浙江大学管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

分布式文件系统及数据库技术
第四讲

主讲人：曹仔科 彭希羡
浙江大学管理学院
数据科学与工程管理系

浙江大学 管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

1、回顾：SQL查询

2、SQL进阶查询II

3、SQL数据更新

数据查询 – SELECT语句

```
SELECT [DISTINCT | ALL]
   {* | [columnExpression [AS newName]] [,...] }
FROM          TableName [alias] [, ...]
[WHERE        condition]
[GROUP BY     columnList]  [HAVING condition]
[ORDER BY     columnList]
```

数据查询 – SELECT语句

WHERE	筛选出满足条件的行
GROUP BY	将查询结果分组，具有相同属性值的行为同一组
HAVING	筛选出满足条件的组
ORDER BY	将查询结果进行排序

这些都是可选的

使用聚集函数+分组

ISO标准定义的五个聚集函数：

- **COUNT**：返回指定列中数据的个数
- **SUM**：返回指定列中数据的总和
- **AVG**：返回指定列中数据的平均值
- **MIN**：返回指定列中数据的最小值
- **MAX**：返回指定列中数据的最大值
- **GROUP BY**：对表中记录分组再进行操作

GROUP BY列的使用规则

- 当在SELECT语句中使用GROUP BY子句时，SELECT所选的列要么出现在聚集函数中，要么出现在GROUP BY子句中。

- GROUP BY子句中用到的列可以不出现在SELECT所选列表。

```
-- 按性别分组计算平均年龄
SELECT gender, AVG(age) AS avg_age
FROM Students
GROUP BY gender;
```

HAVING子句使用规则

- **HAVING子句一定是与GROUP BY配合使用的**
- **HAVING子句的意义在于实现组层面(group level)的筛选，行层面(record level)的筛选交给WHERE子句**
- **记住：一般情况下，HAVING子句的条件至少包含一个聚集运算，从而根据分组聚集运算的结果来进行筛选**
  - 否则可以将过滤条件移到WHERE子句中

浙江大学 管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

SQL进阶查询II

多表连接查询

- 目前使用SELECT语句只查询同一个表
- 很多情况下，需要将多个表的列合并在一个结果表，这需要**连接操作**
  - 连接操作：通过匹配相关行来合并两个表中的信息

Create Students和Course表

```
--创建学生信息表
CREATE TABLE Students (
    student_id INT IDENTITY (1,1) PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    gender VARCHAR(10),
    enrollment_date DATE,
    major VARCHAR(50),
    gpa DECIMAL(3, 2)
);
```

```
--创建课程表
CREATE TABLE Courses (
    course_id INT IDENTITY(1,1) PRIMARY KEY,
    course_name VARCHAR(50) NOT NULL,
    teacher VARCHAR(20) NOT NULL,
    credit INT NOT NULL
);
```

Students和Course表中数据

```
-- 插入学生数据
INSERT INTO Students (name, age, gender, enrollment_date, major, gpa) VALUES
('张三', 20, 'M', '2023-09-01', '信息管理与信息系统', 3.75),
('李四', 22, 'M', '2023-08-15', '商务大数据', 3.82),
('王芳', 19, 'F', '2023-09-10', '智能财务', 3.90),
('赵敏', 21, 'F', '2023-07-20', '计算机科学', 3.68),
('刘伟', 25, 'M', '2023-08-01', '信息管理与信息系统', 3.45),
('陈静', 18, 'F', '2023-09-05', '商务大数据', 3.95),
('周杰', 23, 'M', '2023-07-15', '智能财务', 3.60),
('吴婷', 20, 'F', '2023-08-25', '计算机科学', 3.78),
('郑浩', 22, 'M', '2023-09-01', '信息管理与信息系统', 3.85),
('孙莉', 19, 'F', '2023-08-10', '商务大数据', 3.72),
('钱程', 24, 'M', '2023-07-05', '智能财务', 3.55),
('马超', 21, 'M', '2023-08-20', '计算机科学', 3.88),
('林琳', 18, 'F', '2023-09-15', '信息管理与信息系统', 3.92),
('陈天', 20, 'M', NULL, '商务大数据', 3.65),
('欧阳明', 22, 'M', '2023-07-25', '智能财务', 3.80),
('麦东', NULL, 'F', '2023-09-17', '计算机科学', 3.70);
```

```
-- 插入课程数据
INSERT INTO Courses (course_name, teacher, credit) VALUES
('数据库原理', '王教授', 4),
('高等数学', '李教授', 6),
('数据结构', '张教授', 5),
('英语', '刘老师', 3),
('计算机网络', '陈教授', 4),
('算法分析', '赵教授', 5),
('计算机基础', '周老师', 2),
('人工智能导论', '吴教授', 4),
('软件工程', '郑教授', 5),
('编译原理', '孙教授', 4),
('操作系统', '钱教授', 5),
('Web开发', '马老师', 3),
('大数据分析', '林教授', 4);
```

