# Assignment 2: Database Design

### 姓名：谢宝玛 学号：1120233506

## 建立学科字典

### **1.**建表

SQL：

-- 创建学科门类表

CREATE TABLE Category (

CategoryID VARCHAR(2) PRIMARY KEY,

CategoryName VARCHAR(50)

);

-- 创建一级学科表

CREATE TABLE Major1 (

Major1ID VARCHAR(4) PRIMARY KEY,

CategoryID VARCHAR(2),

Major1Name VARCHAR(100),

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

-- 创建二级学科表

CREATE TABLE Major2 (

Major2ID VARCHAR(6) PRIMARY KEY,

Major1ID VARCHAR(4),

Major2Name VARCHAR(100),

FOREIGN KEY (Major1ID) REFERENCES Major1(Major1ID)

);

-- 创建三级学科表

CREATE TABLE Major3 (

Major3ID INT PRIMARY KEY AUTO\_INCREMENT,

Major2ID VARCHAR(6),

Major3Name VARCHAR(100),

FOREIGN KEY (Major2ID) REFERENCES Major2(Major2ID)

);

插入数据：

SQL：  
-- 插入学科门类数据

INSERT INTO Category (CategoryID, CategoryName) VALUES

('07', '理学'),

('08', '工学'),

('09', '农学');

-- 插入一级学科数据

INSERT INTO Major1 (Major1ID, CategoryID, Major1Name) VALUES

('0811', '08', '控制科学与工程'),

('0812', '08', '计算机科学与技术'),

('0813', '08', '建筑学');

-- 插入二级学科数据

INSERT INTO Major2 (Major2ID, Major1ID, Major2Name) VALUES

('081201', '0812', '计算机系统结构'),

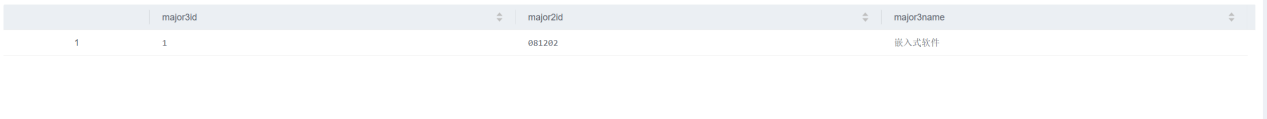
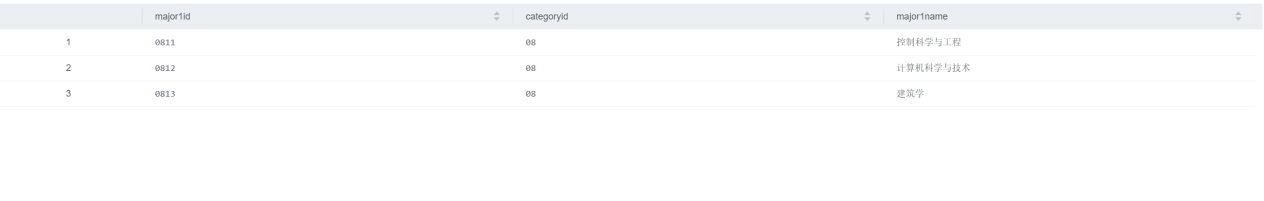
('081202', '0812', '计算机软件与理论'),

('081203', '0812', '计算机应用技术');

-- 插入三级学科数据

INSERT INTO Major3 (Major2ID, Major3Name) VALUES

('081202', '嵌入式软件');

结果：  


### **2.**编码长度改变

#### 一级学科变为3位

SQL：

1）删除外键约束

ALTER TABLE Major2 DROP FOREIGN KEY fk\_major1;

2）修改major1的结构

ALTER TABLE Major1

MODIFY COLUMN Major1ID VARCHAR(3) PRIMARY KEY;

3）重新创建外键约束

ALTER TABLE Major2

ADD CONSTRAINT fk\_major1

FOREIGN KEY (Major1ID)

REFERENCES Major1 (Major1ID);

1. 插入数据

INSERT INTO Major1 (Major1ID, CategoryID, Major1Name) VALUES

('811', '08', '控制科学与工程'),

('812', '08', '计算机科学与技术'),

('813', '08', '建筑学');

### **3.**编码方式改变

#### 增加4级学科

SQL：

-- 创建四级学科表

CREATE TABLE Major4 (

Major4ID INT PRIMARY KEY AUTO\_INCREMENT,

Major3ID INT,

Major4Name VARCHAR(100),

FOREIGN KEY (Major3ID) REFERENCES Major3(Major3ID)

