

4

## SQL与可编程对象

## 4.1.4 数据库操作语言 (DML) ——多表查询

## 引例——查询所有学生的选课、成绩及生源地情况

**分析：**要查询的信息来自两个数据表，  
一个是提供生源地信息的student表，  
一个是提供选课信息的courseenroll表。

查询信息涉及多个数据表！

Student表

	StudentCode	StudentName	Gender	Birthday	Photo	Email	Phone	Location
1	1,001	杜斯	女	1998-11-05 00:00:00	1000.jpg	1001@elearning.com	13358301911	上海
2	1,002	汪洋	男	1999-11-09 00:00:00	1001.jpg	1002@elearning.com	13834992564	北京
3	1,003	林豆豆	女	1997-12-15 00:00:00	1002.jpg	1003@elearning.com	13243964904	江苏
4	1,004	张小贝	男	2005-01-19 00:00:00	1003.jpg	1004@elearning.com	13142540852	山东
5	1,005	马林	男	1985-11-09 00:00:00	1004.jpg	1005@elearning.com	13971113836	浙江
6	1,006	张小宝	男	1992-07-01 00:00:00	1005.jpg	1006@sina.cn	13713143513	黑龙江
7	1,007	宋思明	男	1995-10-02 00:00:00	1006.jpg	1007@sina.cn	13916801546	江西
8	1,008	徐米拉	女	2000-03-30 00:00:00	1007.jpg	1008@sina.cn	13225414354	湖北
9	1,009	杨康	男	2001-11-09 00:00:00	1008.jpg	1009@sina.cn	13345482022	上海
10	1,010	郭靖	男	1993-04-05 00:00:00	1009.jpg	1010@sina.cn	13974339695	北京
11	1,011	朱丽雯	女	1997-06-02 00:00:00	1010.jpg	1011@126.com	13167420737	新疆
12	1,012	王思云	女	1976-09-15 00:00:00	1011.jpg	1012@126.com	13343234010	江苏

courseenroll表

123	StudentCode	123	CourseCode	123	Score	123	TestTime
	1,001		C001		72		2018-02-20 14:15:08
	1,001		C002		65		2017-12-15 09:20:11
	1,001		C003		93		2018-02-20 14:15:08
	1,001		C005		[NULL]		[NULL]
	1,002		C001		70		2017-07-21 16:21:40
	1,002		C002		88		2018-06-22 09:15:57
	1,002		C003		55		2017-08-15 09:20:33
	1,002		C004		[NULL]		[NULL]
	1,002		C005		[NULL]		[NULL]
	1,003		C001		75		2017-11-01 15:55:01
	1,003		C002		90		2018-01-10 10:55:18
	1,003		C003		76		2018-06-22 09:16:20
	1,003		C005		[NULL]		[NULL]



## 多表连接查询

涉及多表的查询，要说明表之间的连接关系。

学生的选课、成绩及生源地的部分查询结果

123 StudentCode	ABC StudentName	ABC CourseCode	123 Score	ABC Location
1,001	杜斯	C001	72	上海
1,001	杜斯	C002	65	上海
1,001	杜斯	C003	93	上海
1,001	杜斯	C005	[NULL]	上海
1,002	汪洋	C001	70	北京
1,002	汪洋	C002	88	北京
1,002	汪洋	C003	55	北京
1,002	汪洋	C004	[NULL]	北京
1,002	汪洋	C005	[NULL]	北京
1,003	林豆豆	C001	75	江苏
1,003	林豆豆	C002	90	江苏
1,003	林豆豆	C003	76	江苏

来自student表的字段

问题：如何连接多个表？

分析：可以理解成，先查询student表中的记录，对于每一条记录，在courseenroll表中找到该记录的学号对应的选课记录。

学号（StudentCode）是两个表的**公共属性**。公共属性是连接的桥梁。

多表连接把来自多个表的字  
段组合在一起（列扩展）

来自courseenroll表的字段



## 多表连接查询——用**WHERE**子句实现内连接

```
SELECT 字段列表 FROM 表1, 表2  
WHERE 表1.字段名1 <比较运算符> 表2.字段名2;
```

**比较运算符:** =、<、>、<=、>=、<>(或 ^= 或 !=)

**字段名1和字段名2:** 必须同类型，但名称可不同。

## 用WHERE子句实现内连接

例如：查询所有学生的选课、成绩及生源地情况。

来自courseenroll表

来自student表

两个表中都有  
StudentCode这个字段

StudentCode 前需要限定表名

```
SELECT Student.StudentCode, StudentName, CourseCode, Score, Location  
FROM courseenroll, Student  
WHERE courseenroll.StudentCode=student.StudentCode;
```

student和courseenroll用StudentCode相等连接

	StudentCode	StudentName	CourseCode	Score	Location
1	1,001	杜斯	C001	72	上海
2	1,001	杜斯	C002	65	上海
3	1,001	杜斯	C003	93	上海
4	1,001	杜斯	C005	[NULL]	上海
5	1,002	汪洋	C001	70	北京
6	1,002	汪洋	C002	88	北京
7	1,002	汪洋	C003	55	北京
8	1,002	汪洋	C004	[NULL]	北京
9	1,002	汪洋	C005	[NULL]	北京
10	1,003	林豆豆	C001	75	江苏
11	1,003	林豆豆	C002	90	江苏
12	1,003	林豆豆	C003	76	江苏

