4.2 PL/SQL与可编程对象

4.2 PL/SQL与可编程对象

- PL/SQL块结构
- PL/SQL运算和常用函数
- 流程控制语句
- 视图
- 存储过程
- 自定义函数
- 触发器

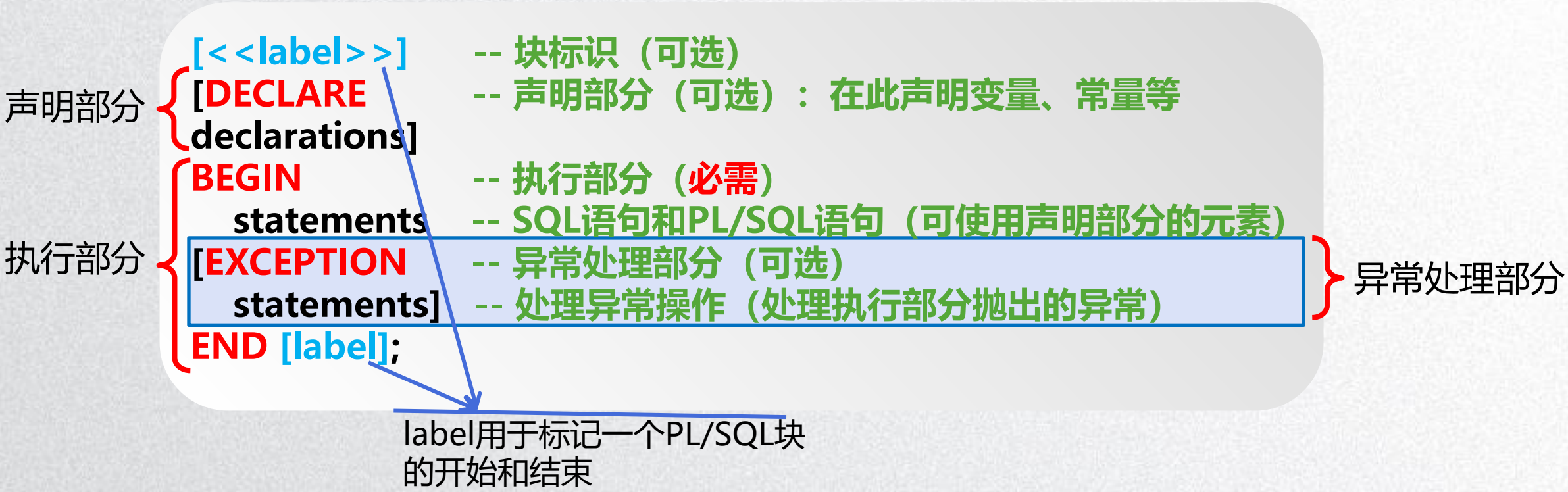
4.2.1 PL/SQL块结构

PL/SQL块结构

PL/SQL（Procedure Language & Structured Query Language）

- 是Kingbase ES对标准SQL的过程化扩展，
- 主要包含标准SQL和流程控制语句等内容。
- 是用于编写存储过程、函数、触发器等可编程对象的基本结构。

PL/SQL块结构定义



PL/SQL块分为**匿名块**和**命名块**。

- **匿名块**不在数据库中存储，通常用于执行一次性的任务。
- **命名块**可独立编译并存储在数据库中，可以被多次调用。

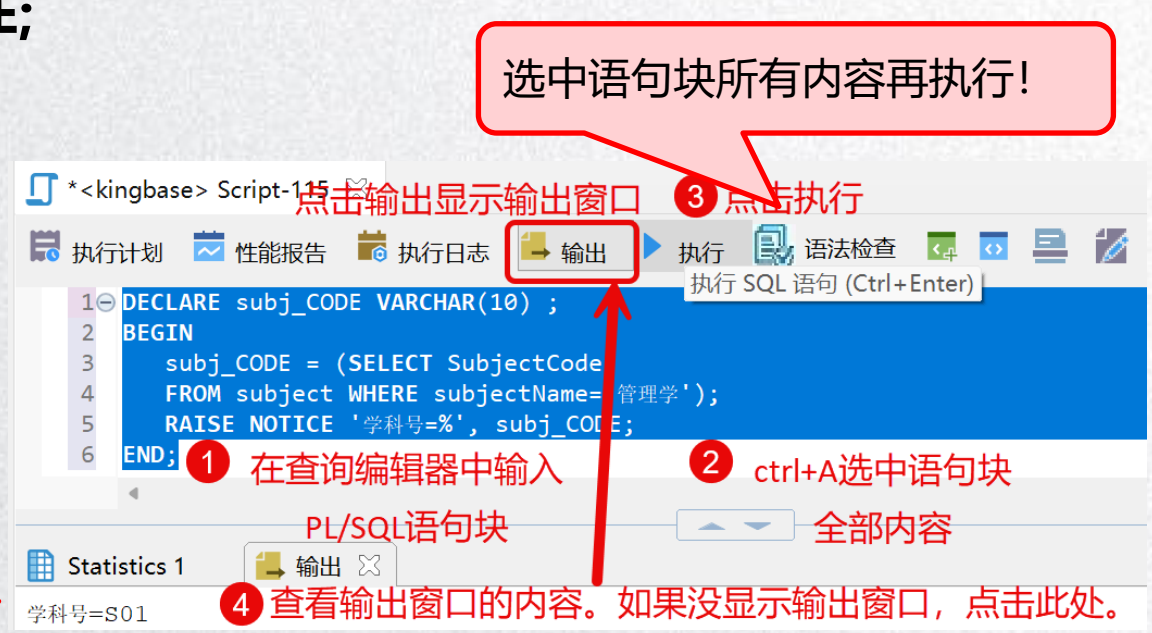
Kingbase ES 有4种命名块：存储过程、函数、触发器和包。

PL/SQL匿名块

【例】定义一个匿名块，从subject表中查找**管理学**的学科号。

```
DECLARE subj_CODE VARCHAR(10) ;
BEGIN
   subj_CODE = (SELECT SubjectCode
   FROM subject WHERE subjectName='管理学');
   RAISE NOTICE '学科号=%', subj_CODE;
END;
```

-- RAISE NOTICE将指定的消息输出到客户端



打开输出窗口查看!

4.2.2 PL/SQL运算和常用函数

PL/SQL运算和常用函数

PL/SQL增加了对运算、流程控制等的支持，用来实现复杂的数据检索和更新操作。

标识符：由用户定义的名称，用来标识各种对象, 如数据库、表、字段、变量、常量等。合法的标识符由字母、数字、下划线（_）和美元符号（$）组成，且必须以字母或下划线开头。避免使用SQL关键字等。

标识符还有一种所谓受限标识符，即被双引号括起的一个字符串序列。受限标识可以包含任何字符（若要包含一个双引号，则需写两个双引号）。

常量：在程序运行过程中值不变的量。

变量：指在程序运行过程中其值可以被改变的量。

使用前必须先在PL/SQL块的声明部分声明。

字符串型常量、日期时间型常量要用单引号扩起。

'上海'

'2017-10-23 10:40:30'

变量和常量的声明

声明变/常量

变量或常量名 [CONSTANT] 数据类型 [NOT NULL] [{DEFAULT | := | = } 表达式];

