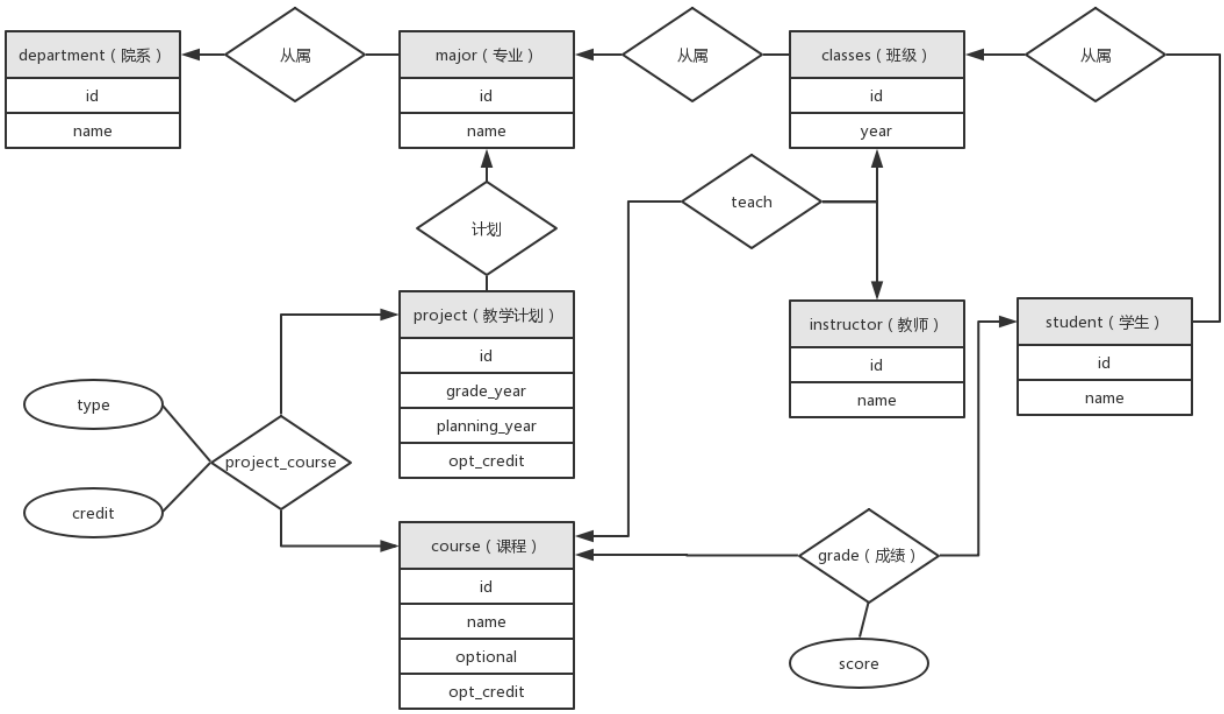
数据库第5次上机报告

罗阳豪 16130120191

需求分析

该学籍管理系统要求能够记录学籍管理所需的必要信息，需要记录学院、专业、班级、学生、课程、教师等信息，还有他们之间的关系。在这些信息之上，要能实现一定的查询操作，包括查询学生成绩，统计平均成绩，查询教过某学生的全部老师，查询应被开除的学生名单。

概念结构设计



逻辑结构设计

根据E-R图建立10张表，如下

- department(id, name);
- major(id, dept_id, name);
- classes(id, major_id, grade_year);
- student(id, class_id, name);
- instructor(id, name);
- course(id, name, optional, opt_credit);
- project(id, major_id, grade_year, planning_year, opt_credit);
- project_course(project_id, course_id, type, credit);
- grade(student_id, course_id, score);

- teach(class_id, instructor_id, course_id).