## 用WHERE子句实现内连接

例如：查询各学科开设的课程，显示学科号、学科名和课程名。

来自course表

两个表中都有  
SubjectCode字段。

来自subject表

SubjectCode 前需要限定表名

```
SELECT subject.SubjectCode , SubjectName, CourseName  
FROM subject, course  
WHERE subject.SubjectCode=course.SubjectCode;
```

subject和course用SubjectCode相等连接

	ABC SubjectCode	ABC SubjectName	ABC CourseName
1	S01	管理学	多媒体技术及应用
2	S01	管理学	信息系统与数据库技术
3	S01	管理学	计算机网络技术
4	S03	心理学	管理心理学
5	S03	心理学	心理学与生活
6	S03	心理学	感觉与知觉
7	S03	心理学	沟通心理学
8	S01	管理学	操作系统
9	S06	理学	高等数学微积分
10	S06	理学	线性代数
11	S07	文学	外国文学史
12	S07	文学	唐诗经典



## 多表连接查询 —— 使用**JOIN ... ON**实现**内连接**

**SELECT** 字段列表

INNER关键字  
可以省略

**FROM** 表1 **[INNER]** **JOIN** 表2

**ON** 表1.字段名1 <比较运算符> 表2.字段名2;

比较运算符： =、 <、 >、 <=、 >=、 <>(!=或^=)

**字段名1和字段名2：** 必须同类型，但名称可不同。

## 用JOIN ... ON实现内连接

例如：使用关键词JOIN...ON查询各学科开设的课程，  
显示学科号、学科名和课程名。

↓                    ↓  
来自subject表      来自course表

```
SELECT subject.SubjectCode, SubjectName, CourseName  
FROM subject JOIN course  
ON subject.SubjectCode = course.SubjectCode;
```

	ABC SubjectCode ↕	ABC SubjectName ↕	ABC CourseName ↕
1	S01	管理学	多媒体技术及应用
2	S01	管理学	信息系统与数据库技术
3	S01	管理学	计算机网络技术
4	S03	心理学	管理心理学
5	S03	心理学	心理学与生活
6	S03	心理学	感觉与知觉
7	S03	心理学	沟通心理学
8	S01	管理学	操作系统
9	S06	理学	高等数学微积分
10	S06	理学	线性代数
11	S07	文学	外国文学史
12	S07	文学	唐诗经典

## 理解内连接

例如：查询各学科开设的课程，显示学科号、学科名和课程名。

实现方法1（使用WHERE的查询条件）：

```
SELECT subject.SubjectCode, SubjectName, CourseName  
FROM subject, course  
WHERE subject.SubjectCode=course.SubjectCode;
```

实现方法2（使用JOIN...ON关键字）：

```
SELECT subject.SubjectCode, SubjectName, CourseName  
FROM subject JOIN course  
ON subject.SubjectCode = course.SubjectCode;
```

对于“=”（相等关系），只有SubjectCode的某个取值在两个表中都出现，该取值对应的记录才会出现在最后的查询结果中。

	RBC SubjectCode	RBC SubjectName	RBC CourseName
1	S01	管理学	多媒体技术及应用
2	S01	管理学	信息系统与数据库技术
3	S01	管理学	计算机网络技术
4	S03	心理学	管理心理学
5	S03	心理学	心理学与生活
6	S03	心理学	感觉与知觉
7	S03	心理学	沟通心理学
8	S01	管理学	操作系统
9	S06	理学	高等数学微积分
10	S06	理学	线性代数
11	S07	文学	外国文学史
12	S07	文学	唐诗经典

问题：

- 1.两个表的位置能不能交换？
- 2.如何理解多表连接过程？
  - 查询subject表中的记录，
  - 对于每一条记录 $s_i$ ，在course表中查找该记录的SubjectCode对应的记录，
  - 若在course表中找到相应记录 $c_k$ ，那么将这两条记录 $s_i, c_k$ 中的字段组合成一条记录且这条记录出现在最后的查询结果中，否则不出现。



## 多表连接查询 —— 外连接

**SELECT** 字段列表

**FROM 左表 LEFT|RIGHT [OUTER] JOIN 右表**

**ON 左表.字段名1 <比较运算符> 右表.字段名2**

比较运算符： =、 <、 >、 <=、 >=、 <>(!=或^=)