或

说明：

①变量或常量名必须符合PL/SQL标识符的命名规范。

②每次声明只能定义一个变量或常量。

③声明常量时加关键字CONSTANT，必须为常量赋初值，默认值为NULL。

④声明变量时若加关键字NOT NULL，则必须为变量赋初值。

⑤如果变量没有赋初值，则默认值为NULL。

⑥使用赋值运算符“=”或“:=”为变量或常量初始化。

举例：变量和常量的声明

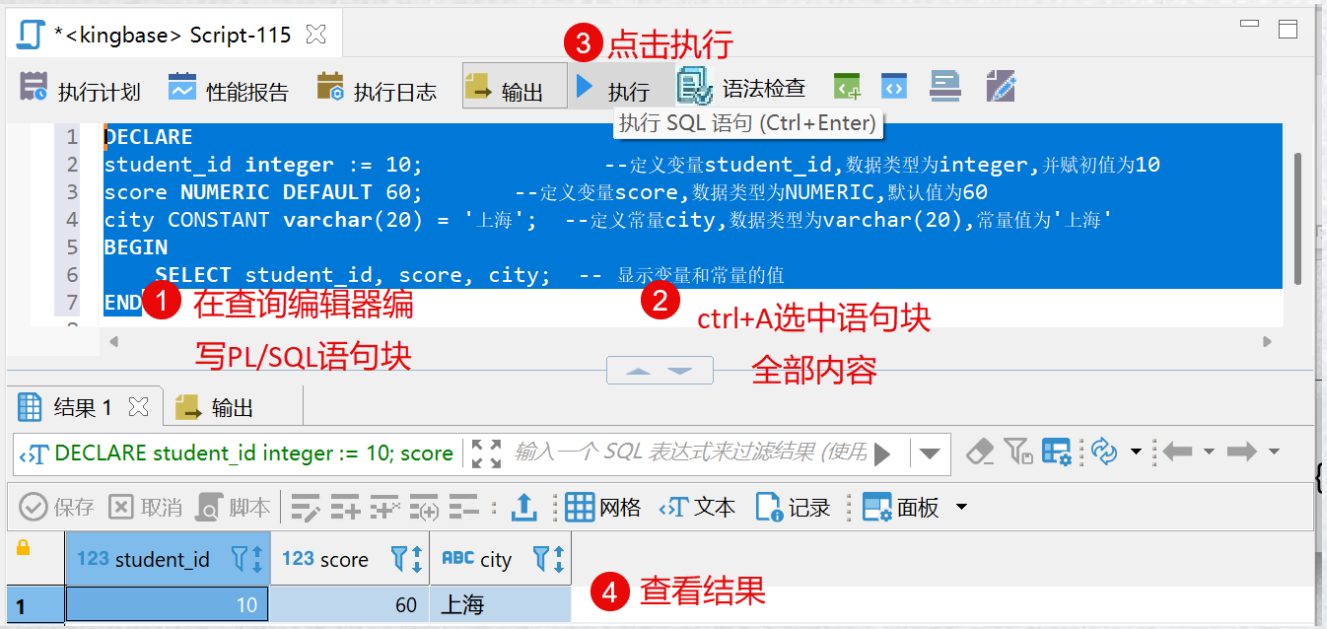
【例】PL/SQL块中声明变量和常量。

声明部分

```
DECLARE
student_id integer := 10;                --定义变量student_id,数据类型为integer,并赋初值为10
score NUMERIC DEFAULT 60;                --定义变量score,数据类型为NUMERIC,默认值为60
city CONSTANT varchar(20) = '上海';      --定义常量city,数据类型为varchar(20),常量值为'上海'
```

执行部分

```
BEGIN
        SELECT student_id, score, city;  -- 显示变量和常量的值
END
```



变量声明%TYPE

%TYPE：声明变量与已定义的变量类型相同，或与数据库表的某个字段的数据类型相同。

变量名 数据源%TYPE;　--数据源是已声明的某个变量或某表的某个字段

【例】使用%TYPE声明变量

声明部分

```
DECLARE
--定义变量v1,数据类型同student表中studentcode,初值为1
v1 student.studentcode%TYPE :=1;

--定义变量v2,类型同student表中studentName,初值为'李明'
v2 student.studentName%TYPE ='李明';

--定义变量v3,数据类型同v1
v3 v1%TYPE;
```

执行部分

```
BEGIN
        SELECT v1,v2,v3;
END
```

123 v1	ABC v2	123 v3
1	李明	[NULL]

变量的赋值

（1）使用赋值语句给变量赋值。赋值运算符为:= 或 =。

```
变量或常量名:=表达式;
变量或常量名=表达式;
```

（2）使用SELECT INTO语句给变量赋值。

```
SELECT 字段1 [, 字段2]... INTO 变量1 [,变量2]
FROM 表;
```

变量的赋值

【例】给变量赋值

声明部分

```
DECLARE
v1 student.studentcode%TYPE;      -- 定义变量v1, 数据类型同student表中studentcode
v2 student.studentName%TYPE;      -- 定义变量v2, 数据类型同student表中studentName
v3 v1%TYPE;                       -- 定义变量v3, 数据类型同v1
```

执行部分

```
BEGIN
    SELECT TOP 1 studentCode, studentName INTO v1,v2 FROM student ; --变量赋值
    v3 = 10;                      -- 变量赋值
    SELECT v1, v2, v3;            -- 显示各个变量的值
END
```

v1	v2	v3
1,001	杜斯	10

运算符和表达式

PL/SQL运算符分为算术运算符、比较运算符、逻辑运算符和位操作运算符等。

类　　别	所包含的运算符
算术运算符	+（加）、−（减）、*（乘）、/（除）、^（指数运算符）
比较运算符	>（大于）、< （小于）、=（等于）、<>（不等于）、!= （不等于）、^= （不等于）、>=（大于等于）、<= （小于等于）、[NOT] IN（属于某集合）、[NOT] BETWEEN AND（在两值之间）、IS [NOT] NULL（是空值）、[NOT] LIKE（字符串匹配）、[NOT] SIMILAR TO（字符串匹配）
逻辑运算符	NOT（逻辑非）、AND（逻辑与）、OR（逻辑或）
位操作运算符	&（位与）、|（位或）、～（位非）、#（位异或）、<<（左移）、>>（右移）
字符串连接运算符	||（将两个字符串连接成一个字符串）

表达式由运算对象、运算符及圆括号组成。

```
SELECT 12+45;    -- 结果为57，
--用SELECT查看表达式的值
```

常用函数

KingbaseES 数据库提供了很多内置函数，可在SQL语句中使用。

SELECT SQRT(4);

（1）数学函数

函 数 名	函 数 功 能	函 数 名	函 数 功 能
ABS(x)	计算x的绝对值	RANDOM ()	产生[0,1)之间的随机浮点数
MOD(x,y)	计算x除以y的余数	SIGN(x)	返回x的符号
DIV(x,y)	计算x除以y的余数	PI()	返回圆周率的值
SQRT(x)	计算x的平方根	SIN(x)	计算x（弧度）的正弦值
EXP(x)	计算e的x次方	SIND(X)	计算x（角度）的正弦值
POW(x,y)	计算x的y次方	COS(x)	计算x（弧度）的余弦值
LN(x)	计算x的自然对数	FLOOR(x)	返回小于或等于数字x的最大整数
LOG(x,y)	计算以x为底的y的对数	CEILING(x)	返回大于或等于数字x的最小整数
LEAST(x1,..,xn)	返回集合中最小的值	TRUNC(x,y)	返回数字x截断到y小数位
GREATEST(x1,..,xn)	返回集合中最大的值	ROUND(x[,n])	将x四舍五入到最接近的数值，n指定进行四舍五入的小数位数（默认值是0）

常用函数

(2) 日期时间函数

函 数 名	函数功能
NOW()	返回当前系统日期和时间，返回值类型为 timestamp
CURRENT_DATE()	返回当前系统日期
CURRENT_TIME(precision)	根据 precision 指定的精度参数，返回带有时区的时间
DATE_PART (text,{timestamp | interval})	从日期/时间值中抽取子域，text可以取day、month、year、hour 、 minute 、 second 、 quarter 等 ， 例 如 ，DATE_PART('day', timestamp'2025-2-16 20:08:40') 返回16 DATE_PART ('month', interval '2 years 3 months') 返回3
MONTH(d)	返回日期的月份值，例如，Month('2024-5-1')返回5
YEAR(d)	返回日期的年份值，例如，YEAR('2024-5-1')返回2024
EXTRACT(type FROM d)	从日期d中提取指定type的值。type可取 day、month、year、hour、 minute、 second、 quarter等，d是一个 timestamp、time 或 interval类型值的表达式， 例 如 ， EXTRACT(year FROM TIMESTAMP '2024-12-16 12:21:13') 返回2024
ADD_MONTHS (d,n)	返回起始日期d加上n（n>0）月的日期；当n<0时，返回起始日期d减去|n|月的日期 例 如 ， ADD_MONTHS(date'2025-4-20',-1) 返 回 2025-3-20 00:00:00

例如：查询2月出生的学生

```
SELECT * FROM student
WHERE Month(Birthday)=2;
```

常用函数

（3）字符串函数

例如：查询个人简介长度大于10的学生

```
SELECT * FROM student WHERE CHAR_LENGTH(Introduction)>=10;
```

函　数　名	函 数 功 能	函　数　名	函 数 功 能
CHAR_LENGTH(s)	返回字符串长度，即字符的数量	BIT_LENGTH(s)	返回字符串比特（位）长度
REVERSE(s)	返回颠倒的字符串	REPLACE(s1,s2,s3)	用s3替换s1中包含的s2
SUBSTRING(s,m,n)	返回字符串s的起始位置为m，长度为n的子串	CONCAT(s1,s2)	将两个参数连成一个完整的字符串
REPEAT(s,count)	返回将字符串s重复count次后得到的新字符串	RPAD(s,len,pad_char) LPAD(s,len,pad_char)	在字符串s的右侧或左侧填充指定的字符pad_char，直到达到指定的长度len
LEFT(s,n)	返回字符串s左边的n个字符	LOWER(s)	将字符串中的字母转换为小写字母
RIGHT(s,n)	返回字符串s右边的n个字符	UPPER(s)	将字符串中的字母转换为大写字母
LTRIM(s)	删除字符串s开始处的空格	ASCII(s)	返回字符串s的第一个字符的ASCII码
RTRIM(s)	删除字符串s结尾处的空格	CHR(n)	返回给定代码n的字符，n是整数

常用函数

(4) 聚合函数

聚合函数对一组值进行计算并返回一个结果，常用于对记录的分类汇总。
聚合函数经常与SELECT语句的GROUP BY子句一同使用。

函　数　名	函 数 功 能
AVG([ALL | DISTINCT] expression)	计算某个字段的平均值（此字段的值必须为数值型）
COUNT([ALL | DISTINCT] expression)	统计某个字段的值个数
COUNT(*)	统计记录个数
MAX([ALL | DISTINCT] expression)	查找某个字段的最大值
MIN([ALL | DISTINCT] expression)	查找某个字段的最小值
SUM([ALL | DISTINCT] expression)	计算某个字段的值总和（此字段的值必须为数值型）