代码中加入注释（comments）

- **单行注释：**
  - 单行注释用“--”开始，到该行结束

```
-- Select all:
SELECT * FROM Customers;
```

- **多行注释：**
  - 多行注释用“/*”开始，到“*/”结束

```
/*Select all the columns
of all the records
in the Customers table:*/
SELECT * FROM Customers;
```

创建的选课表Takes

```
CREATE TABLE Takes (
    student_id INT,
    course_id INT,
    grade INT,
    CONSTRAINT FK_takes_student
        FOREIGN KEY (student_id) REFERENCES Students(student_id),
    CONSTRAINT FK_takes_course
        FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

- 创建记录选课信息表Takes
- 设置适当的外键，即引用约束

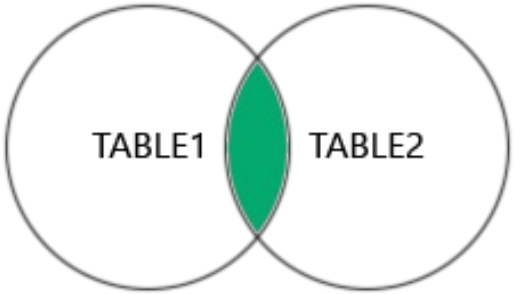
向选课表Takes插入一些数据

浙江大学 管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

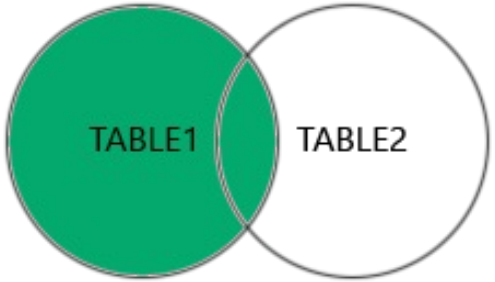
```
-- 插入选课数据
INSERT INTO Takes (student_id, course_id, grade) VALUES
-- 张三 (student_id=1)
(1, 1, 85),  -- 数据库原理
(1, 2, 90),  -- 高等数学
(1, 3, 78),  -- 数据结构
-- 李四 (student_id=2)
(2, 2, 92),  -- 高等数学
(2, 4, 88),  -- 英语
(2, 5, 76),  -- 计算机网络
-- 王芳 (student_id=3)
(3, 1, 95),  -- 数据库原理
(3, 6, 85),  -- 算法分析
(3, 7, 90),  -- 计算机基础
-- 赵敏 (student_id=4)
(4, 3, 82),  -- 数据结构
(4, 8, 88),  -- 人工智能导论
(4, 9, 92),  -- 软件工程
-- 刘伟 (student_id=5)
(5, 5, 78),  -- 计算机网络
(5, 10, 85), -- 编译原理
(5, 11, 90), -- 操作系统
-- 陈静 (student_id=6)
(6, 1, 92),  -- 数据库原理
(6, 12, 85), -- Web开发
(6, 13, 88);  -- 大数据分析
```

四种类型的连接

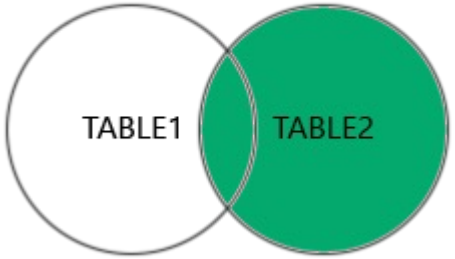
INNER JOIN



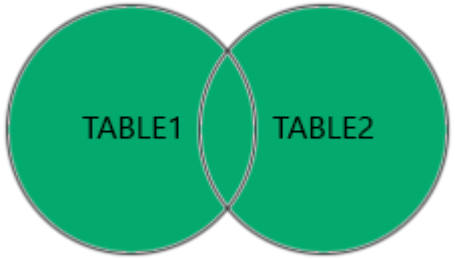
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



内连接查询

- 查询学生及其选修的课程成绩