字段名1和字段名2：必须同类型，但名称可不同。

**LEFT [OUTER] JOIN**(左外连接)：包含**左表全部记录，右表匹配**

**RIGHT [OUTER] JOIN**(右外连接)：包含**右表全部记录，左表匹配**

问题：

1.两个表的位置能不能交换？

2.如何理解左外联接过程？

- 查询左表中的记录，
- 对于左表每一条记录 $s_i$ ，在右表中查找与该记录匹配的记录，
- 若在右表中找到匹配的记录 $c_k$ ，那么将这两条记录 $s_i, c_k$ 中的字段组合成一条记录 $m_t$ ，添加到查询结果，
- 若在右表中**没有**找到匹配的记录，那么认为来自右表中的字段均为 NULL，组合成记录 $m_t$ ，添加到查询结果。

左外连接时，左表中的所有记录都会出现在最后的查询结果中。

## 内连、左外、右外连举例

例如：分别使用内连接、左外连接、右外连接查询女生选课信息。

```
SELECT student.StudentCode, StudentName, CourseCode, Score  
FROM student JOIN courseenroll  
ON student.StudentCode = courseenroll.StudentCode  
WHERE Gender='女';
```

查询所有的女生选课情况。

```
SELECT student.StudentCode, StudentName, CourseCode, Score  
FROM student LEFT JOIN courseenroll  
ON student.StudentCode = courseenroll.StudentCode  
WHERE Gender='女';
```

查询所有女生学号和姓名，并显示她们的选课信息。(没选课的女生信息也显示)

```
SELECT student.StudentCode, StudentName, CourseCode, Score  
FROM student RIGHT JOIN courseenroll  
ON student.StudentCode = courseenroll.StudentCode  
WHERE Gender='女';
```

查询选课表中所有女生的选课信息，并显示她们的学号和姓名。

# 内连、左外、右外连举例

例如：分别使用内连接、左外连接、右外连接查询女生选课信息。

左表（表1）是student表，右表（表2）是courseenroll表；

JOIN..... ON

	StudentCode	StudentName	CourseCode	Score
1	1,001	杜斯	C001	72
2	1,001	杜斯	C002	65
3	1,001	杜斯	C003	93
4	1,001	杜斯	C005	[NULL]
5	1,003	林豆豆	C001	75
6	1,003	林豆豆	C002	90
7	1,003	林豆豆	C003	76
8	1,003	林豆豆	C005	[NULL]
9	1,003	林豆豆	C012	[NULL]

9行 - 4ms [16:01:58]

LEFT JOIN..... ON

	StudentCode	StudentName	CourseCode	Score
4	1,001	杜斯	C005	[NULL]
5	1,003	林豆豆	C001	75
6	1,003	林豆豆	C002	90
7	1,003	林豆豆	C003	76
8	1,003	林豆豆	C005	[NULL]
9	1,003	林豆豆	C012	[NULL]
10	1,008	徐米拉	[NULL]	[NULL]
11	1,011	朱丽雯	[NULL]	[NULL]
12	1,012	王思云	[NULL]	[NULL]

21行 - 4ms [16:04:34]

RIGHT JOIN..... ON

	StudentCode	StudentName	CourseCode	Score
1	1,001	杜斯	C001	72
2	1,001	杜斯	C002	65
3	1,001	杜斯	C003	93
4	1,001	杜斯	C005	[NULL]
5	1,003	林豆豆	C001	75
6	1,003	林豆豆	C002	90
7	1,003	林豆豆	C003	76
8	1,003	林豆豆	C005	[NULL]
9	1,003	林豆豆	C012	[NULL]