常用函数

（5）数据类型转换函数

函　数　名	函 数 功 能
CAST(ex AS 目标类型)	将ex转换为目标数据类型。ex可以是常量、列、表达式。 例如，CAST ('2023-02-21' AS DATE)返回'2023-02-21 00:00:00'
TO_CHAR (timestamp或date 数据, fmt)	将日期及时间戳数据转成字符串。 fmt参数指定转换格式： 'MONTH'返回月份名称；'DAY'返回是星期几；'IW' 返回本年的第几个星期； 'DDD'返回本年的第几天；'Q'返回是第几季度 例如，TO_CHAR ('2024-08-30', 'DAY')返回'FRIDAY' TO_CHAR ('2024-08-30', 'YY/MM/DD') 返回'24/08/30' TO_CHAR ('2024-08-30 00:00:11', 'YYYY-MM-DD') 返回'2024-08-30'
TO_CHAR (int或numeric数据, fmt)	将整数或数字转换为字符串，fmt用于指定数字的显示格式。 例如，TO_CHAR(-125.8, '9999D99S') 返回 ' 125.80-' 格式模型 '9999D99S'说明： 9：表示数字位置。若整数部分的数字位数少于9的数量，最左边补空格；若小数部分的数字位数少于9的数量，最右边补0 D：表示小数点 S：表示符号位置（正号或负号）
TO_NUMBER(text,fmt)	用fmt指定的格式，将字符串text转换为数字 例如，to_number('-12,454.8') 返回-12,454.8

常用函数

（6）系统信息函数

函　数　名	函 数 功 能
VERSION()	返回KingbaseES数据库的版本号
CURRENT_DATABASE()	返回当前数据库名
GETUSERNAME()	返回当前用户名。
MD5(s)	计算字符串s的 MD5 哈希值, 用来对用户的数据进行加密

4.2.3 流程控制语句

PL/SQL 流程控制语句

类似于其他编程语言中的大括号{}

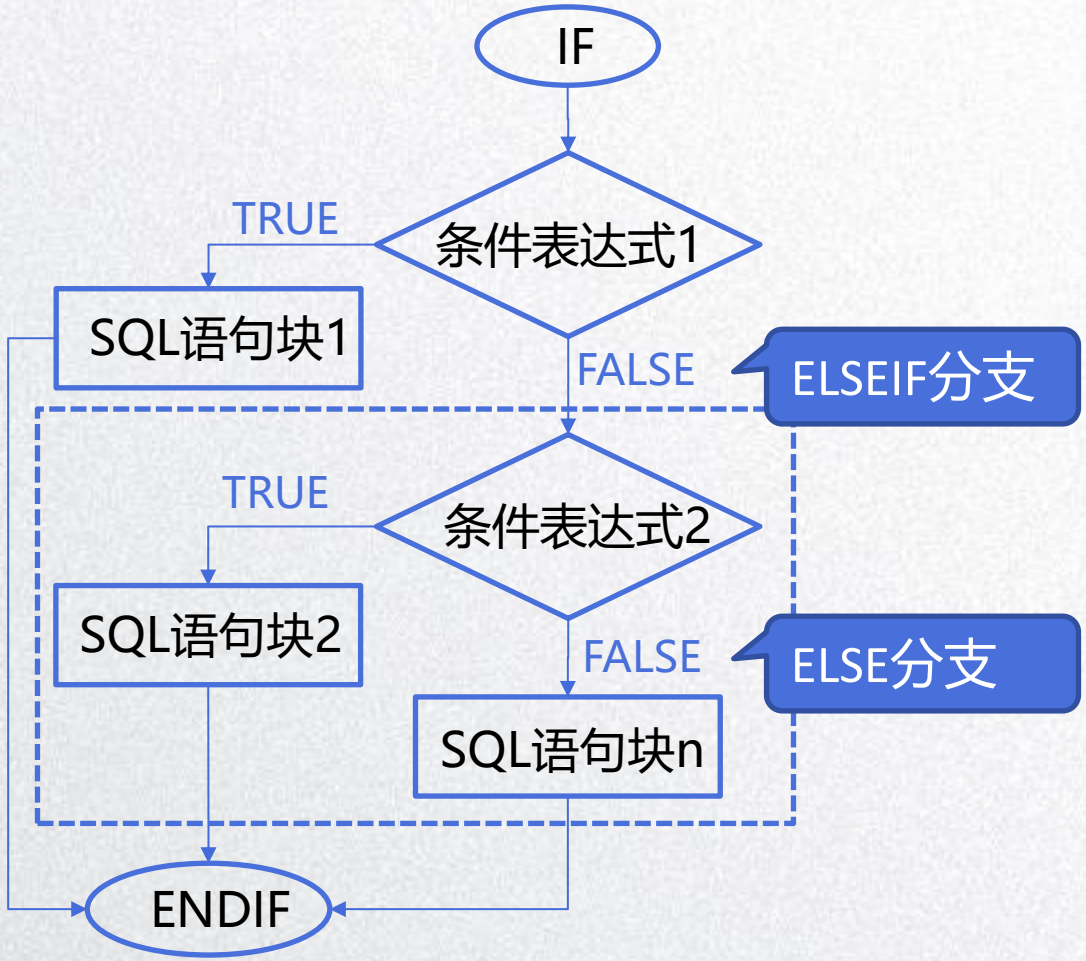
（1）BEGIN...END语句

多条SQL语句用BEGIN...END组合起来
形成一个SQL语句块

（2）IF语句

IF条件表达式1 THEN SQL语句块1
[ELSEIF 条件表达式2 THEN SQL语句块2]...
[ELSE SQL语句块n]
END IF

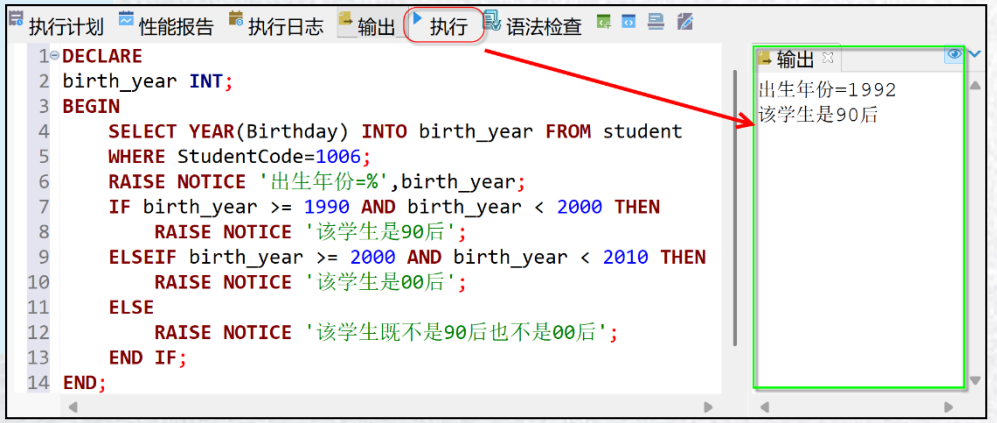
如果表达式的值为true，则执行IF后面的语句块；否则，如果有ELSEIF，则计算ELSEIF后面表达式，依此类推；最后如果有ELSE语句，执行ELSE后面的语句



举例：IF-ELSEIF-ELSE语句

【例】编写PL/SQL块，查询学号为1006的学生出生年份，判断该生是90后、00后，或都不是。

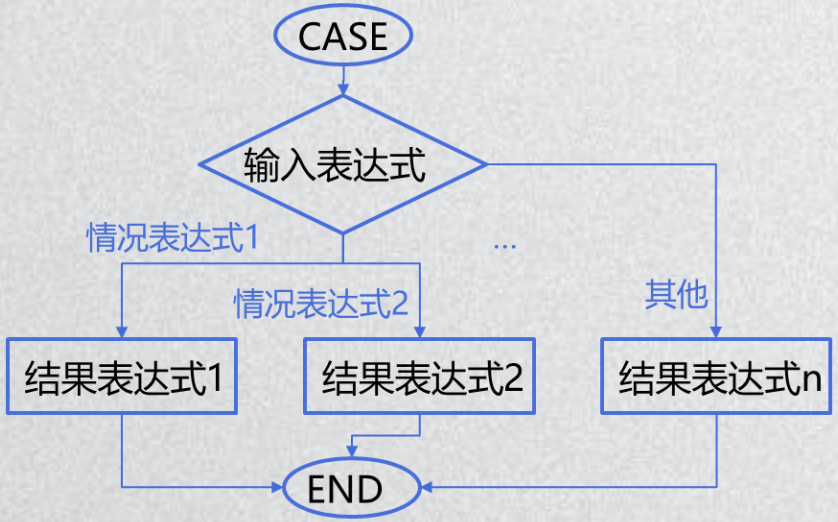
```
DECLARE
birth_year INT;
BEGIN
    SELECT YEAR(Birthday) INTO birth_year FROM student
    WHERE StudentCode=1006;
    RAISE NOTICE '出生年份=%',birth_year;
    IF birth_year >= 1990 AND birth_year < 2000 THEN
        RAISE NOTICE '该学生是90后';
    ELSEIF birth_year >= 2000 AND birth_year < 2010 THEN
        RAISE NOTICE '该学生是00后';
    ELSE
        RAISE NOTICE '该学生既不是90后也不是00后';
    END IF;
END;
```