```
SELECT s.name AS student_name, t. course_id, t.grade
FROM Students s
INNER JOIN Takes t ON s.student_id = t.student_id;
```

- 注意：必须在ON后边限定匹配条件（即通过哪一列来进行连接操作）

Where 语句写法：

```
SELECT s.name AS student_name, t. course_id, t.grade
FROM Students s, Takes t
Where s.student_id = t.student_id;
```

内连接查询

浙江大学管理学院 SCHOOL OF MANAGEMENT ZHEJIANG UNIVERSITY

```
-- 查询学生及其选修的课程信息
SELECT s.name AS student_name, c.course_name, t.grade
FROM Students s
INNER JOIN Takes t ON s.student_id = t.student_id
INNER JOIN Courses c ON t.course_id = c.course_id;
```

- 注意：涉及到三个表的连接，需要两个匹配条件。

Where 语句写法：

```
SELECT s.name, c.course_name, t.grade
FROM Students s, Takes t, Courses c
WHERE s.student_id = t.student_id AND t.course_id = c.course_id
```

带条件内连接查询

浙江大学管理学院 SCHOOL OF MANAGEMENT ZHEJIANG UNIVERSITY

- 内连接产生的表是一个新的表，可以加入SELECT语句的其他子句：WHERE，ORDER BY， GROUP BY，以及进行聚集运算。

```
-- 查询成绩大于85分的学生和课程信息
SELECT s.name, c.course_name, t.grade
FROM Students s
INNER JOIN Takes t ON s.student_id = t.student_id
INNER JOIN Courses c ON t.course_id = c.course_id
WHERE t.grade > 85;
```

仅用Where 语句怎么写呢?

左外连接查询

```
-- 查询所有学生及其选课情况（包括未选课的学生）
SELECT s.name, c.course_name, t.grade
FROM Students s
LEFT OUTER JOIN Takes t ON s.student_id = t.student_id
LEFT OUTER JOIN Courses c ON t.course_id = c.course_id;
```

- 左外连接： LEFT OUTER JOIN

左外连接查询


```
-- 如果只查询未选课的学生?
SELECT s.name
FROM Students s
LEFT OUTER JOIN Takes t ON s.student_id = t.student_id
WHERE t.student_id IS NULL;
```

- 没有选课记录的学生不能匹配选课表中记录，但会保留在左外连接查询结果中，选课表中的字段填充为NULL。

右外连接查询

```
-- 查询所有课程及其被选情况（包括未被选的课程）
SELECT c.course_name, s.name, t.grade
FROM Students s
RIGHT OUTER JOIN Takes t ON s.student_id = t.student_id
RIGHT OUTER JOIN Courses c ON t.course_id = c.course_id;
```

- 如果将第一个RIGHT OUTER JOIN改为INNER JOIN，会改变结果吗？为什么？

右外连接查询


