目录

[环境准备 2](#_Toc87537882)

[数据库 2](#_Toc87537883)

[数据库设计 3](#_Toc87537884)

[Department表 3](#_Toc87537885)

[Student表 3](#_Toc87537886)

[Teacher表 4](#_Toc87537887)

[Course表 4](#_Toc87537888)

[TeachingArrangement表 4](#_Toc87537889)

[CourseOption表 4](#_Toc87537890)

# 环境准备

## 数据库

安装MySQL Server 8.x版本和MySQL Workbench（为了可视化操作）。

Windows用户要注意：

使用官方社区版安装包安装MySQL服务器时，勾选进行高级配置，在高级配置中选定“保持给出的大小写”，否则会导致与Unix上的行为不一致。我这里还初始化了一个用户admin。

配置正确的话，系统服务里会多出一个MySQL80（也可能是其他数字结尾，取决于你安装的版本）。

执行exp1中的初始化脚本。.ps1后缀是PowerShell脚本，.sh后缀是Linux脚本（不保证兼容所有解释器）。

* init/init：建库建表
* init/dump：删除库
* input：初始化测试数据（会删除已有的数据！）
* login：登录，会自动切换到EducationDB库（也就是脚本建的库）

# 数据库设计

E-R图见ProcessOn。

我的代码中，库名是EducationDB。

数据库使用utf8mb4（也即完整的utf8字符集。mysql的utf8是真正utf8的缩减版）

说明：所有主键均使用BIGINT类型。

## Department表

* (编号, 系名称)
* 约束：
  + 系名称非空唯一
* 类型：
  + 系名称：VARCHAR(32)

该表存储系的名称，附带一个主键（作为其他表引用时的外键）。

如果直接在各个表中存储系名，容易造成信息冗余。直接存名称的另一个显而易见的坏处是，如果录入时录入错了，会导致库里出现不存在的系，而且**无法被检索**。这样的错误毫无疑问会造成管理上的困难，并且难以排查。极端情况下可能会造成实际上的经济损失，甚至可能出现法律问题（例如，数据库打出的证书上具有不存在的系名，由此导致的合同有效性争议）。

分开系名和id的存储，还有一个额外的好处：如果哪个系改名了，只需要改一个表中的一行，不会对服务器造成较大负担。

## Student表

* (学号, 姓名, 性别, 生日, 所属系)
* 约束：
  + 全部字段非空
  + 性别必须是以下三项之一：
    - 男
    - 女
    - 其他（主要由于指示无法或尚未判定性别的特殊情形，例如准备手术的双性畸形）
  + 所属系是关联到Department表的主键，作为本表外键
* 类型：
  + 姓名：VARCHAR(32)
  + 性别：VARCHAR(8)（为了便于开发中的自行扩展，预留了多于需要的长度）
  + 生日：DATE

该表存储学生信息。

**我们这里的设计和指导书有一处重要不同：存储的并非年龄，而是出生日期。**

## Teacher表

* (教工号, 姓名, 性别, 所属系)
* 约束：
  + 全部字段非空
  + 所属系是关联到Department表的主键，作为本表外键
* 类型：
  + 姓名：VARCHAR(32)
  + 性别：VARCHAR(8)（为了便于开发中的自行扩展，预留了多于需要的长度）

该表记录教师信息。

## Course表

* (课程号, 课程名, 学时, 学分)
* 约束：
  + 全部字段非空
* 类型：
  + 课程名：VARCHAR(32)
  + 学时：SMALLINT
  + 学分：NUMERIC(4,2)

该表记录课程信息。

## TeachingArrangement表

* (编号, 教工号, 课程号)
* 约束：
  + 全部字段非空
  + 教工号和课程号分别是引用自Teacher和Course表的外键
  + 教工号和课程号构成一对唯一索引

该表记录教学安排。

业务逻辑：只记录当前教学周期内的安排（也就是不对历史记录进行持久化）。

## CourseOption表

* (编号, 学号, 课程号, 成绩, 选课年份)
* 约束：
  + 除了成绩列外，所有字段非空
  + 学号是关联到Student表的外键
  + 课程号是关联到Course的外键
* 类型：
  + 成绩：NUMERIC(5,2)（最大记录到100.00）
  + 选课年份：TINYINT