(3)标准SQL的CASE表达式

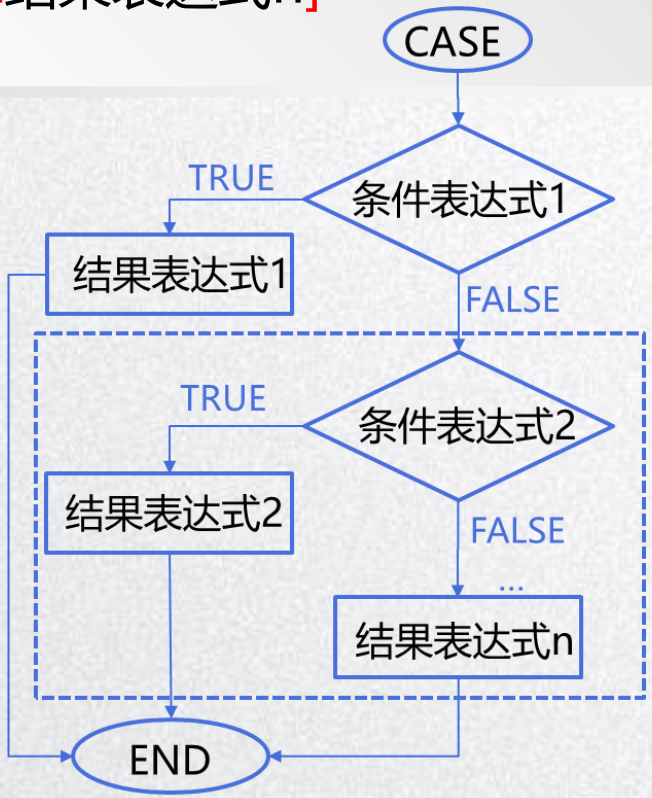
① **简单CASE表达式**：将某个表达式与一组情况表达式进行比较以确定结果。

```
CASE 输入表达式
WHEN 情况表达式1 THEN 结果表达式1
  …
  [ELSE 结果表达式n]
END
```



② **搜索CASE表达式**：计算一组条件表达式以确定结果。

```
CASE
  WHEN 条件表达式1 THEN 结果表达式1
  …
  [ELSE结果表达式n]
END
```



简单CASE表达式举例

【例】查询教师的信息，若AdminYN字段取值true,输出“管理员”，否则“教师”。

```
SELECT TeacherName,
    CASE AdminYN
        WHEN True THEN '管理员'
        WHEN False THEN '教师'
    END AS "岗位"
FROM teacher;
```

字符串值用单引号，字段名用双引号

整个case语句的结果作为岗位字段的值

	TeacherName	岗位
1	许虎	教师
2	刘蔷	教师
3	高华	教师
4	林致文	教师
5	王名勋	教师
6	李菱智	教师
7	戴榕	教师
8	许威华	教师

搜索CASE表达式举例

【例】统计每个学生平均成绩并划分等级。

```
SELECT StudentCode AS "学号", ROUND(AVG(Score),2) AS "平均成绩",
  CASE
    WHEN AVG(Score)>=90 THEN 'A'
    WHEN AVG(Score)>=80 THEN 'B'
    WHEN AVG(Score)>=70 THEN 'C'
    WHEN AVG(Score)>=60 THEN 'D'
    WHEN AVG(Score)<60 THEN 'E'
  END AS "等级"
FROM courseenroll
GROUP BY StudentCode;
```

学号	平均成绩	等级
1,002	71	C
1,003	80.33	B
1,004	89.67	B
1,001	76.67	C
1,005	63.67	D
1,010	78.67	C

(3)PL/SQL语句块中的**CASE语句**

PL/SQL的CASE语句主要用于PL/SQL块中，与标准SQL的CASE表达式用法相同，语法略有不同。PL/SQL中的CASE语句以“END CASE;”作为语句结束标志。

① 简单CASE语句

```
CASE 输入表达式
WHEN 情况表达式1 THEN 结果表达式1;
  …
  [ELSE 结果表达式n;]
END CASE;
```

末尾有分号

末尾有分号

② 搜索CASE语句

```
CASE
  WHEN 条件表达式1 THEN 结果表达式1;
  …
  [ELSE结果表达式n;]
END CASE;
```

独立的CASE语句，与上面的SELECT语句是并列关系

【例】编写PL/SQL块，查询学号为1004的学生平均成绩及等级。

输出

```
学号为1004的学生平均成绩： 89.7, B等
```

```
DECLARE
   v_avg courseenroll.score%TYPE ;
   v_grade char(1);
BEGIN
   SELECT AVG(score) INTO v_avg  FROM
courseenroll WHERE studentCode=1004; --平均成绩
  CASE
     WHEN v_avg>=90 THEN v_grade='A';
     WHEN v_avg>=80 THEN v_grade='B';
     WHEN v_avg>=70 THEN v_grade='C';
     WHEN v_avg>=60 THEN v_grade='D';
     WHEN v_avg<60 THEN v_grade='E';
  END CASE;      --等级
   RAISE NOTICE '学号为1004的学生平均成绩: %, %等
', v_avg,v_grade;
END
```

（4）循环语句

实现一条SQL语句或SQL语句块重复执行。

循环类型	语法格式	适用场景
基本LOOP	LOOP -- *循环体中的语句* EXIT WHEN 条件; END LOOP;	最基本的循环结构。需要使用 EXIT WHEN或EXIT语句退出循环，否则会无限循环。 适用于不确定循环次数的情况。
WHILE LOOP	WHILE 条件 LOOP -- *循环体中的语句* END LOOP;	在每次循环开始前检查条件，条件为真时执行循环体。条件为假时退出。适用于循环次数不确定，但需要在循环开始前检查条件的情况。
FOR LOOP	FOR 循环变量 IN [REVERSE] 初始值..终止值 LOOP -- *循环体中的语句* END LOOP;	**数值范围 FOR 循环** 循环次数是确定的，循环变量会自动递增或递减。可以使用 REVERSE 关键字使循环变量递减。循环变量超出范围时退出。适用于已知循环次数的情况。
	FOR 循环变量 IN (查询语句) LOOP -- *循环体中的语句* END LOOP;	**隐式游标 FOR 循环** 循环变量依次遍历查询结果集，是 PL/SQL 中 遍历查询结果集的最优方式。
FOREACH	FOREACH 元素变量 IN 集合 LOOP -- *循环体中的语句* END LOOP;	自动遍历集合中的每个元素，并将每个元素赋值给循环变量。用于处理复杂数据结构，如数组、表或其他集合类型。

循环语句支持在循环未结束时提前退出循环或者退出本次循环的功能。

- 退出本次循环，开始下一次循环：CONTINUE、CONTINUE WHEN语句；
- 退出循环：EXIT、EXIT WHEN语句。

举例：FOR 元素变量 IN **(查询语句)** LOOP

【例】编写PL/SQL块，查询学分大于3的课程及其名称。

FOR c IN (查询语句) LOOP
FOR 循环自动绑定查询，row变量（这里用c）自动关联查询结果集

```
BEGIN
    FOR c IN (SELECT * FROM "course" ) LOOP
        IF c.credits>3 THEN
            RAISE NOTICE '课程名：%', c.CourseName;
        ELSE
            NULL;
        END IF;
    END LOOP;
END;
```

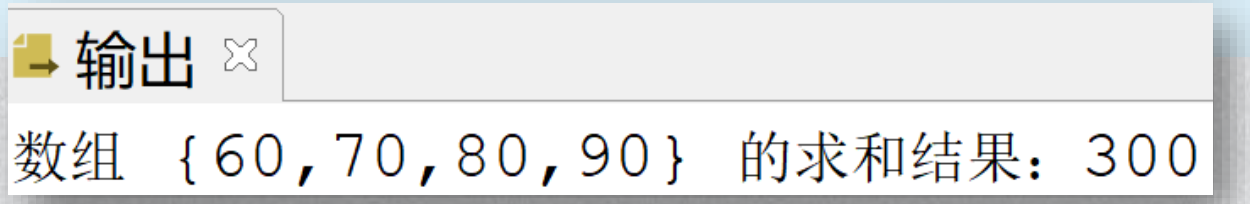
输出

```
课程名：信息系统与数据库技术
课程名：操作系统
课程名：高等数学微积分
```

举例：FOREACH 元素变量 IN **数组** LOOP

【例】编写PL/SQL块，遍历一个数组并求和输出。

```
DECLARE
   arr int[] := ARRAY[60,70,80, 90]; -- 定义数组
   s int8 := 0;  -- 求和变量
   x int;       -- 循环变量（存储数组中的每个元素）
BEGIN
   FOREACH x IN ARRAY arr LOOP
      s := s + x;  -- 累加求和
   END LOOP;
   RAISE NOTICE '数组 % 的求和结果： %', arr, s;
END;
```