应用程序设计中遇到的问题及解决方法

在该应用设计过程中，主要遇到了两个问题

一是，表太多，为了使数据库设计尽可能符合三范式，数据被拆分在很多表中，完成一个查询通常要连接四五个表。这样一来，连接的顺序和连接的条件的斟酌就变得很麻烦，很容易引起混乱。解决的方法就是，通过图来辅助自己分析，比如在E-R图中找关系，这比看SQL要直观得多。

二是，数据之间的依赖关系很复杂，插入数据时，保持关系的完整性就十分有挑战。解决方法就是，尽量多得设置约束，如外键、主键，还有就是通过业务流程保证数据的完整性，做仔细的参数检查。

建立数据库的主要代码

```
CREATE TABLE department (
  id VARCHAR(32) PRIMARY KEY NOT NULL,
  name VARCHAR(128) NOT NULL
);

CREATE TABLE major (
  id VARCHAR(32) PRIMARY KEY NOT NULL,
  dept_id VARCHAR(32),
  name VARCHAR(128) NOT NULL,
  FOREIGN KEY (dept_id) REFERENCES department (id)
);

CREATE TABLE classes (
  id VARCHAR(32) PRIMARY KEY NOT NULL,
  major_id VARCHAR(32),
  grade_year YEAR,
  FOREIGN KEY (major_id) REFERENCES major (id)
);

CREATE TABLE student (
  id VARCHAR(32) PRIMARY KEY NOT NULL,
  class_id VARCHAR(32),
  name VARCHAR(128) NOT NULL,
  FOREIGN KEY (class_id) REFERENCES classes (id)
);

CREATE TABLE instructor (
  id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
  name VARCHAR(128)
);

CREATE TABLE course (
  id VARCHAR(32) PRIMARY KEY NOT NULL,
  name VARCHAR(128) NOT NULL,
  optional BOOLEAN DEFAULT FALSE NOT NULL,
```

```
  opt_credit NUMERIC(2, 1) DEFAULT 0 NOT NULL
);

CREATE TABLE project (
  id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
  major_id VARCHAR(32),
  grade_year YEAR,
  planning_year YEAR,
  opt_credit INTEGER,
  FOREIGN KEY (major_id) REFERENCES major (id)
);

CREATE TABLE project_course (
  project_id INTEGER,
  course_id VARCHAR(32),
  type INTEGER,
  credit NUMERIC(2, 1),
  PRIMARY KEY (project_id, course_id),
  FOREIGN KEY (project_id) REFERENCES project (id),
  FOREIGN KEY (course_id) REFERENCES course (id)
);

CREATE TABLE grade (
  student_id VARCHAR(32),
  course_id VARCHAR(32),
  score NUMERIC(4, 1),
  PRIMARY KEY (student_id, course_id),
  FOREIGN KEY (student_id) REFERENCES student (id),
  FOREIGN KEY (course_id) REFERENCES course (id)
);

CREATE TABLE teach (
  class_id VARCHAR(32),
  instructor_id INTEGER,
  course_id VARCHAR(32),
  CONSTRAINT uni_class_instructor UNIQUE (class_id, instructor_id),
  PRIMARY KEY (class_id, instructor_id, course_id),
  FOREIGN KEY (class_id) REFERENCES classes (id),
  FOREIGN KEY (instructor_id) REFERENCES instructor (id),
  FOREIGN KEY (course_id) REFERENCES course (id)
);
```