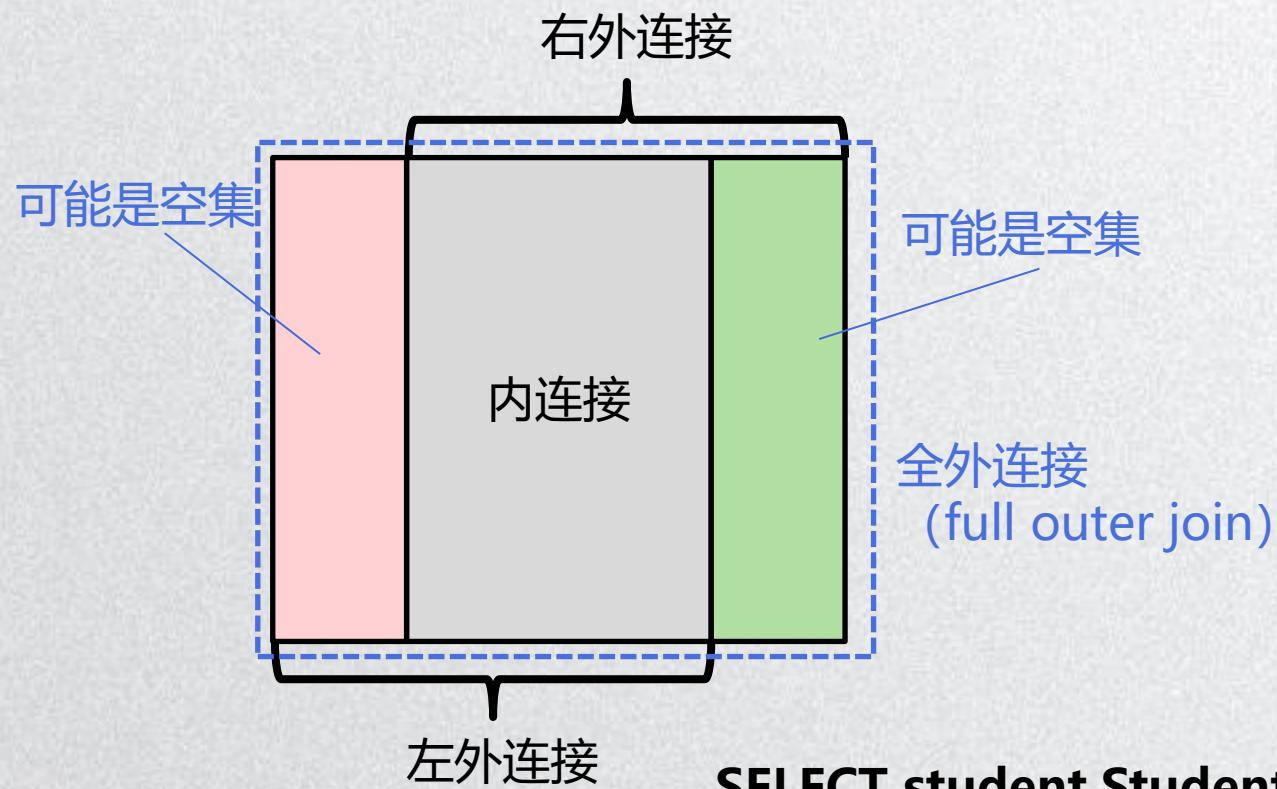
9行 - 3ms [16:49:53]

左外连接：包含**左表全部记录，右表匹配**

右外连接：包含**右表全部记录，左表匹配**

问题：思考三种连接之间的查询结果有什么不同？有何关系？

## 内连、左外、右外连的查询结果之间的关系



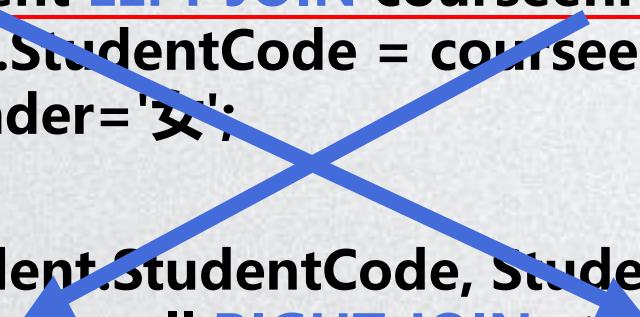
左表 (表1) : student  
右表 (表2) : courseenroll

问题：图中的粉色部分和绿色部分分别对应哪些记录？

```
SELECT student.StudentCode, StudentName, CourseCode, Score
FROM student JOIN/LEFT JOIN/RIGHT JOIN courseenroll
ON student.StudentCode = courseenroll.StudentCode
WHERE Gender='女';
```

## 左外连和右外连的对应关系

```
SELECT student.StudentCode, StudentName, CourseCode, Score  
FROM student LEFT JOIN courseenroll  
ON student.StudentCode = courseenroll.StudentCode  
WHERE Gender='女';
```



```
SELECT student.StudentCode, StudentName, CourseCode, Score  
FROM courseenroll RIGHT JOIN student  
ON student.StudentCode = courseenroll.StudentCode  
WHERE Gender='女';
```



注意：查询结果记录集是相同的。显示查询结果时，记录的顺序不一定相同。



## 多表连接查询——3个以上表连接

3个数据表以上形成**多表连接查询**。

```
SELECT 字段列表  
FROM 表1 JOIN 表2 ON 表1.字段 i <比较运算符> 表2.字段 j  
      JOIN 表3 ON 表x.字段k <比较运算符> 表3.字段 l  
      ...  
      [JOIN 表z ON 表y.字段m <比较运算符> 表z.字段n];
```

x<=2  
y<=z-1

```
SELECT StudentName, CourseName, Score  
FROM student JOIN courseenroll  
      ON student.StudentCode = courseenroll.StudentCode  
      JOIN course ON courseenroll.CourseCode = course.CourseCode  
WHERE Gender='女';
```

## 3个以上表连接查询举例

来自subject表

来自student表

来自courseenroll表

来自course表

例如：查询选修了学科名为“心理学”相关课程的学生名、课程名和成绩。

	StudentName	CourseName	Score	SubjectName
1	汪洋	管理心理学	[NULL]	心理学
2	马林	管理心理学	[NULL]	心理学
3	杜斯	心理学与生活	[NULL]	心理学
4	汪洋	心理学与生活	[NULL]	心理学
5	林豆豆	心理学与生活	[NULL]	心理学
6	马林	心理学与生活	[NULL]	心理学
7	郭靖	沟通心理学	80	心理学

```
SELECT StudentName,CourseName,Score,SubjectName  
FROM student JOIN courseenroll  
    ON student.StudentCode=courseenroll.StudentCode  
    JOIN course ON courseenroll.CourseCode=course.CourseCode  
    JOIN subject ON course.SubjectCode=subject.SubjectCode  
WHERE SubjectName='心理学';
```