```
-- 如果只查询没有被任何学生选的课程?
SELECT c.course_name
FROM Takes t
RIGHT OUTER JOIN Courses c ON t.course_id = c.course_id
WHERE t.course_id IS NULL;
```

- 没有选课记录的课程不能匹配选课表中记录，但会保留在右外连接查询结果中，选课表中的字段填充为NULL。

全外连接查询

```
-- 查询所有学生和所有课程的关联情况
SELECT s.name, c.course_name, t.grade
FROM Students s
FULL OUTER JOIN Takes t ON s.student_id = t.student_id
FULL OUTER JOIN Courses c ON t.course_id = c.course_id;
```

- 没有选课记录的课程或者没有选课的学生都会保留在全外连接查询结果中，缺失值的字段填充为NULL。

自连接

浙江大学管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

自连接：同一表与自己进行连接的特殊操作

```
--查询年龄相同的学生对：
SELECT s1.name AS student1, s2.name AS student2, s1.age
FROM Students s1
INNER JOIN Students s2 ON s1.age = s2.age AND s1.student_id <>
s2.student_id;
```

连接操作的简写

- INNER JOIN = JOIN
- LEFT OUTER JOIN = LEFT JOIN
- RIGHT OUTER JOIN = RIGHT JOIN
- FULL OUTER JOIN = FULL JOIN

子查询（嵌套查询）

- SELECT 语句嵌套是 SQL 中强大的功能，它允许在一个 SQL 查询语句中包含另一个完整的 SELECT 查询的结构。

- 嵌套查询可以出现在 SQL 语句的多个位置:
  - 在一个 **SELECT** 语句的 **FROM**、 **WHERE** 和 **HAVING** 子句中使用另一个 **SELECT**语句
  - 字段计算表达式中使用另一个**SELECT**语句
  - 在**INSERT, UPDATE**和**DELETE**语句中使用另一个**SELECT**语句

子查询的类型

标量子查询：子查询返回的结果是一个数据（一行一列）

列子查询：返回的结果是一列（一列多行）

行子查询：返回的结果是一行（一行多列）

表子查询：返回的结果是多行多列（多行多列）

清楚被嵌套的子查询的类型对选择合适的操作很关键。

标量子查询

- 前面已经见过标量子查询的简单应用（“交”集运算）。还可以将标量子查询作为返回结果中的一列：

```
-- 查询学生信息及其选修课程数量
SELECT
    s.student_id,
    s.name,
    s.age,
    (SELECT COUNT(*)
     FROM Takes t
     WHERE t.student_id = s.student_id) AS course_count
FROM Students s;
```

标量子查询

- 应用在WHERE子句中：

```
-- 查询选修了"数据库原理"课程的学生id
SELECT student_id
    FROM Takes
    WHERE course_id = (
        SELECT course_id
        FROM Courses
        WHERE course_name = '数据库原理'
    )
```

标量子查询

```
-- 查询每门课程的成绩及与平均分的差异
SELECT
    c.course_name,
    t.grade,
    (SELECT AVG(grade) FROM Takes WHERE course_id = c.course_id) AS course_avg,
    t.grade - (SELECT AVG(grade) FROM Takes WHERE course_id = c.course_id) AS diff_from_avg
FROM Takes t
JOIN Courses c ON t.course_id = c.course_id;
```

标量子查询

```
-- 查询高于平均成绩的学生选课记录
SELECT s.name, c.course_name, t.grade
FROM Students s
JOIN Takes t ON s.student_id = t.student_id
JOIN Courses c ON t.course_id = c.course_id
WHERE t.grade > (
    SELECT AVG(grade)
    FROM Takes
);
```

标量子查询：使用聚集函数

- **注意**：前面的查询不能写为：**WHERE t.grade > AVG(grade)**
  - 因为聚集函数**不能**直接用于**WHERE**子句中

- **正确方法**：先用子查询求出平均成绩，然后再用外部**SELECT**语句找出高于平均成绩的学生。

多行/列子查询—使用IN

浙江大学管理学院 SCHOOL OF MANAGEMENT ZHEJIANG UNIVERSITY