4.2.4 视图

引入视图的原因

- 安全因素
- 构建更符合特定用户直觉的个性化的关系集合

Stu	S		Birthday	Photo	Email	Phone	Lo	Introduction	RegisterDate	Pa
1,001	杜斯	女	1998-11-05 00:00:00	1000.jpg	1001@elearning.com	13358301911	上海	数学专业毕业在即	2017-07-21 15:40:44	666666
1,002	汪洋	男	1999-11-09 00:00:00	1001.jpg	1002@elearning.com	13834992564	北京	对文学感兴趣	2017-07-25 22:30:48	666666
1,003	林豆豆	女	1997-12-15 00:00:00	1002.jpg	1003@elearning.com	13243964904	江苏	网络选课拓展知识面	2018-06-04 14:36:36	123456
1,004	张小贝	男	2005-01-19 00:00:00	1003.jpg	1004@elearning.com	13142540852	山东	到这里提前上大学	2017-08-28 09:20:58	666666
1,005	马林	男	1985-11-09 00:00:00	1004.jpg	1005@elearning.com	13971113836	浙江	已经工作，充电	2017-09-01 20:11:04	666666
1,006	张小宝	男	1992-07-01 00:00:00	1005.jpg	1006@sina.cn	13713143513	黑龙江	回来恶补一些专业知识	2017-09-17 18:15:07	666666
1,007	宋思明	男	1995-10-02 00:00:00	1006.jpg	1007@sina.cn	13916801546	江西	无话可说	2017-09-23 15:55:11	666666
1,008	徐米拉	女	2000-03-30 00:00:00	1007.jpg	1008@sina.cn	13225414354	湖北	到这里才发现学海无涯	2017-09-30 19:51:14	666666
1,009	杨康	男	2001-11-09 00:00:00	1008.jpg	1009@sina.cn	13345482022	上海	发现e课堂真好!	2017-10-03 13:41:19	666666
1,010	郭靖	男	1993-04-05 00:00:00	1009.jpg	1010@sina.cn	13974339695	北京	对心理学感兴趣	2017-11-05 14:45:33	666666

有些字段不应对所有用户可见，比如密码。

辅导员希望了解的信息 →

学生名单				
学号	**姓名**	**班级**	**GPA**	**排名**
1001	杜斯	CS1801	85	2
1002	汪洋	CS1801	86	1
1003	林豆豆	CS1802	74	3

视图

视图是在数据表基础上定义的一个**虚拟表**，在使用视图时从数据表提取查询结果。

视图只有结构而不实际存储数据。它的数据来自定义视图的SELECT语句所引用的基本表，并在使用视图时动态地从基本表中提取。

班主任

学生名单	
学号	姓名
1001	杜斯
1002	汪洋
1003	林豆豆

选课人数	
课号	人数
C001	5
C002	5
C003	4

教务长

数据库设计者

student表

StudentCode	StudentName	Gender	Birthday	Photo	Email	Phone	Location	Introduction	RegisterDate	Password
1001	杜斯	女	1998-11-05	1000.jpg	1001@elearning.com	13358301911	上海	数学专业毕业在即	2018-03-12 21:00:01	666666
1002	汪洋	男	1999-11-09	1001.jpg	1002@elearning.com	13834992564	北京	对文学感兴趣	2017-08-25 10:26:44	wy6666
1003	林豆豆	女	1997-12-15	1002.jpg	1003@elearning.com	13243964904	江苏	网络选课和基础知识	2017-08-05 11:42:52	666666
1004	张小贝	男	2005-01-19	1003.jpg	1004@elearning.com	13142540852	山东	到这里提前上大学	2017-08-28 09:20:58	666666
1005	马林	男	1985-11-09	1004.jpg	1005@elearning.com	13971113836	浙江	已经工作，为电	2017-09-01 20:11:04	666666
1006	张小宝	男	1992-07-01	1005.jpg	1006@sina.cn	13713143513	黑龙江	回来再补一些专业	2017-09-17 18:15:07	666666
1007	宋思明	男	1995-10-02	1006.jpg	1007@sina.cn	13916801546	江西	无话可说	2017-09-23 15:55:11	666666
1008	徐米拉	女	2000-03-30	1007.jpg	1008@sina.cn	13225414354	湖北	到这里才发现学海	2017-09-30 19:51:14	666666

coursenenroll表

StudentCode	CourseCode	Score	TestTime
1001	C001	70	2018-02-20 14:15:08
1001	C002	80	2017-12-15 09:20:11
1001	C003	90	2018-02-20 14:15:08
1002	C001	80	2017-07-21 16:21:40
1002	C002	60	2017-07-12 16:20:00
1002	C003	40	2017-08-15 09:20:33

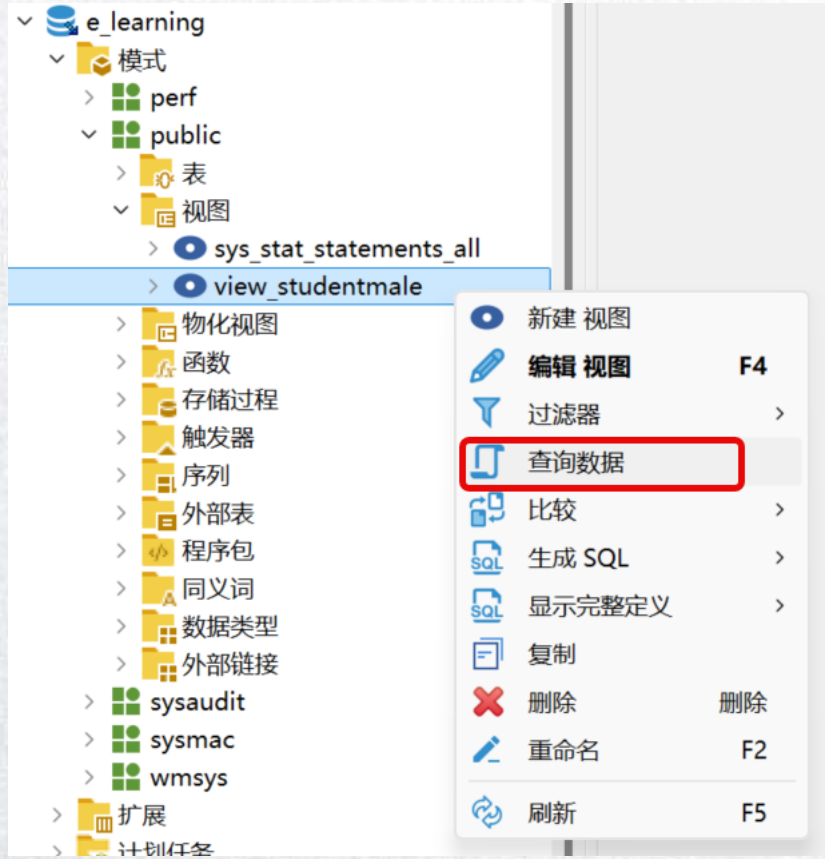
使用SQL语句创建视图

```
CREATE VIEW 视图名称[(列名称[, ...])]
AS
SELECT查询语句;
```

视图不是PL/SQL命名块!

【例】创建男生信息视图view_studentmale，支持查询所有男生的学生**学号**、**姓名**、**性别**和**出生日期**。

```
CREATE VIEW view_studentmale
AS
SELECT StudentCode 学号, StudentName 姓名,Gender 性别,
Birthday 生日
FROM student WHERE Gender='男';
```



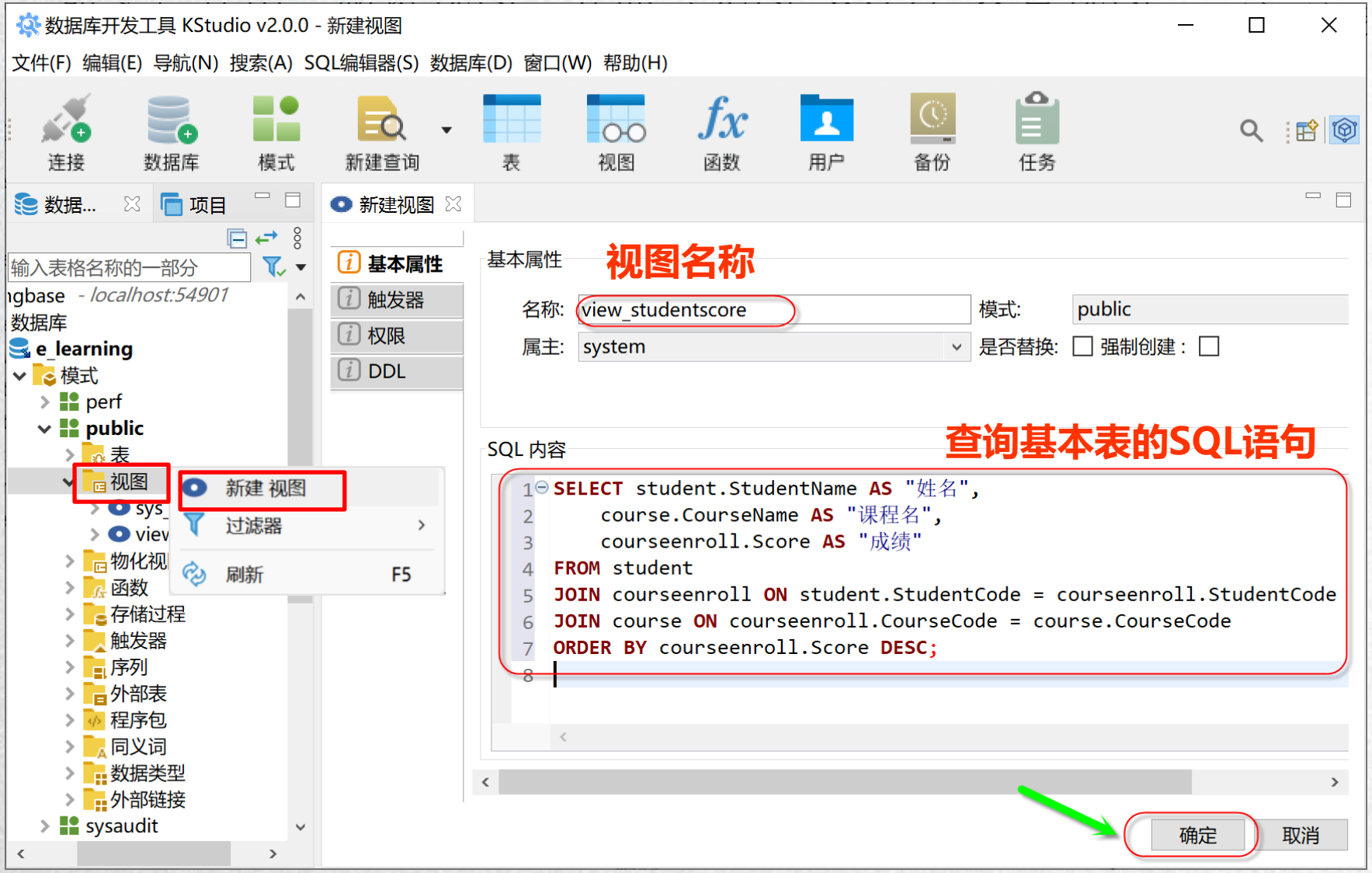
查看视图数据：展开“e_learning→模式→public→视图”，右击view_studentmale项，从快捷菜单中选择“查询数据”命令。

