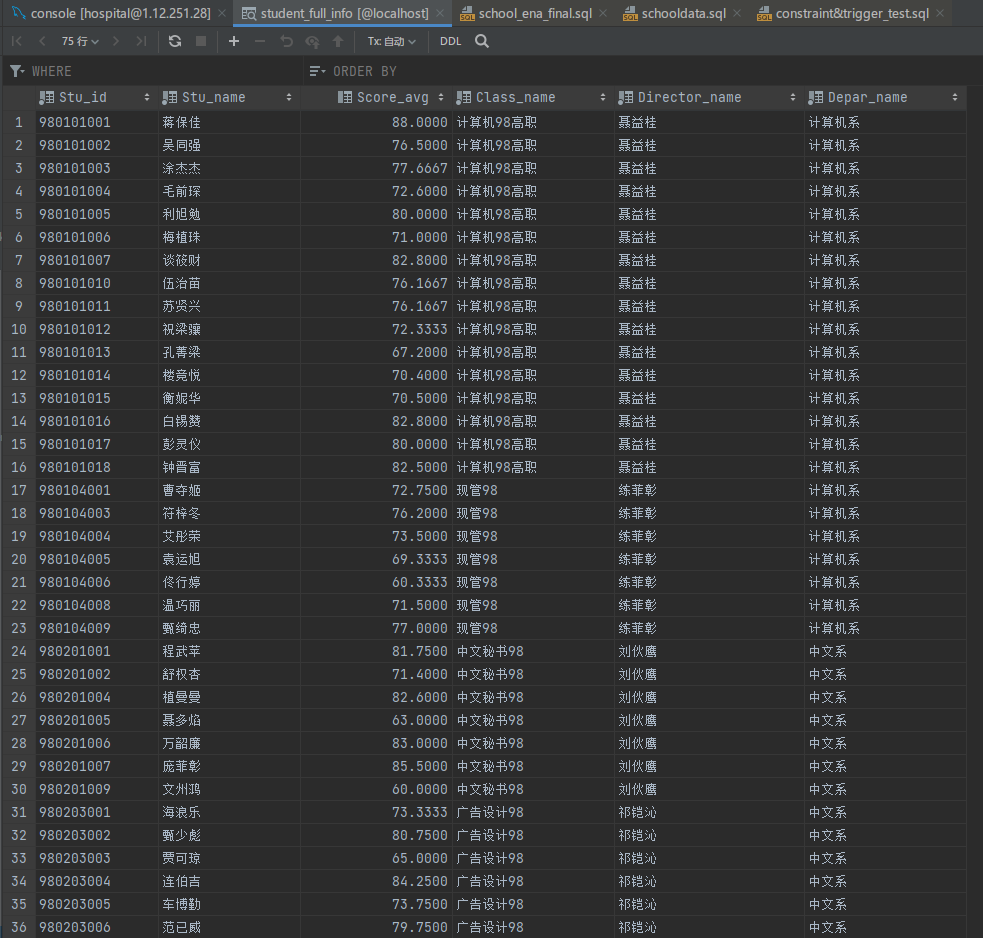


图4-5学生基本信息情况视图

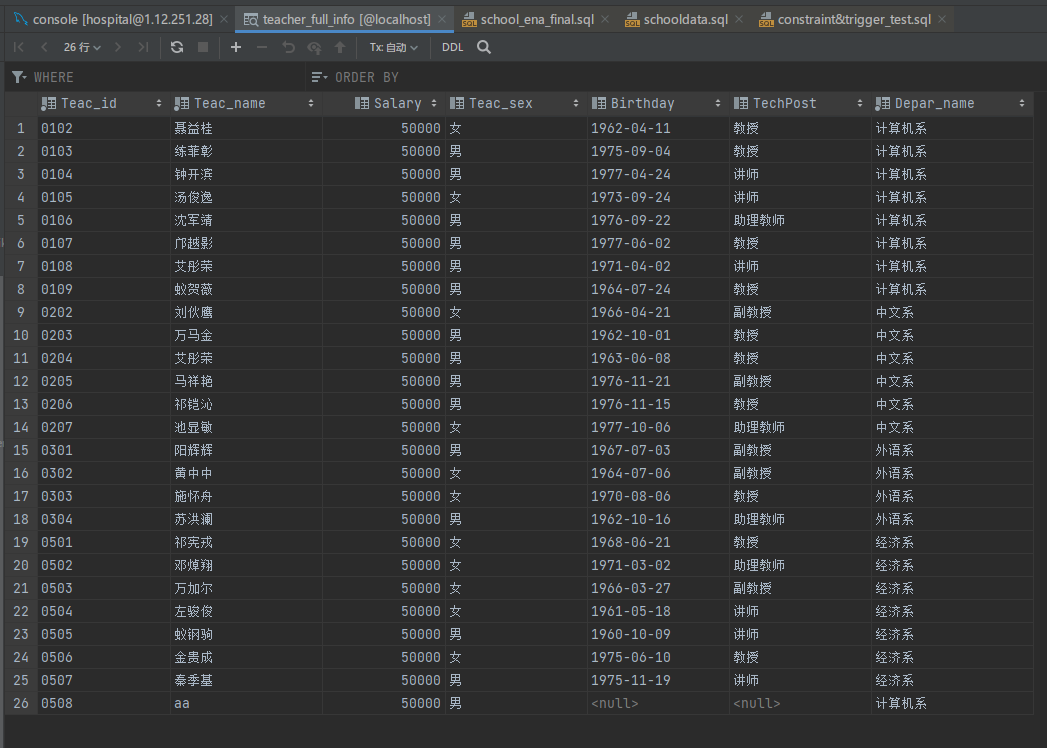


图4-6教师基本情况视图

### 4.3.2约束的测试

1、Department Phone CHECK 测试

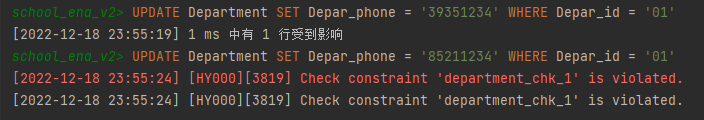


图4-7 Department Phone CHECK测试

2、Student Phone CHECK测试

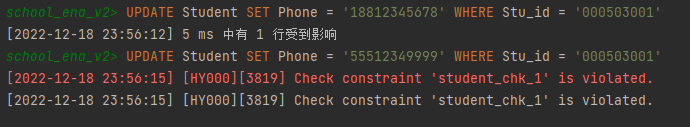


图4-8 Student Phone CHECK测试

3、Teacher Phone CHECK测试

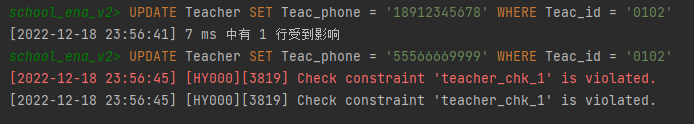


图4-9 Teacher Phone CHECK测试

### 4.3.3触发器测试

1、Auto\_change\_num触发器测试，插入学生自动修改系人数和班级人数

A、插入语句：

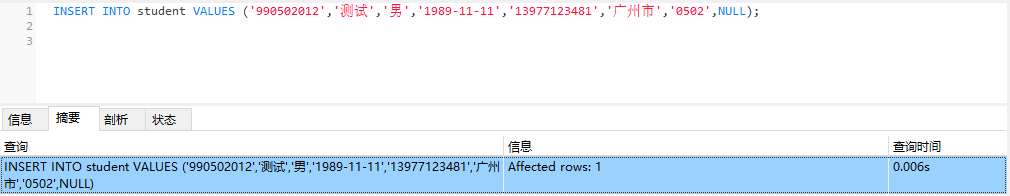


图4-10插入语句

B、Auto\_score\_avg触发器测试Class表和Deparment表在插入操作前状态：



图4-11 Auto\_change\_num触发器测试Class表在插入操作前状态图

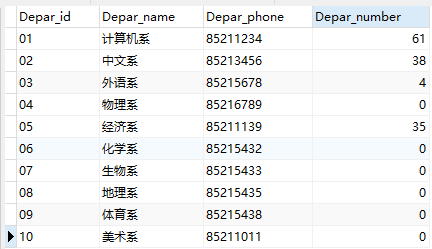


图4-12 Auto\_change\_num触发器测试Deparment表在插入操作前状态图

C、Auto\_change\_num触发器测试Class表和Deparment表在插入操作后状态：

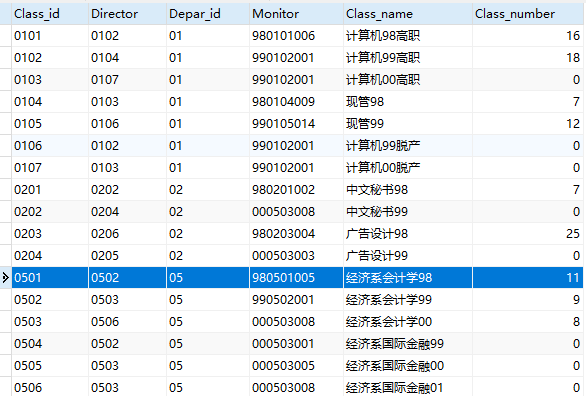


图4-13 Auto\_change\_num触发器测试Class表在插入后状态图

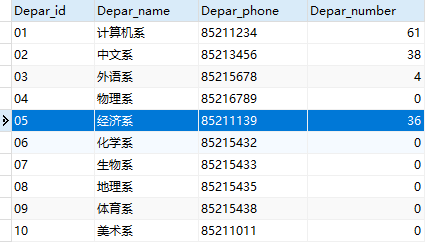


图4-14 Auto\_change\_num触发器测试Deparment表在插入操作前状态图

2、Auto\_score\_avg触发器测试，插入成绩自动修改平均成绩

A、插入成绩前：

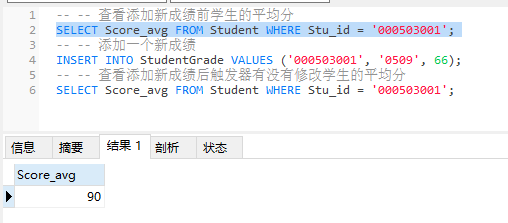


图4-15 插入前成绩图

B、插入语句

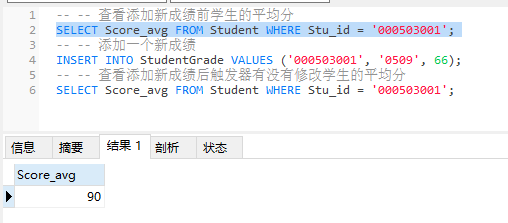


图4-16 插入语句图

C、插入成绩后

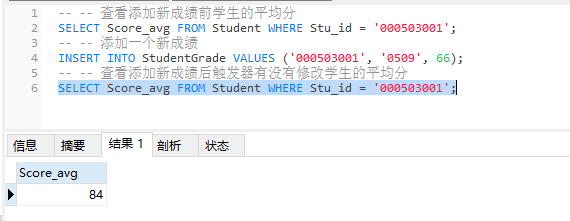


图4-17 插入后成绩图

3、Auto\_create\_Teacher触发器测试，插入老师自动插入账户到User表：

A、插入教师信息到教师表前

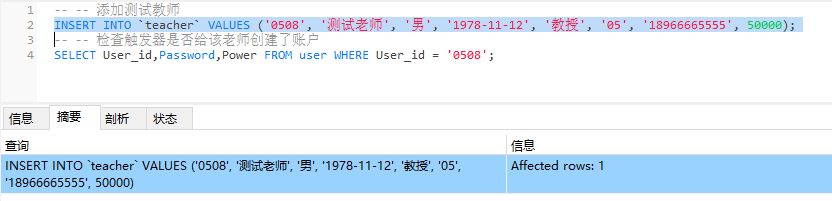


图4-18 插入教师信息到教师表前图

B、插入教师信息到教师表后

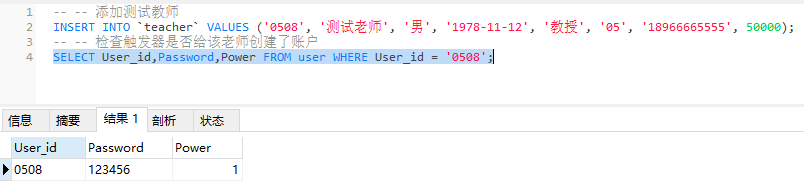


图4-19 插入教师信息到教师表后图

4、Set\_salary\_min触发器测试，限制工资最低值：