);

-- 插入四级学科数据

INSERT INTO Major4 (Major3ID, Major4Name) VALUES

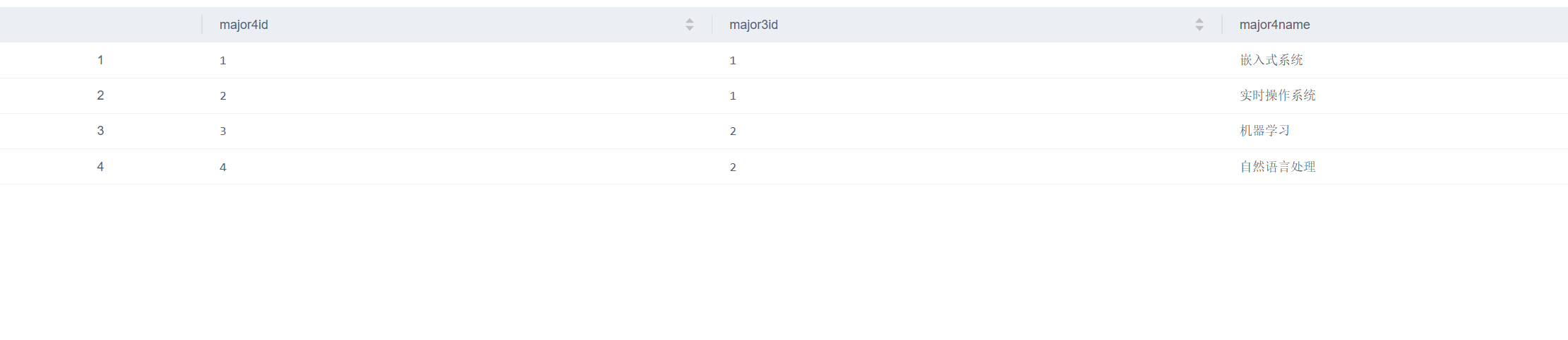
((SELECT Major3ID FROM Major3 WHERE Major3Name = '嵌入式软件'), '嵌入式系统'),

((SELECT Major3ID FROM Major3 WHERE Major3Name = '嵌入式软件'), '实时操作系统'),

((SELECT Major3ID FROM Major3 WHERE Major3Name = '人工智能'), '机器学习'),

((SELECT Major3ID FROM Major3 WHERE Major3Name = '人工智能'), '自然语言处理');

结果：



#### 将门类与一级学科合并

将机器学习化为一级学科

1)查找 "工学" 的 CategoryID (假设为 '08')：

SQL:

SELECT CategoryID FROM Category WHERE CategoryName = '工学';-- 假设查询结果是 '08'

2)插入 "机器学习" 到 Major1 表：

你需要生成一个新的、唯一的 Major1ID。假设你决定使用 '0814'。

SQL:

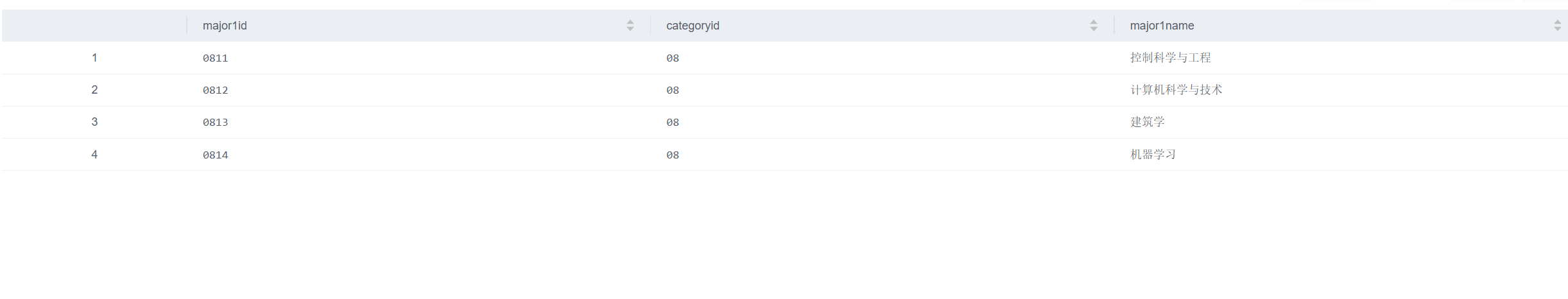
INSERT INTO Major1 (Major1ID, CategoryID, Major1Name) VALUES