该表记录选课信息和成绩信息。当成绩列为空时，表示选课了，但尚未给成绩。

**为什么引用课程号而非排课编号：排课表记录的是多名教师上同一堂课的情形。我们假设完全相同的课不会同时安排不同的若干排课。**

**假设：一门课要么开在上半学年，要么开在下半学年。**

注：AVG会忽略NULL。

# 实验一

## 把表转为人类读得懂的信息

### 列出所有学生信息

SELECT Sno AS 学号, Sname AS 姓名, Sgender AS 性别, Sbirth AS 生日, Dname AS 所属系

FROM Student, Department

WHERE Student.Dno = Department.Dno;

### 列出所有教师信息

SELECT Tno AS 教工号, Tname AS 姓名, Tgender AS 性别, Dname AS 所属系

FROM Teacher, Department

WHERE Teacher.Dno = Department.Dno;

### 列出所有排课信息

SELECT C.Cno AS 课程号, Cname AS 课程名, Tname AS 任课教师, Ctime AS 学时, Ccredit AS 学分

FROM Course AS C, Teacher AS T, TeachingArrangement AS TA

WHERE C.Cno = TA.Cno AND T.Tno = TA.Tno;

### 列出所有选课信息

SELECT S.Sno AS 学号, Sname AS 姓名, C.Cno AS 课程号, Cname AS 课程, COyear AS 修读年份, COgrade AS 成绩

FROM Student AS S, Course AS C, CourseOption AS CO

WHERE S.Sno = CO.Sno AND C.Cno = CO.Cno;

## 官方题目

### 查询学生学号、姓名、出生年份

SELECT Sno AS 学号, Sname AS 姓名, YEAR(Sbirth) AS 出生年份 FROM Student;

（如果按原数据库设计，就将YEAR(birth)改为YEAR(NOW()) - Sage即可）

### 对每个系按学生总人数降序排列

SELECT Dname AS 系, COUNT(\*) AS 人数 FROM Student, Department

WHERE Department.Dno = Student.Dno

GROUP BY Student.Dno

ORDER BY COUNT(\*) DESC;

### 选修课程号2、3的学生学号、课程名和成绩

（这里忽略了尚未出成绩的学生，改的话也很好改。为了方便查看，这里加了学生姓名）

SELECT S.Sno AS 学号, Sname AS 姓名, C.Cname AS 课程名, COgrade AS 成绩

FROM Student AS S, Course AS C, CourseOption AS CO

WHERE S.Sno = CO.Sno AND C.Cno = CO.Cno

AND (CO.Cno = 2 OR CO.Cno = 3)

AND NOT ISNULL(COgrade);

### 选修某课程多于3人的教师姓名

这题目有点绕，我理解为重点是某课程选修多于3人，查出任课教师姓名。

SELECT DISTINCT Tname AS 教师

FROM Teacher AS T, (

SELECT Cno, COUNT(\*) AS Camount

FROM CourseOption AS CO

GROUP BY Cno

HAVING COUNT(Cno) > 3

) AS COstat,

TeachingArrangement AS TA

WHERE T.Tno = TA.Tno AND COstat.Cno = TA.Cno;

（测试数据中，所有教师均有大于3人选课的课程，演示时可以主动提出换个数。）

这个是能看到具体哪些课程选课人数大于3的：

SELECT Tname AS 任课教师, Cname AS 课程, Camount AS 选课人数

FROM Teacher AS T, Course AS C, (

SELECT Cno, COUNT(\*) AS Camount

FROM CourseOption AS CO

GROUP BY Cno

HAVING COUNT(Cno) > 3

) AS COstat,

TeachingArrangement AS TA

WHERE C.Cno = COstat.Cno AND T.Tno = TA.Tno

AND COstat.Cno = TA.Cno;