```
-- 查询选修了学分大于4的课程的学生
SELECT *
FROM Students
WHERE student_id IN (
    SELECT student_id
    FROM Takes
    WHERE course_id IN (
        SELECT course_id
        FROM Courses
        WHERE credit > 4));
```

派生表（在FROM子句中嵌套）

-- 查询每个有选课记录的学生的姓名及其平均成绩

```
SELECT s.name, t.avg_grade
FROM Students s
JOIN (
    SELECT student_id, AVG(grade) AS avg_grade
    FROM Takes
    GROUP BY student_id
) t ON s.student_id = t.student_id
```

HAVING子句中使用嵌套


```
-- 查询选修课程数量超过平均选课数的学生
SELECT s.student_id, s.name, COUNT(t.course_id) AS course_count
FROM Students s
JOIN Takes t ON s.student_id = t.student_id
GROUP BY s.student_id, s.name
HAVING COUNT(t.course_id) > (
    SELECT AVG(course_count)
    FROM (
        SELECT COUNT(course_id) AS course_count
        FROM Takes
        GROUP BY student_id
    ) sub
);
```

ALL与 ANY

- 关键词ALL与 ANY与生成数字类型的**多行/列子查询**一起使用：
- **ALL**：仅当子查询产生所有的值都满足条件，才为真
- **ANY**：当子查询产生的任何一个值满足条件，就为真

ALL与 ANY

-- 查询成绩高于所有课程平均分的学生

```
SELECT s.name, t.grade, t.course_id
FROM Students s
JOIN Takes t ON s.student_id = t.student_id
WHERE t.grade > ALL (
    SELECT AVG(grade)
    FROM Takes
    GROUP BY course_id
);
```

ALL与 ANY

```
-- 查询选修了任意学分大于4的课程的学生

SELECT DISTINCT s.name
FROM Students s
JOIN Takes t ON s.student_id = t.student_id
WHERE t.course_id = ANY (
    SELECT course_id
    FROM Courses
    WHERE credit > 4
);
```

子查询：EXISTS与NOT EXISTS

- 关键词**EXISTS**和**NOT EXISTS**仅能与一个子查询配合使用，返回结果为真或者假

- **EXISTS**返回真值当且仅当子查询返回的结果表至少存在一行(非空)
  - **NOT EXISTS**正好相反

- **EXISTS**和**NOT EXISTS**仅用于检查子查询结果表中是否存在行，所以子查询可返回任意结果
  - 一般可以用下列形式：**EXISTS (SELECT * FROM…)**

子查询：EXISTS与NOT EXISTS

```
-- 找出没有选修任何课程的学生
SELECT * FROM Students s
WHERE NOT EXISTS (
    SELECT *
    FROM Takes t
    WHERE t.student_id = s.student_id
)
```

上面的语句如果将NOT EXISTS改为EXISTS，那就是查询至少选修了一门课程的学生。

CTE（Common Table Expression，公共表表达式）

- CTE（Common Table Expression，公共表表达式）是 SQL 中的一种**临时命名结果集**，它可以在一个查询中被引用多次。CTE 使用 WITH关键字定义，并在紧随其后的 SELECT、INSERT、UPDATE 或 DELETE语句中可见。
- 临时性：CTE 只在查询执行期间存在
- 可重用性：在同一查询中可以多次引用同一个 CTE
- 递归能力：支持递归查询（递归 CTE）

CTE（Common Table Expression，公共表表达式）

```
WITH cte_name (column1, column2, ...)
AS
(
    -- CTE 查询定义
    SELECT ...
)
-- 使用 CTE 的主查询
SELECT * FROM cte_name;
```

CTE示例