## 子查询

当一个查询结果是另一个查询的条件时，称该查询为**子查询**。  
整个语句称为嵌套查询。

例如：查询与“章咪咪”来自同一地区的学生的学号和姓名。

	StudentCode	StudentName	Location
1	1,002	汪洋	北京
2	1,010	郭靖	北京
3	1,017	章咪咪	北京

```
SELECT StudentCode,StudentName,Location  
FROM Student  
WHERE Location=  
(SELECT Location  
FROM Student  
WHERE StudentName='章咪咪');
```

如果子查询结果返回多个值，主查询的WHERE子句可使用集合判断。

## WHERE 字段 IN 子查询：字段是否在子查询结果中

例如：查询选修了课程代码为“C001”的学生的学号和姓名。

```
SELECT StudentCode,StudentName  
FROM Student  
WHERE StudentCode IN  
(SELECT StudentCode  
FROM courseenroll  
WHERE CourseCode='C001') ;
```

括号中的语句的查询结果是一个集合。

	StudentCode	StudentName
1	1,001	杜斯
2	1,002	汪洋
3	1,003	林豆豆
4	1,004	张小贝
5	1,005	马林

## IN子查询

---

IN子查询通常也可以利用连接查询完成。

```
SELECT student.StudentCode,StudentName  
FROM student JOIN courseenroll  
ON student.StudentCode=courseenroll.StudentCode  
WHERE CourseCode= 'C001';
```

	StudentCode	StudentName
1	1,001	杜斯
2	1,002	汪洋
3	1,003	林豆豆
4	1,004	张小贝
5	1,005	马林

## IN子查询

子查询可以计算一个变化的聚集函数值返回给主查询。

例如：查询年龄最大的学生的学号和姓名。

```
SELECT StudentCode, StudentName, Birthday  
FROM Student  
WHERE Birthday IN  
    (SELECT MIN(Birthday) FROM Student);
```

-- 或

```
WHERE Birthday = (SELECT distinct MIN(Birthday) FROM Student);
```

	StudentCode	StudentName	Birthday
1	1,015	刘文强	1963-12-04

年龄最大学生

## EXISTS子查询

**WHERE EXISTS (子查询): 子查询的结果集是否为空，非空返回True，空返回False**

例如：查询所有未选修任何课程的学生。

```
SELECT StudentCode, StudentName  
FROM student  
WHERE NOT EXISTS  
(SELECT * FROM courseenroll  
WHERE student.StudentCode= courseenroll.StudentCode);
```

	StudentCode	StudentName
1	1,006	张小宝
2	1,007	宋思明
3	1,008	徐米拉

21 行 - 2ms

扫描student表中每一条记录s，查询该记录s中的StudentCode在courseenroll表中是否有对应的记录：

- 如果有对应的记录，括号中的查询语句返回True，那么NOT EXISTS(...)部分返回False，student表中的该条记录s不会出现在最终的查询结果中。
- 如果没有对应的记录，括号中的查询语句返回False，那么NOT EXISTS(...)部分返回True，student表中的该条记录s会出现在最终的查询结果中。

继续扫描student下一条记录直至末尾。

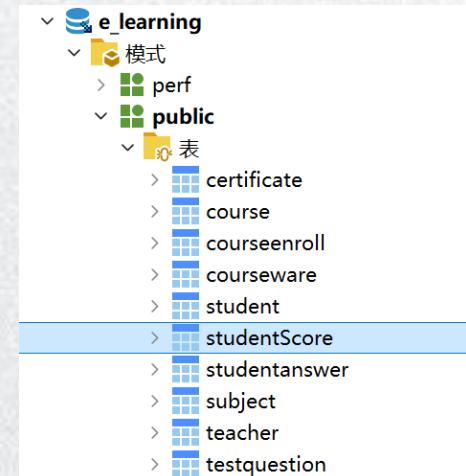


## 存储结果集到新表

将一个查询的结果存入新表，可通过在SELECT语句中使用INTO子句来实现。

```
SELECT 新表的字段列表  
INTO 新表名  
FROM 源表名  
WHERE 查询条件
```

例 查询student表，将查询的学生学号、姓名、课程名、成绩存入studentScore表中。



查看studentScore表数据  
select \* from studentScore;