('0814', '08', '机器学习');

3)删除 Major4 表中名为 "机器学习" 的记录：

SQL:

DELETE FROM Major4 WHERE Major4Name = '机器学习';

结果：  


### **4.**代码有效期改变

#### 如某个二级学科从2010年10月1日开始不再使用

1）修改表结构： 为 Major2 表添加一个 ExpirationDate 列，用于存储过期时间。

SQL

ALTER TABLE Major2ADD ExpirationDate DATE;

2）更新数据： 将 "计算机应用技术" 的过期时间设置为 2025 年 10 月 1 日。

SQL

UPDATE Major2SET ExpirationDate = '2025-10-01'WHERE Major2Name = '计算机应用技术';

3）创建计划任务（可选）： 你需要创建一个计划任务（例如，使用 MySQL 的事件调度器或操作系统的 cron 任务）来定期检查 Major2 表中 ExpirationDate 列的值，并删除已过期的记录。 以下是一个 MySQL 事件调度器的示例，它每天检查并删除过期记录：

SQL

CREATE EVENT delete\_expired\_major2ON SCHEDULE EVERY 1 DAY

DO

DELETE FROM Major2

WHERE ExpirationDate < CURDATE();

### **5.**代码被覆盖或替换

#### 1）081202（计算机软件与理论 ）改变为软件工程，新增081204（计算机软件与理论）

SQL:  
UPDATE Major2

SET Major2Name = '软件工程'

WHERE Major2ID = '081202';

INSERT INTO Major2 (Major2ID, Major1ID, Major2Name)

VALUES ('081204', '0812', '计算机软件与理论');

结果：



## 建立学生数据库

1. 表结构设计

SQL：

-- 学籍表

CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(50) NOT NULL

);

-- 课程表

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL

);

-- 成绩表

CREATE TABLE scores (

score\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

score INT,

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

1. 插入数据

-- 插入学生数据

INSERT INTO students (student\_id, student\_name) VALUES

(1, '张三'),

(2, '李四'),

(3, '王五'),

(4, '孙六');

-- 插入课程数据

INSERT INTO courses (course\_id, course\_name) VALUES

(1, '数据库'),

(2, '操作系统'),

(3, '编译原理'),

(4, '计算机网络');

-- 插入成绩数据

INSERT INTO scores (score\_id, student\_id, course\_id, score) VALUES

-- 数据库成绩

(1, 1, 1, 90),

(2, 2, 1, 80),

(3, 3, 1, 85),

(4, 4, 1, 70),

-- 操作系统成绩

(5, 1, 2, 55),

(6, 2, 2, 73),

(7, 3, 2, 66),

(8, 4, 2, 77),

-- 编译原理成绩

(9, 1, 3, 78),

(10, 2, 3, 87),

(11, 3, 3, 66),

(12, 4, 3, 56),

-- 计算机网络成绩

(13, 1, 4, 45),

(14, 2, 4, 68),

(15, 3, 4, NULL), -- 王五的计算机网络成绩为空

(16, 4, 4, 82);

1. 连接查询

SELECT

s.student\_name AS '姓名',

c.course\_name AS '课程',

sc.score AS '成绩'

FROM

scores sc

JOIN

students s ON sc.student\_id = s.student\_id

JOIN

courses c ON sc.course\_id = c.course\_id

ORDER BY

c.course\_id, s.student\_id;

1. 结果：



## 连表查询

SQL：

SELECT

s.student\_name AS '姓名',

MAX(CASE WHEN c.course\_name = '编译原理' THEN sc.score END) AS '编译原理',

MAX(CASE WHEN c.course\_name = '操作系统' THEN sc.score END) AS '操作系统',

MAX(CASE WHEN c.course\_name = '计算机网络' THEN sc.score END) AS '计算机网络',

MAX(CASE WHEN c.course\_name = '数据库' THEN sc.score END) AS '数据库'

FROM

students s

LEFT JOIN

scores sc ON s.student\_id = sc.student\_id

LEFT JOIN

courses c ON sc.course\_id = c.course\_id

GROUP BY

s.student\_name

ORDER BY

CASE s.student\_name

WHEN '李四' THEN 1

WHEN '孙六' THEN 2

WHEN '王五' THEN 3

WHEN '张三' THEN 4

END;

结果：  