```
-- 使用 CTE 实现复杂集合操作
WITH
MathStudents AS (
    SELECT student_id FROM Takes
    WHERE course_id = (SELECT course_id FROM Courses WHERE course_name = '高等数学')
),
CSStudents AS (
    SELECT student_id FROM Takes
    WHERE course_id = (SELECT course_id FROM Courses WHERE course_name = '计算机基础')
)
-- 查询选修了高等数学但没有选修计算机基础的学生
SELECT * FROM MathStudents
EXCEPT
SELECT * FROM CSStudents;
```

子查询：一些使用规则

- 子查询SELECT语句中不能使用ORDER BY子句

- 当子查询是比较表达式中的一个操作数时，子查询必须出现在表达式的右面
  - **WHERE grade > (SELECT AVG(grade) FROM Takes);** (正确)
  - **WHERE (SELECT AVG(grade) FROM Takes) < grade;** (错误)

两个表的并、交与差

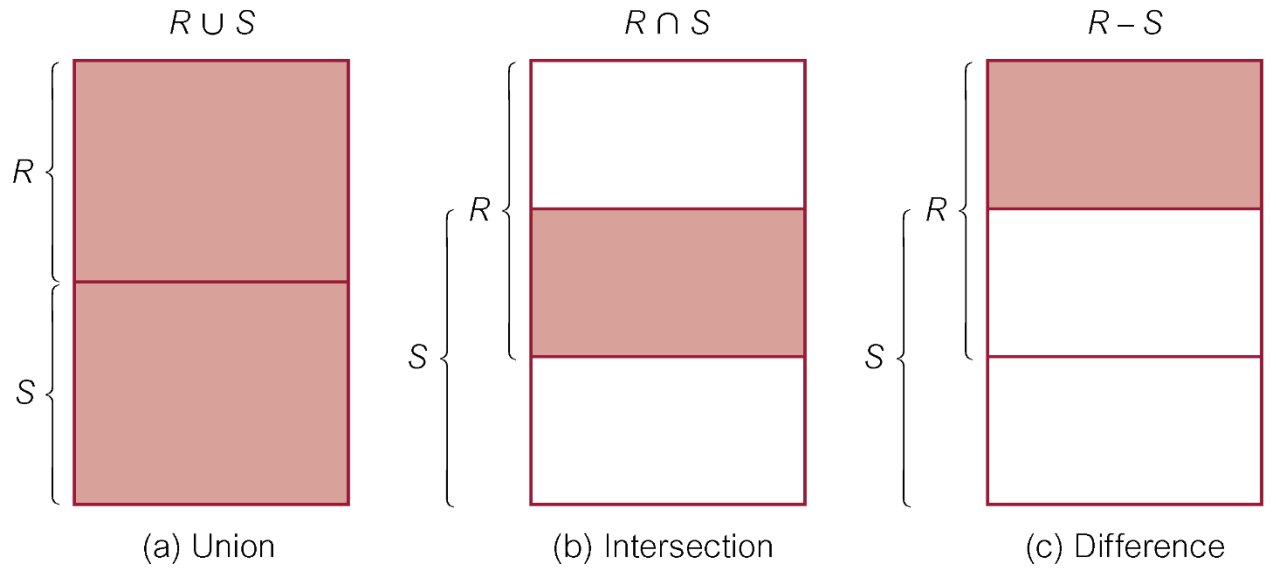
- **并（UNION）**：A和B两个表的并操作结果是一个包含两个表中所有行的表
- **交（INTERSECT）**：A和B两个表的交操作结果是一个包含两个表中共同行的表
- **差（EXCEPT）**：A和B两个表的差操作(**A EXCEPT B**)的结果是一个包含那些在A中而不在B中的行的表

- 注意：两个表必须具有**并相容性**

两个表的并、交与差

Table R *operator* [ALL] Table S

- ***operator*: UNION, INTERSECT, EXCEPT**
- 如果指定了ALL，则结果包括一切重复的行



并

```
-- 获取所有学生和教师的姓名(去重)
SELECT name FROM Students
UNION
SELECT teacher FROM Courses;
```

```
-- 获取所有学生和教师的姓名(包含重复)
SELECT name FROM Students
UNION ALL
SELECT teacher FROM Courses;
```

交