	StudentCode	StudentName	courseName	score
1	1,001	杜斯	多媒体技术及应用	72
2	1,001	杜斯	信息系统与数据库技术	65
3	1,001	杜斯	计算机网络技术	93
4	1,001	杜斯	心理学与生活	[NULL]
5	1,002	汪洋	多媒体技术及应用	70
6	1,002	汪洋	信息系统与数据库技术	88
7	1,002	汪洋	计算机网络技术	55

```
SELECT courseenroll.StudentCode,StudentName,courseName,score  
INTO studentScore  
FROM student,courseenroll,course  
WHERE student.StudentCode = courseenroll.StudentCode AND  
course.CourseCode = courseenroll.CourseCode ;
```



## 小结：多表连接查询

**SELECT** 字段列表

**FROM** 表1 **连接关键字** 表2

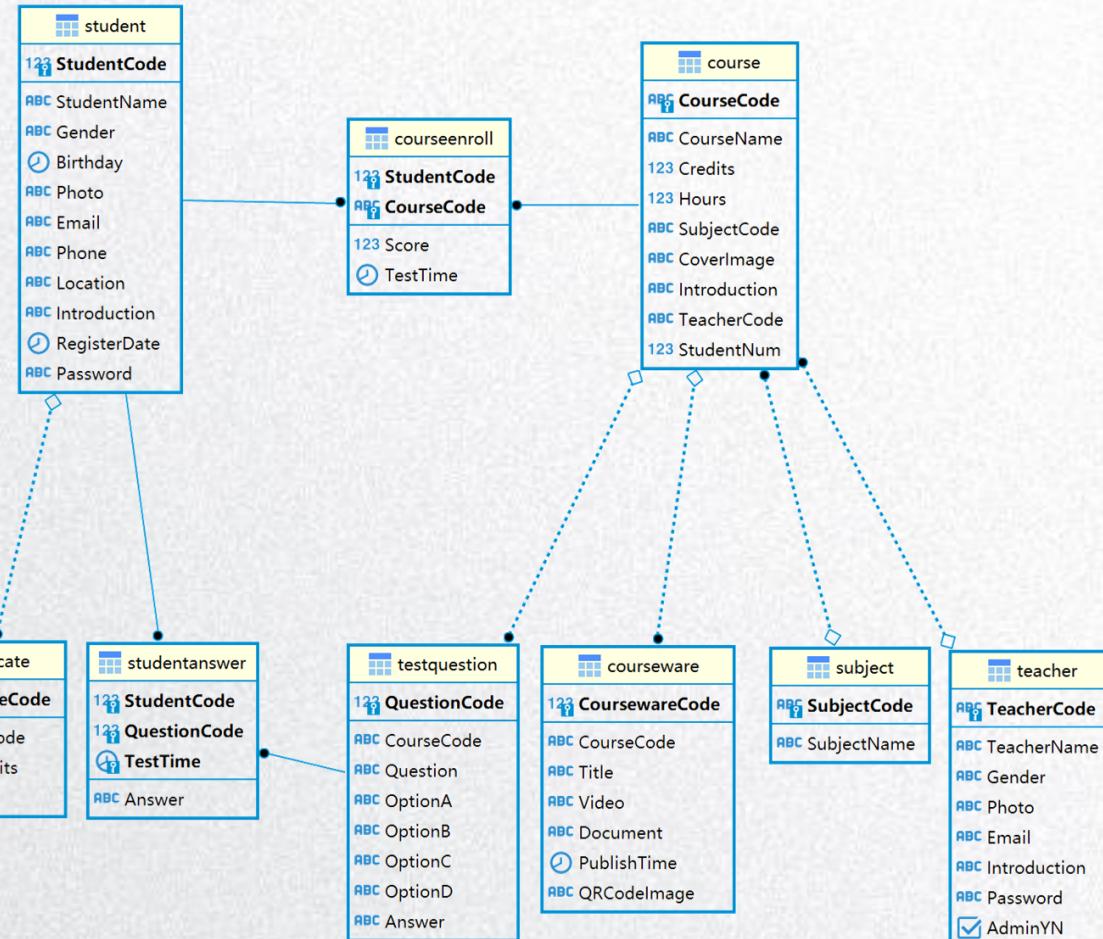
**ON** 表1.字段名1 <比较运算符> 表2.字段名2

**SELECT student.StudentCode, StudentName, CourseCode, Score**

**FROM student JOIN/LEFT JOIN/RIGHT JOIN courseenroll**

**ON student.StudentCode = courseenroll.StudentCode**

**WHERE Gender='女';**



# **THANK YOU!**

## 4.1.5 数据更新语句 ——INSERT          UPDATE          DELETE



# 数据操纵语言DML

实现对数据的查询和更新。

## 常用语句

语句	功能
SELECT	从表中查询数据
INSERT	向表中添加记录
UPDATE	修改表中数据
DELETE	从表中删除记录



## 插入语句INSERT INTO

向表中插入一条数据记录。

**INSERT INTO 表名[(字段名1, 字段名2, .....)]**

**VALUES (表达式1, 表达式2, .....);**

- 表达式和字段一一对应
- 必填字段必须在字段列表中，其他值为Null
- 插入完整记录可省略字段名，但字段值次序要与表一致

必填字段：非空且  
没有默认值的字段

# INSERT INTO插入单条记录

例如：向subject表中插入一条记录，学科号为S10、学科名为“哲学”。

**INSERT INTO subject VALUES ('S10', '哲学');**

The screenshot shows a SQL client interface with a query window containing the following code:

```
1 INSERT INTO subject
2   VALUES ('S10', '哲学');
```