使用KStudio数据库对象管理工具创建视图

【例】利用KStudio视图创建向导，创建视图view_studentscore，显示姓名、课程名和成绩，按成绩降序排序。

步骤：右击视图，从快捷菜单中选择“新建 视图”命令。

填写视图名称；
填写SQL内容。



举例：基于视图创建新视图

【例】创建视图view_studentexcellent，选拔优培生（平均成绩大于等于85的学生），并按平均成绩从高到低排列。

```
CREATE VIEW view_studentexcellent
AS
SELECT "姓名" ,ROUND(AVG("成绩"),1) AS "平均成绩"
FROM view_studentscore
GROUP BY "姓名"
HAVING "平均成绩" >=85
ORDER BY "平均成绩" DESC;
```

已有的视图名

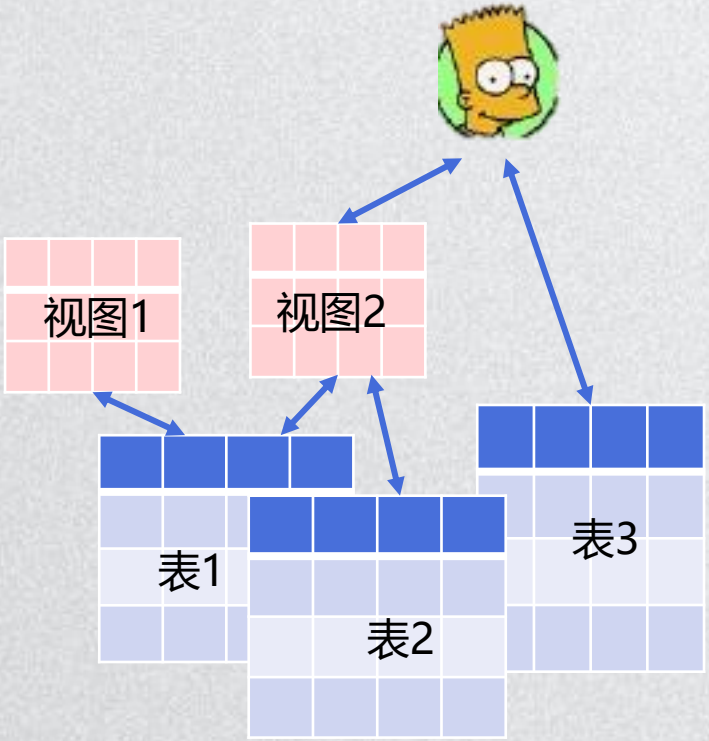
view_studentscore

	姓名	课程名	成绩
1	林豆豆	心理学与生活	[NULL]
2	马林	心理学与生活	[NULL]
3	杜斯	心理学与生活	[NULL]
4	马林	管理心理学	[NULL]
5	汪洋	管理心理学	[NULL]
6	汪洋	心理学与生活	[NULL]
7	林豆豆	唐诗经典	[NULL]
8	张小贝	多媒体技术及应用	99
9	杜斯	计算机网络技术	93
10	林豆豆	信息系统与数据库技术	90
11	汪洋	信息系统与数据库技术	88
12	张小贝	信息系统与数据库技术	88
13	郭靖	高等数学微积分	86
14	张小贝	计算机网络技术	82
15	郭靖	沟通心理学	80
16	马林	计算机网络技术	77
17	林豆豆	计算机网络技术	76
18	林豆豆	多媒体技术及应用	75
19	杜斯	多媒体技术及应用	72

200 25

视图的使用

视图与表的用法相同，可以通过视图查询和更新数据库。



举例1：使用视图查询数据

【例】使用视图view_studentscore 统计“多媒体技术及应用”课程的平均分。

```
SELECT 课程名, AVG(成绩) AS 平均分
FROM view_studentscore
GROUP BY 课程名
HAVING 课程名='多媒体技术及应用';
```

	课程名	平均分
1	多媒体技术及应用	74

```
SELECT "课程名", AVG(成绩) AS "平均分"
FROM view_studentscore
GROUP BY "课程名"
HAVING "课程名"='多媒体技术及应用';
```

受限标识符

举例2：使用视图更新数据

【例】向视图view_studentmale中插入一条记录（1104, "赵谦", "男", "2000-12-12"）。

```
INSERT INTO view_studentmale
VALUES('1104', '赵谦', '男', '2002-12-12' );
```

实际上是添加到Student表

慎重使用视图进行更新操作：

① 定义时，使用了运算或聚合函数，以及DISTINCT、GROUP BY等语句的视图，不可更新（增、删、改）；

② 无论插入、修改和删除操作，一次只能操作一个基本表中的数据；

③ 插入操作，必须包含视图引用的基本表的所有不能为空的列；

删除操作，只能在基于单表定义的视图上进行。

维护视图

- 修改视图
  - 在视图设计窗口进行修改
  - 直接修改视图定义语句
- 删除视图
  - 在KStudio左栏选中删除
  - Drop View语句删除
- 重命名视图
  - 在KStudio左栏选中，选择“重命名”可以修改视图名

删除视图对基本表没有任何影响，因为视图只是个虚拟表。

小结：视图

在数据表基础上定义的一个虚拟表，
在使用视图时从数据表提取查询结果。

班主任

学生名单	
学号	姓名
1001	杜斯
1002	汪洋
1003	林豆豆

选课人数	
课号	人数
C001	5
C002	5
C003	4

教务长

数据库
设计者

StudentCode	StudentName	Gender	Birthday	Photo	Email	Phone	Location	Introduction	RegisterDate	Password
1001	杜斯	女	1998-11-05	1000.jpg	1001@elearning.com	13358301911	上海	数学专业毕业在即	2018-03-12 21:00:01	666666
1002	汪洋	男	1999-11-09	1001.jpg	1002@elearning.com	13834992564	北京	对文学感兴趣	2017-08-25 10:26:44	wy6666
1003	林豆豆	女	1997-12-15	1002.jpg	1003@elearning.com	13243964904	江苏	网络选课和基础知识	2017-08-05 11:42:52	666666
1004	张小贝	男	2005-01-19	1003.jpg	1004@elearning.com	13142540852	山东	到处里得前上大学	2017-08-28 09:20:58	666666
1005	马林	男	1985-11-09	1004.jpg	1005@elearning.com	13971113836	浙江	已经工作，为电	2017-09-01 20:11:04	666666
1006	张小宝	男	1992-07-01	1005.jpg	1006@sina.cn	13713143513	黑龙江	回来再补一些专业	2017-09-17 18:15:07	666666
1007	宋思明	男	1995-10-02	1006.jpg	1007@sina.cn	13916801546	江西	无话可说	2017-09-23 15:55:11	666666
1008	徐米拉	女	2000-03-30	1007.jpg	1008@sina.cn	13225414354	湖北	到处里才反观学海	2017-09-30 19:51:14	666666

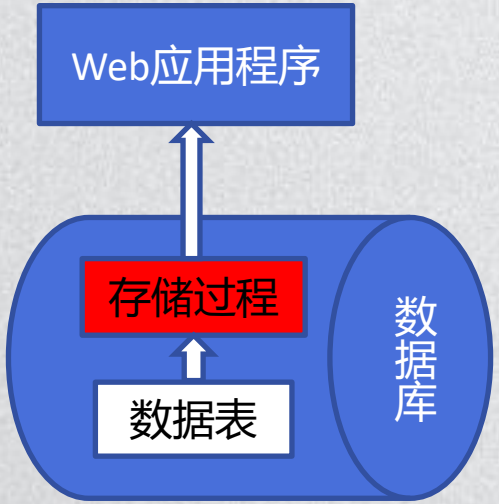
StudentCode	CourseCode	Score	TestTime
1001	C001	70	2018-02-20 14:15:08
1001	C002	80	2017-12-15 09:20:11
1001	C003	90	2018-02-20 14:15:08
1002	C001	80	2017-07-21 16:21:40
1002	C002	60	2017-07-12 16:20:00
1002	C003	40	2017-08-15 09:20:33