A、高于最低值，可以修改：

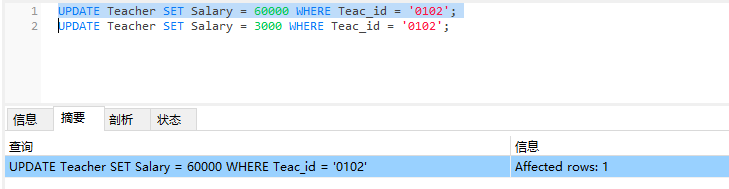


图4-20 高于最低值时修改图

B、低于最低值，抛出自定义错误：

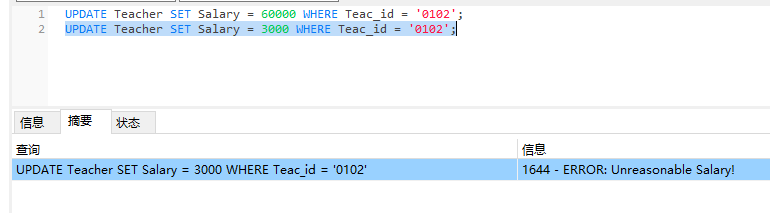


图4-21 低于最低值时修改图

# 附录

/\*==============================================================\*/

/\* DBMS name: MySQL 8.0 \*/

/\* Created on: 18/12/2022 上午10:31:17 \*/

/\*==============================================================\*/

Set sql\_mode = 'STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE, NO\_ZERO\_DATE, ERROR\_FOR\_DIVISION\_BY\_ZERO, NO\_AUTO\_CREATE\_USER, NO\_ENGINE\_SUBSTITUTION';

drop database school\_ena\_v2;

CREATE DATABASE school\_ena\_v2 CHARACTER SET utf8;

USE school\_ena\_v2;

drop procedure if exists getFailStuByCourseId;

DROP TABLE IF EXISTS Teacher\_full\_info;

DROP TABLE IF EXISTS Student\_full\_info;

DROP TABLE IF EXISTS Course\_top1;

DROP TABLE IF EXISTS Course\_select\_num;

DROP TABLE IF EXISTS Class\_top1;

DROP TABLE IF EXISTS Class\_select\_num2;

drop table if exists Class;

drop table if exists Course;

drop table if exists CourseTeacher;

drop table if exists Department;

drop table if exists Student;

drop table if exists StudentGrade;

drop table if exists Teacher;

drop table if exists User;

/\*==============================================================\*/

/\* Table: Class \*/

/\*==============================================================\*/

create table Class

(

Class\_id varchar(6) not null,

Director varchar(4),