Below the query window is a statistics panel titled "Statistics 1". It displays the following information:

Name	Value
Updated Rows	1

Under the "Query" section, it shows the executed query and the finish time:

Query: `INSERT INTO subject  
VALUES ('S10', '哲学')`  
Finish time: Sat Feb 08 09:54:13 CST 2025

At the bottom, a message indicates: `1 row(s) updated - 3ms [09:54:13]`.

	SubjectCode	SubjectName
1	S01	管理学
2	S02	经济学
3	S03	心理学
4	S04	艺术
5	S06	理学
6	S07	文学
7	S08	工程
8	S09	法学

	SubjectCode	SubjectName
1	S01	管理学
2	S02	经济学
3	S03	心理学
4	S04	艺术
5	S06	理学
6	S07	文学
7	S08	工程
8	S09	法学
9	S10	哲学

## INSERT INTO插入单条记录

例如：向student表中插入一条记录。

```
INSERT INTO student(StudentCode, Gender, StudentName, Birthday, Email)  
VALUES (1099, '女', '张琳', '1998-1-13', 'zl@163.com');
```

StudentCode	StudentName	Gender	Birthday	Photo	Email	Phone	Location	Introduction	RegisterDate	Password
1,099	张琳	女	1998-01-13	[NULL]	zl@163.com	[NULL]	[NULL]	[NULL]	2025-02-08 09:59:18	666666

往student表中插入数据时，非必填字段例可以不提供值。  
这段字段要么为空，要么是默认值。

# INSERT INTO插入多条记录

一次向表中插入多条数据记录。

**INSERT INTO 表名[(字段名1, 字段名2, .....)]**

**VALUES (表达式1, 表达式2, .....),**

**(表达式1, 表达式2, .....),**

**...**

**(表达式1, 表达式2, .....);**

每条数据记录  
之后用逗号，  
句尾用分号

```
INSERT INTO student(StudentCode, Gender, StudentName, Birthday, Email)
VALUES (1101, '女', '张小琳', '2005-4-7', 'zxl@163.com'),
(1102, '男', '王涛', '1999-1-21', 'wangtao@126.com'),
(1103, '女', '白冰冰', '2004-7-31', 'baibingbing@163.com');
```

Statistics 1	
SQL	INSERT INTO student(StudentCode, Gender, Stude
输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)	
<input checked="" type="checkbox"/> 保存	<input type="button" value="取消"/>
<input type="button" value="脚本"/>	<input type="button" value="网格"/>
<input type="button" value="文本"/>	<input type="button" value="文本"/>
<input type="button" value="记录"/>	<input type="button" value="面板"/>
Name	Value
Updated Rows	3
Query	INSERT INTO student(StudentCode, Gender, StudentName, Birthday, Email) VALUES (1101, '女', '张小琳', '2005-4-7', 'zxl@163.com'), (1102, '男', '王涛', '1999-1-21', 'wangtao@126.com'), (1103, '女', '白冰冰', '2004-7-31', 'baibingbing@163.com');
Finish time	Tue Mar 18 13:26:33 CST 2025

# INSERT INTO插入多条记录

例如：新建一个男生表malestudent (StudentCode, StudentName, Gender, Birthday) , 向malestudent表中插入两条记录。

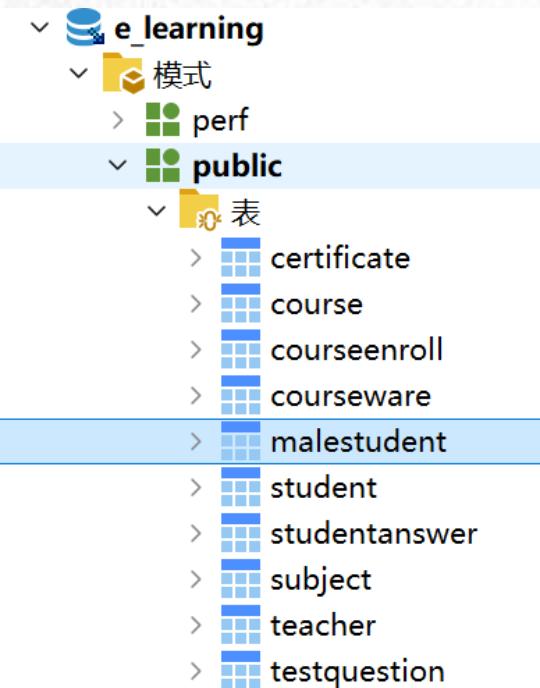
-- 创建新表

```
CREATE TABLE malestudent (
    StudentCode integer GENERATED BY DEFAULT AS IDENTITY,
    StudentName VARCHAR(6) NOT NULL,
    Gender CHAR(1) NOT NULL DEFAULT '男',
    Birthday DATE NOT NULL,
    PRIMARY KEY (StudentCode)
);
```