查询的主要代码

查询学生所选修的课程及成绩，并给出必修课平均成绩和选修课平均成绩

```
# 1.    查询学生所选修的课程及成绩，并给出必修课平均成绩和选修课平均成绩；
# 查询成绩
SELECT
  student.id AS "Student ID",
  student.name AS "Student Name",
  course.id AS "Course ID",
```

```
  course.name AS "Course Name",
  grade.score AS "Score"
FROM
  grade
  JOIN student ON grade.student_id = student.id
  JOIN course ON grade.course_id = course.id
ORDER BY student_id, course_id;

# 计算平均成绩
SELECT *
FROM (
    SELECT grade.student_id AS "Student ID", student.name AS "Student Name", avg(grade.score) AS
"Required Course Average Score"
    FROM grade JOIN student ON grade.student_id = student.id
    WHERE course_id IN (
      SELECT course_id
      FROM
        project_course JOIN
        project ON project_course.project_id = project.id JOIN
        classes ON project.major_id = classes.major_id JOIN
        student ON classes.id = student.class_id
      WHERE student.id = grade.student_id AND project_course.type = 1
    )
    GROUP BY student_id
    ORDER BY student_id
  ) req NATURAL JOIN (
    SELECT grade.student_id AS "Student ID", student.name AS "Student Name", avg(grade.score) AS
"Selected Course Average Score"
    FROM grade JOIN student ON grade.student_id = student.id
    WHERE course_id IN (
      SELECT course_id
      FROM
        project_course JOIN
        project ON project_course.project_id = project.id JOIN
        classes ON project.major_id = classes.major_id JOIN
        student ON classes.id = student.class_id
      WHERE student.id = grade.student_id AND project_course.type = 2
    )
    GROUP BY student_id
    ORDER BY student_id
  ) sel NATURAL JOIN (
    SELECT grade.student_id AS "Student ID", student.name AS "Student Name", avg(grade.score) AS
"Optional Course Average Score"
    FROM grade JOIN student ON grade.student_id = student.id
    WHERE course_id NOT IN (
      SELECT course_id
      FROM
        project_course JOIN
        project ON project_course.project_id = project.id JOIN
        classes ON project.major_id = classes.major_id JOIN
        student ON classes.id = student.class_id
      WHERE student.id = grade.student_id
    )
```

```
    GROUP BY student_id
    ORDER BY student_id
  ) opt;
```

查某一个学生被哪些教师教过课

```
# 2.    查某一个学生被哪些教师教过课;
# 假定其学号为“16130120191”
SELECT course.id AS "Course ID", course.name AS "Course Name", instructor.name AS "Instructor
Name"
FROM
  project_course JOIN
  course ON course.id = project_course.course_id JOIN
  project ON project_course.project_id = project.id JOIN
  classes ON project.major_id = classes.major_id JOIN
  student ON classes.id = student.class_id JOIN
  teach ON classes.id = teach.class_id AND project_course.course_id = teach.course_id JOIN
  instructor ON instructor.id = teach.instructor_id
WHERE student.id = '16130120191';
```

查询应被开除的学生（假定差2学分即被开除）

```
# 3.    查询应被开除的学生（假定差2学分即被开除）。
SELECT fail_1.id, fail_1.name
FROM (
    SELECT student.id, student.name, sum(project_course.credit) AS fail_credit
    FROM
    project_course JOIN
    project ON project.id = project_course.project_id JOIN
    classes ON classes.major_id = project.major_id JOIN
    student ON classes.id = student.class_id JOIN
    grade ON project_course.course_id = grade.course_id AND project_course.type = 1 AND
student.id = grade.student_id AND grade.score < 60
    GROUP BY project.id, student.id
  UNION (
    SELECT student.id, student.name, sum(project_course.credit) AS fail_credit
    FROM
      project_course JOIN
      project ON project.id = project_course.project_id JOIN
      classes ON classes.major_id = project.major_id JOIN
      student ON classes.id = student.class_id JOIN
      grade ON project_course.course_id = grade.course_id AND project_course.type = 1 AND
student.id = grade.student_id AND grade.score < 60
    GROUP BY student.id
  ) UNION (
    SELECT grade.student_id AS "Student ID", student.name AS "Student Name",
sum(course.opt_credit) AS fail_credit
```

```
    FROM grade JOIN student ON grade.student_id = student.id JOIN course ON course.id =
grade.course_id
    WHERE course_id NOT IN (
      SELECT course_id
      FROM
        project_course JOIN
        project ON project_course.project_id = project.id JOIN
        classes ON project.major_id = classes.major_id JOIN
        student ON classes.id = student.class_id
      WHERE student.id = grade.student_id
    ) AND grade.score < 60
    GROUP BY student_id
    ORDER BY student_id
  )) fail_1
WHERE fail_credit > 2;
```