4.2.5 存储过程

存储过程

存储过程是完成一定数据访问和处理功能的PL/SQL命名语句块，是一种数据库对象。

存储过程类似于高级编程语言程序中的过程（方法、函数），它由应用程序调用执行，将结果返回给调用它的程序。



- **快速（在服务器端预编译）**
- **灵活（参数化更灵活）**
- **安全（为特定用户开放）**

使用SQL语句创建存储过程

创建:

可选的，如果存储过程已存在，替换原有存储过程

多个参数之间用逗号分隔。()不可省
[IN | OUT | INOUT] 参数名 参数类型
IN：输入；OUT：输出；INOUT：输入/输出

```
CREATE [OR REPLACE] PROCDURE 存储过程名( [形式参数列表])
AS
[DECLARE 变量声明]          -- 可选
BEGIN
    <存储过程体>            -- PL/SQL语句序列
    [EXCEPTION 异常处理]    -- 可选
END ;
```

调用:

```
CALL 存储过程名([参数列表]);
```

参数列表：传递给存储过程的零个或多个参数，用逗号分隔。

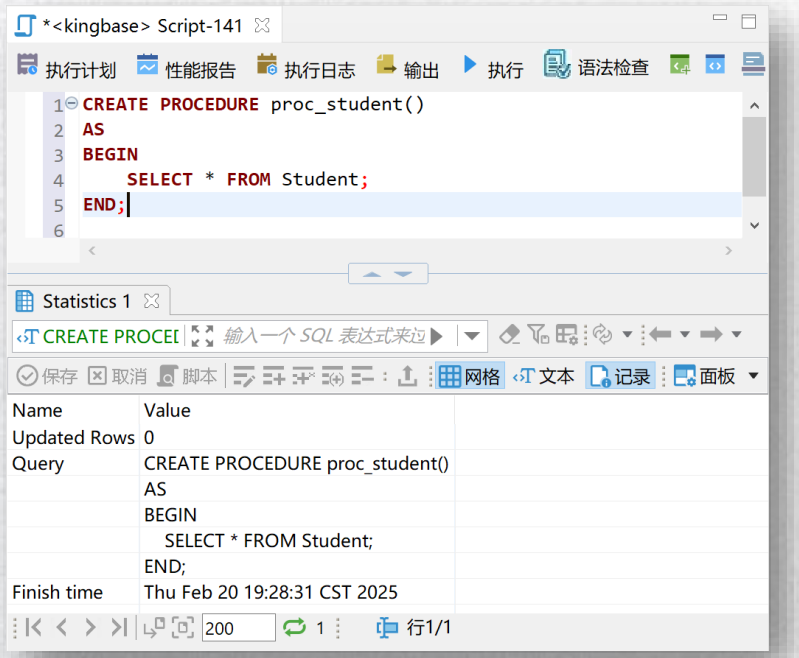
举例1：存储过程（无参数）

【例】创建无参数存储过程proc_student，查询所有学生信息。

创建存储过程：

```
CREATE PROCEDURE proc_student()
AS
BEGIN
    SELECT * FROM student;
END;
```

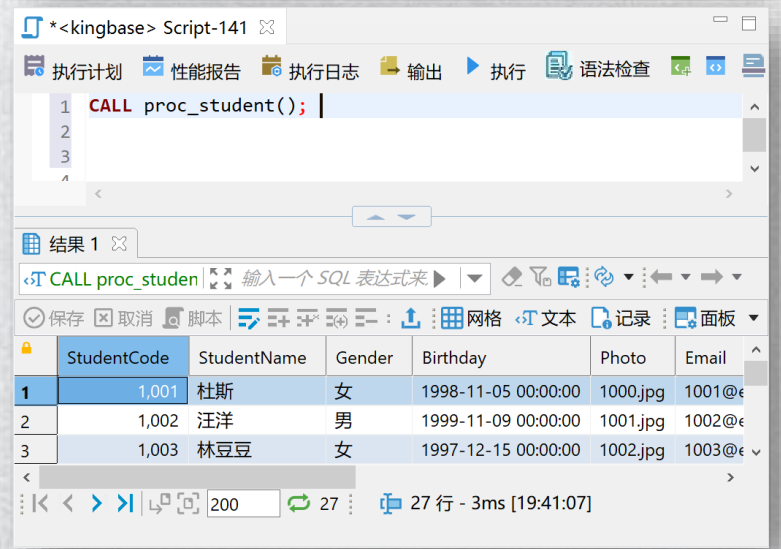
括号不能省略



调用存储过程：

```
CALL proc_student();
```

括号不能省略



举例2：存储过程（有输入参数）

【例】创建带有输入参数的存储过程proc_searchstudent，按姓名查询特定学生信息。

创建存储过程：

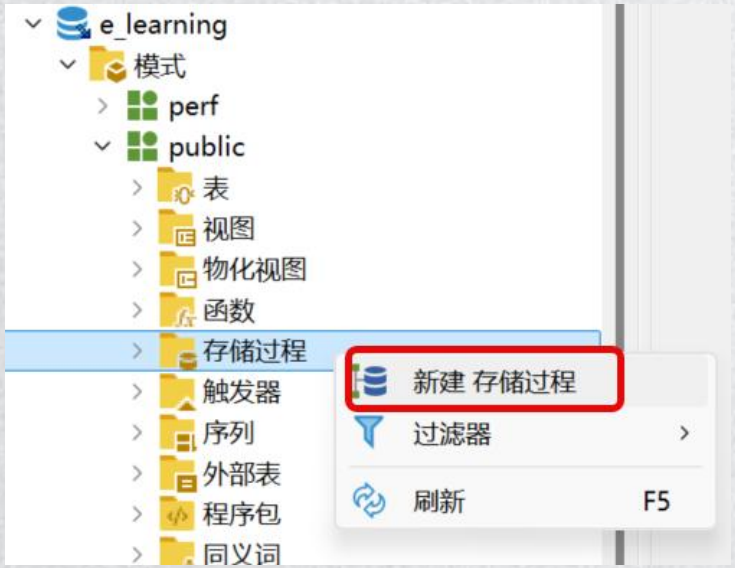
```
CREATE PROCEDURE proc_searchstudent(IN stname varchar(16))
                                        --stname是IN类型参数
AS
BEGIN
    SELECT * FROM STUDENT WHERE StudentName=stname;
                                        -- 按参数stname查询
END;
```

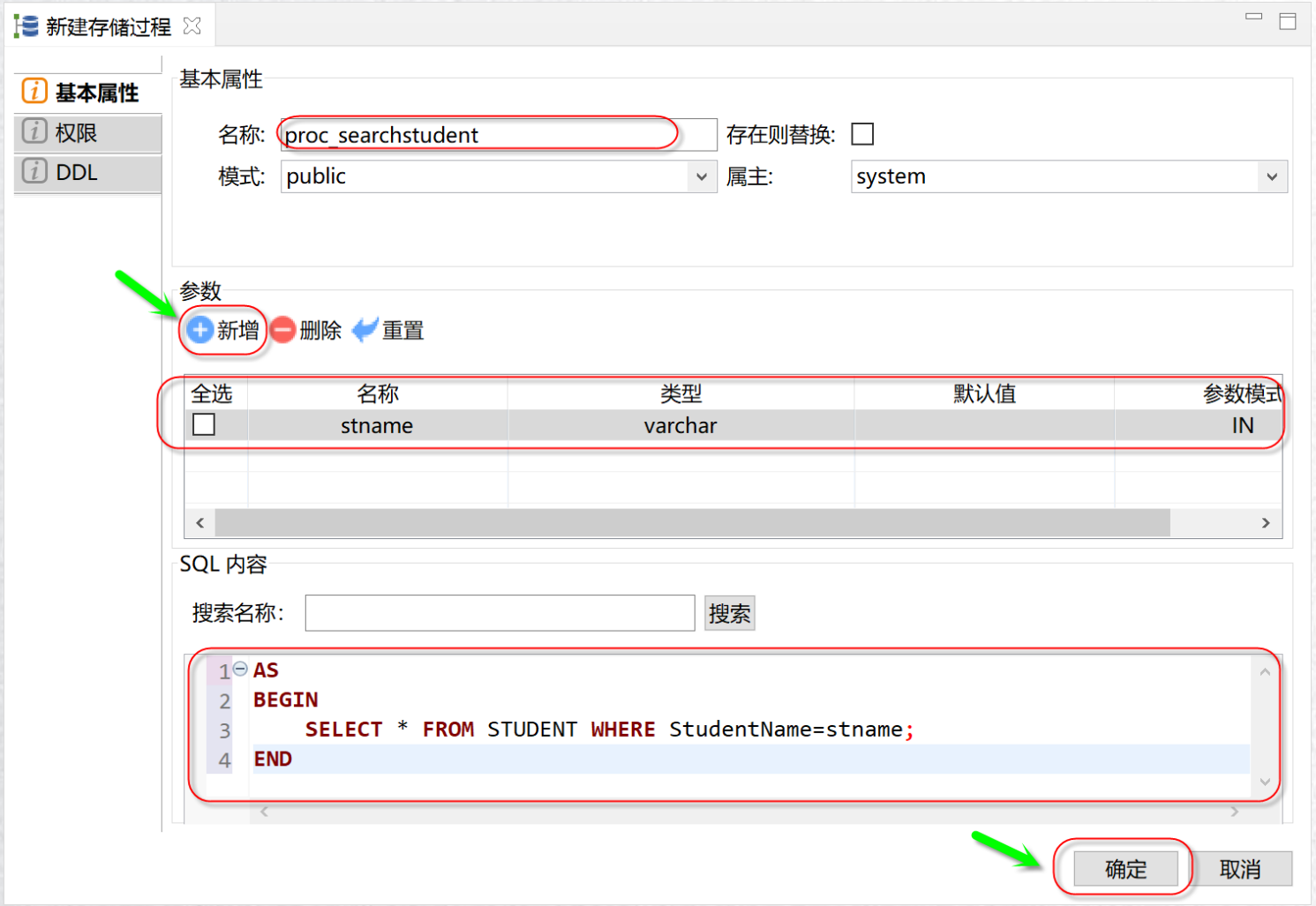
调用存储过程：

```
CALL proc_searchstudent('林豆豆');
```