### 未选课学生的姓名和所在系

这里我们的理解是排除既往选课记录。

SELECT Sname AS 姓名, Dname AS 所属系

FROM Student AS S, Department AS D

WHERE S.Dno = D.Dno AND

Sno NOT IN (

SELECT DISTINCT Sno

FROM CourseOption AS CO

WHERE ISNULL(COgrade)

);

### 求每个学生得分最高的课，输出姓名、课程号、最高分

为了便于查看，这里还额外加入了课程名。

SELECT Sname AS 姓名, C.Cno AS 课程号, Cname AS 课程名, COmax AS 最高分

FROM Student AS S, Course AS C, CourseOption AS CO, (

SELECT Sno, MAX(COgrade) AS COmax

FROM CourseOption AS CO

GROUP BY Sno

) AS COstat

WHERE S.Sno = COstat.Sno AND CO.COgrade = COstat.COmax AND

C.Cno = CO.Cno AND CO.Sno = S.Sno;

### 所有讲授数据结构的教师姓名

SELECT DISTINCT Tname AS 教师

FROM Teacher AS T, Course AS C, TeachingArrangement AS TA

WHERE T.Tno = TA.Tno AND C.Cno = TA.Cno AND C.Cname = "数据结构";

### 选修了李正科老师课程的学生信息

测试数据中选修了李正科老师的课的学生是全部，建议考虑换个别的老师名字来演示。

SELECT DISTINCT S.Sno AS 学号, Sname AS 姓名, Sgender AS 性别, Sbirth AS 生日, Dname AS 所属系

FROM Student AS S, Department AS D, CourseOption AS CO, (

SELECT C.Cno

FROM Course AS C, Teacher AS T, TeachingArrangement AS TA

WHERE T.Tname = "李正科" AND T.Tno = TA.Tno AND TA.Cno = C.Cno

) AS COstat

WHERE S.Sno = CO.Sno AND CO.Cno = COstat.Cno AND D.Dno = S.Dno;

### 新建表并导入CS系的数据

说实在的，我感觉这个题对脚本使用者没啥意义。。。

CREATE TABLE S1 (

SNO BIGINT AUTO\_INCREMENT PRIMARY KEY,

SNAME VARCHAR(32) NOT NULL,

SA DATE NOT NULL,

SD BIGINT NOT NULL,

FOREIGN KEY (SD) REFERENCES Department(Dno)

); -- 没错，就是把Student表的几个字段改了个名重新复制了一遍

INSERT INTO S1 (SNO, SNAME, SA, SD)

SELECT Sno, Sname, Sbirth, S.Dno FROM Student AS S, Department AS D

WHERE S.Dno = D.Dno AND D.Dname = "计算机科技";

### 利用视图将CS系1号课程的所有学生成绩提高5分

CREATE VIEW STUDENT\_CS AS

SELECT S.\* FROM Student AS S, Department AS D

WHERE S.Dno = D.Dno AND D.Dname = "计算机科技";

UPDATE CourseOption AS CO

JOIN STUDENT\_CS AS S

ON CO.Sno = S.Sno

SET COgrade = CASE

WHEN COgrade > 95.00 THEN 100.00

ELSE COgrade + 5

END

WHERE CO.Cno = 1;

### 带参数存储过程查询数据结构分数在指定分数段的学生

题目要求查询姓名、所属系、成绩。这里我们采用闭区间（否则无法获取满分或0分的数据）。

DELIMITER $$

CREATE PROCEDURE foo(

IN low NUMERIC(5,2),

IN high NUMERIC(5,2)

) BEGIN

SELECT Sname AS 姓名, Dname AS 所属系, COgrade AS 成绩

FROM Student AS S, Department AS D, CourseOption AS CO, Course AS C

WHERE S.Sno = CO.Sno AND CO.Cno = C.Cno AND D.Dno = S.Dno

AND C.Cname = "数据结构" AND CO.COgrade >= low AND CO.COgrade <= high;

END$$

DELIMITER ;

CALL foo(60, 80);

**SQL SERVER的执行方式与此不同，应该是EXEC。**