-- 插入多条数据

```
INSERT INTO malestudent(Gender, StudentName, Birthday)
VALUES ('男', '孟飞','2000-10-02'),
       ('男', '姜峰','2001-11-04');
```

表中StudentCode字段设置为“自增”，可以不给该字段赋值，系统会自动生成。



	StudentCode	StudentName	Gender	Birthday
1	1	孟飞	男	2000-10-02 00:00:00
2	2	姜峰	男	2001-11-04 00:00:00

# 自增列语法回顾

**GENERATED BY DEFAULT AS IDENTITY[(START WITH 起始值 INCREMENT BY 增量)]**

如，**GENERATED BY DEFAULT AS IDENTITY(START WITH 1001 INCREMENT BY 1)**

- GENERATED BY DEFAULT  
插入数据时，若没有为该列提供值，数据库会自动生成一个值。但用户可以手动插入值
- AS IDENTITY  
说明该列是一个自增列，数据库会自动管理其值的生成。
- START WITH 起始值  
设置自增序列的起始值，若省略则默认1。
- INCREMENT BY 增量  
定义自增的步长，若省略则默认1。

如果省略括号及其内容，则从1开始，每次递增1。

# 插入语句INSERT INTO与SELECT合并执行

从其他表提取一组记录插入到目标表中。

目标表须已存在，且结构与查询返回的字段值类型一致。

**INSERT INTO 目标表 [ (字段1, 字段2, ..... ) ]  
SELECT 源字段名表 FROM 源表名 [ WHERE 添加条件];**

例如：向malestudent表中插入记录，数据为student表中所有男生的信息。

**INSERT INTO malestudent**

**SELECT StudentCode, StudentName, Gender, Birthday  
FROM student WHERE Gender='男';**

注意：

1. INSERT INTO malestudent后不要加分号。
2. malestudent必须已经存在。

	StudentCode	StudentName	Gender	Birthday
1	1	孟飞	男	2000-10-02
2	2	姜峰	男	2001-11-04
3	1,002	汪洋	男	1999-11-09
4	1,004	张小贝	男	2005-01-19
5	1,005	马林	男	1985-11-09
6	1,006	张小宝	男	1992-07-01



## 修改语句UPDATE

对符合条件的记录的某个或某些字段值修改。

**UPDATE 表名**

**SET 字段名1 = 表达式1 [, 字段名2 = 表达式2..... ]**

**[WHERE 更新条件];**

- 用表达式值替换对应的字段值
- 无WHERE子句，更新所有记录

## 修改语句UPDATE

例如：修改student表中学号为1018的联系电话为18931000978。

```
UPDATE student
SET Phone='18931000978'
WHERE StudentCode=1018;
```

The screenshot shows a database interface with a script editor and a results panel.

**Script Editor:**

```
1 UPDATE student
2 SET Phone='18931000978'
3 WHERE StudentCode=1018
```

**Results Panel:**

Name	Value
Updated Rows	1
Query	UPDATE student SET Phone='18931000978' WHERE StudentCode=1018
Finish time	Sat Feb 08 11:31:16 CST 2025

At the bottom, it says: 1 row(s) updated - 7ms [11:31:16]

## 修改语句UPDATE

例如：修改course表，将学科名为“管理学”的课程的学分增加0.5，学时增加10%。

来自Subject表

```
UPDATE course  
SET Credits=Credits+0.5,  
Hours=Hours*(1+0.1)  
WHERE SubjectCode=(SELECT SubjectCode  
FROM subject  
WHERE SubjectName='管理学' ); -- 子查询
```

Name	Value
Updated Rows	4
Query	UPDATE course SET Credits=Credits+0.5, Hours=Hours*(1+0.1) WHERE SubjectCode=(SELECT SubjectCode FROM subject -- 子查询 WHERE SubjectName='管理学')
Finish time	Tue Mar 18 14:14:25 CST 2025



## 删除语句DELETE

删除符合条件的一条或多条记录。

**DELETE FROM 表名 [ WHERE 删除条件 ];**

无WHERE子句，删除所有记录！

例如：删除subject表中的“哲学”记录。

```
DELETE FROM subject  
WHERE SubjectName='哲学';
```

例如：从subject表中删除没有开过课的学科记录。

```
DELETE FROM subject  
WHERE SubjectCode NOT IN (SELECT SubjectCode FROM course); -- 子查询
```



## 小结：数据操纵语言DML

DML：数据更新语句

——INSERT

UPDATE

DELETE

SELECT语句**不**更新数据库中的任何数据！

# **THANK YOU!**