```
-- 查询选修了"数据库原理"和"数据结构"课程的学生

SELECT student_id FROM Takes
WHERE course_id = (SELECT course_id FROM Courses WHERE
  course_name = '数据库原理')
INTERSECT
SELECT student_id FROM Takes
WHERE course_id = (SELECT course_id FROM Courses WHERE
  course_name = '数据结构');
```

差

-- 查询没有选修任何课程的学生：

```
SELECT student_id FROM Students
EXCEPT
SELECT student_id FROM Takes;
```

浙江大学 管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

SQL数据更新

SQL数据操作

- **INSERT**
  - 向表中添加新的行
- **UPDATE**
  - 修改表中现有的行
- **DELETE**
  - 删除表中已有的行

数据插入 – INSERT … VALUES

INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

- 列名columnList可以忽略。如果忽略，SQL会严格按照表格创建时的顺序执行插入
- columnList和dataValueList数目必须相同，各项必须直接对应
- 数据类型必须一致

从查询结果插入

```
-- 从现有学生创建新班级（示例）

INSERT INTO NewClass (student_name, age)
SELECT name, age
FROM Students
WHERE enrollment_date > '2023-09-01';
```

从查询结果插入

```
--统计课程的学科情况（假设已建Course_statistics表）
INSERT INTO Course_statistics (course_id, course_name, counts)
SELECT c.course_id, c.course_name, COUNT(t.student_id)
FROM Courses c, Takes t
WHERE t.course_id = c.course_id
GROUP BY c.course_id, c.course_name;
```

是否正确?

从查询结果插入

```
--统计课程的学科情况（假设已建Course_statistics表）
INSERT INTO Course_statistics (course_id, course_name, counts)
SELECT c.course_id, c.course_name, COUNT(*)
FROM Courses c, Takes t
WHERE t.course_id = c.course_id
GROUP BY c.course_id, c.course_name;
```

是否正确?

从查询结果插入

```
--统计课程的学科情况（假设已建Course_statistics表）
SELECT c.course_id, c.course_name, COUNT(*)
FROM Takes t
RIGHT OUTER JOIN Courses c ON t.course_id = c.course_id
GROUP BY c.course_id, c.course_name
```

是否正确?

UPDATE

```
UPDATE TableName
SET columnName1 = dataValue1 [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- SET子句指定需要更新的一个或多个列的名字
- WHERE子句限定需要更新的行；如果WHERE子句省略，则对给定列的所有行进行更新

UPDATE

```
-- 更新单个学生的年龄
UPDATE Students
SET age = 23
WHERE name = '李四';

-- 更新学生信息和入学日期
UPDATE Students
SET age = 24, enrollment_date = '2023-08-15'
WHERE name = '刘伟';
```

使用JOIN来UPDATE记录

```
-- 更新所有女生的成绩（提高3%）
UPDATE t
SET grade = grade * 1.03
FROM Takes t
JOIN Students s ON t.student_id = s.student_id
WHERE s.gender = 'F';
```

DELETE

DELETE FROM TableName
[WHERE searchCondition]

- **WHERE**子句限定要进行删除操作的行；如果**WHERE**子句省略，则所有行都会被删除
- 若要既删除内容又删除表定义，可用**DROP TABLE**语句

DELETE

```
-- 删除没有选修任何课程的学生
DELETE FROM Students
WHERE student_id NOT IN (
    SELECT student_id
    FROM Takes
);
```

结合JOIN来DELETE记录

```
-- 删除所有"计算机基础"课程的选课记录
DELETE t
FROM Takes t
JOIN Courses c ON t.course_id = c.course_id
WHERE c.course_name = '计算机基础';
```

浙江大学管理学院
SCHOOL OF MANAGEMENT
ZHEJIANG UNIVERSITY

谢谢！

下次见！

AACSB ACCREDITED
EFMD EQUIS ACCREDITED
ASSOCIATION AMBA ACCREDITED
CAMEA 中国高质量MBA教育认证
IQA CEEMAN Accredited