Depar\_id varchar(2),

Monitor varchar(9) not null,

Class\_name varchar(20) not null,

Class\_number int,

primary key (Class\_id)

);

/\*==============================================================\*/

/\* Table: Course \*/

/\*==============================================================\*/

create table Course

(

Course\_id varchar(4) not null,

Course\_name varchar(20) not null,

Course\_hour smallint not null,

Introduce varchar(200),

primary key (Course\_id)

);

/\*==============================================================\*/

/\* Table: CourseTeacher \*/

/\*==============================================================\*/

create table CourseTeacher

(

Course\_id varchar(4) not null,

Class\_id varchar(6) not null,

Teac\_id varchar(4) not null,

primary key (Course\_id, Class\_id, Teac\_id)

);

/\*==============================================================\*/

/\* Table: Department \*/

/\*==============================================================\*/

create table Department

(

Depar\_id varchar(2) not null,

Depar\_name varchar(50) not null,

Depar\_phone varchar(8),

Depar\_number int,

primary key (Depar\_id),

check (Depar\_phone LIKE '3935[0-9][0-9][0-9][0-9]')

);

/\*==============================================================\*/

/\* Table: Student \*/

/\*==============================================================\*/

create table Student

(

Stu\_id varchar(9) not null,

Stu\_name varchar(8) not null,

Stu\_sex varchar(2) not null,

Birthday date not null,

Phone varchar(15),

Address varchar(100),

Class\_id varchar(6),

Score\_avg int,

primary key (Stu\_id),

check (Phone LIKE '^[1]([3-9])[0-9]{9}$')

);

/\*==============================================================\*/

/\* Table: StudentGrade \*/

/\*==============================================================\*/

create table StudentGrade

(

Stu\_id varchar(9) not null,

Course\_id varchar(4) not null,

Grade smallint,

primary key (Stu\_id, Course\_id)

);

/\*==============================================================\*/

/\* Table: Teacher \*/

/\*==============================================================\*/

create table Teacher

(

Teac\_id varchar(4) not null,

Teac\_name varchar(8) not null,

Teac\_sex varchar(2),

Birthday date,

TechPost varchar(10),

Depar\_id varchar(2),

Teac\_phone varchar(11),

Salary int,

primary key (Teac\_id),

check (Teac\_phone LIKE '^[1]([3-9])[0-9]{9}$')

);

/\*==============================================================\*/

/\* Table: User \*/

/\*==============================================================\*/

create table User

(

User\_id varchar(4) not null,

Password varchar(20) not null,

Power int not null,

primary key (User\_id)

);

create procedure getFailStuByCourseId (IN cid varchar(4))

BEGIN

Select Stu\_id, Grade

From StudentGrade

Where Course\_id = cid and Grade < 60;

END;