使用KStudio数据库对象管理工具创建存储过程

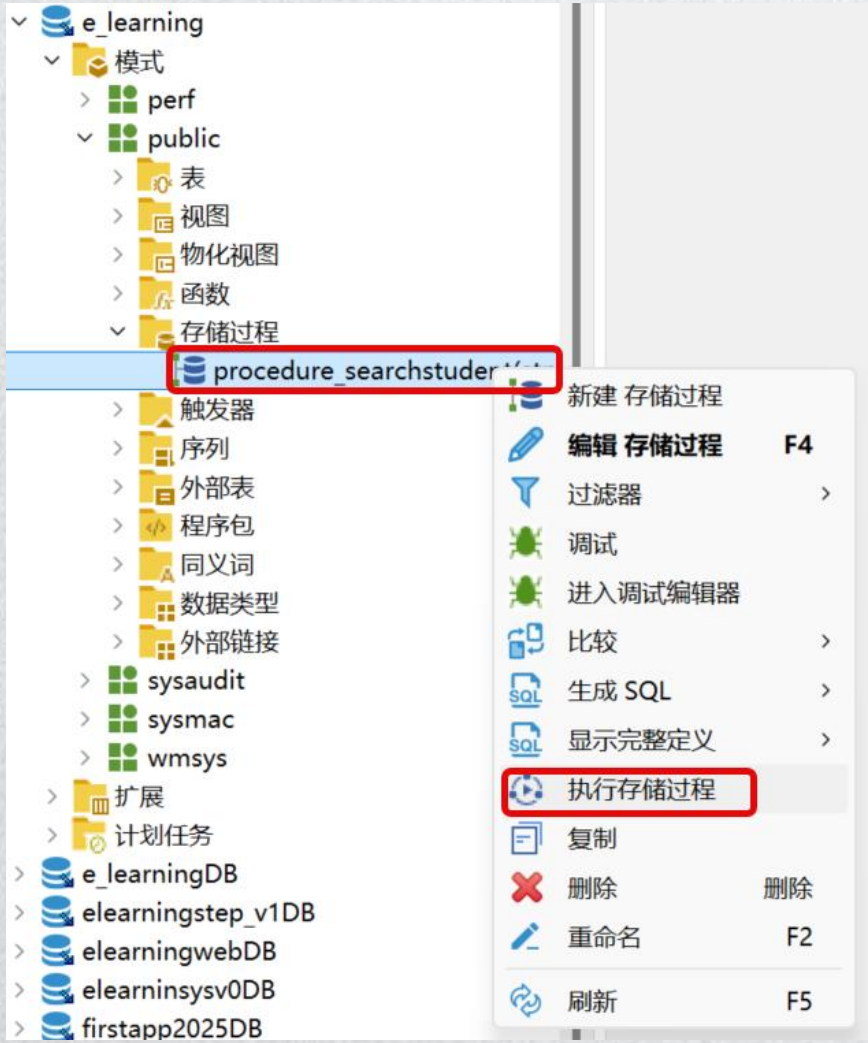
【例】创建带有输入参数的存储过程proc_searchstudent，按姓名查询特定学生信息。





调用存储过程

调用存储过程proc_searchstudent查询学生“林豆豆”的信息。



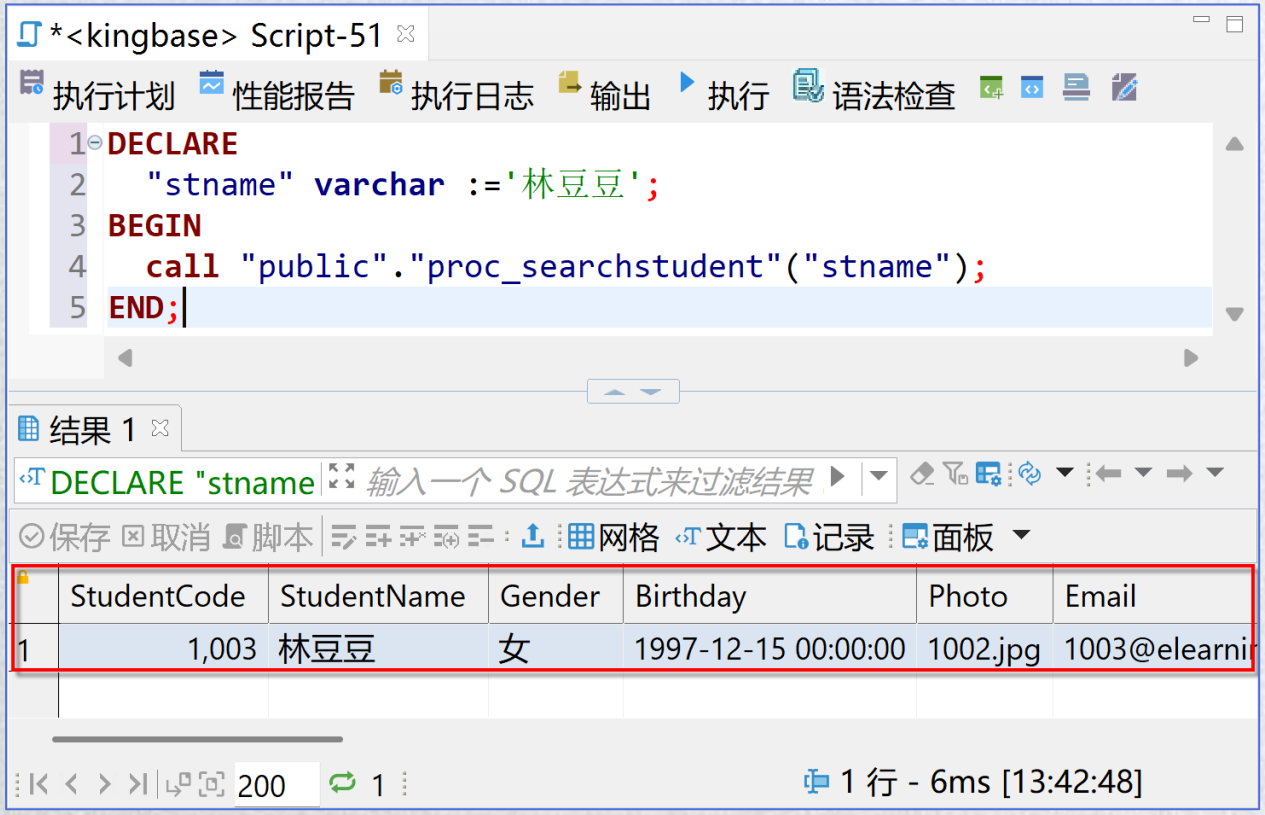
输入参数值，单引号表示字符串

然后，执行产生的调用语句，见下页

调用存储过程

调用存储过程proc_searchstudent，查询学生“林豆豆”的信息。

在查询编辑器中，执行调用存储过程的PL/SQL块



举例3：存储过程（有输入参数和输出参数）

【例】创建带有**输入**和**输出**类型参数的存储过程proc_countstudent，根据课程号统计选课人数。

```
CREATE PROCEDURE proc_countstudent(IN cscode char(4),OUT stnumber int)
        --输入参数cscode表示课程号；输出参数stnumber表示选修总人数
AS
BEGIN
   SELECT COUNT(StudentCode) into stnumber FROM courseenroll
   WHERE Score IS NOT NULL AND CourseCode=cscode; -- 查询结果存stnumber参数
END;
```

调用存储过程，查询“C001”课的选修人数。

```
DECLARE  v_stn int;    -- 声明变量v_stn，用于接收输出参数值
BEGIN
   CALL proc_countstudent('C001', v_stn); -- 调用存储过程
   RAISE NOTICE '选修C001课程的学生人数:%', v_stn; -- 显示选课人数
END;
```

输出

```
选修C001课程的学生人数:5
```

举例4：存储过程（插入操作）

【例】创建一个向subject表中插入学科记录的存储过程 proc_insertsubject。

```
CREATE PROCEDURE proc_insertsubject(sjcode char(3), sjname varchar(10))
--sjcode表示学科号，sjname表示学科名，是输入参数，可省略IN
AS
BEGIN
    INSERT INTO subject VALUES(sjcode, sjname) ; --插入数据记录到subject
END;
```

调用存储过程，完成记录添加

```
CALL proc_insertsubject('S18', '医学') ;
```

举例5：存储过程（修改操作）

【例】创建一个按学号和课程号修改成绩的存储过程proc_updatescore。

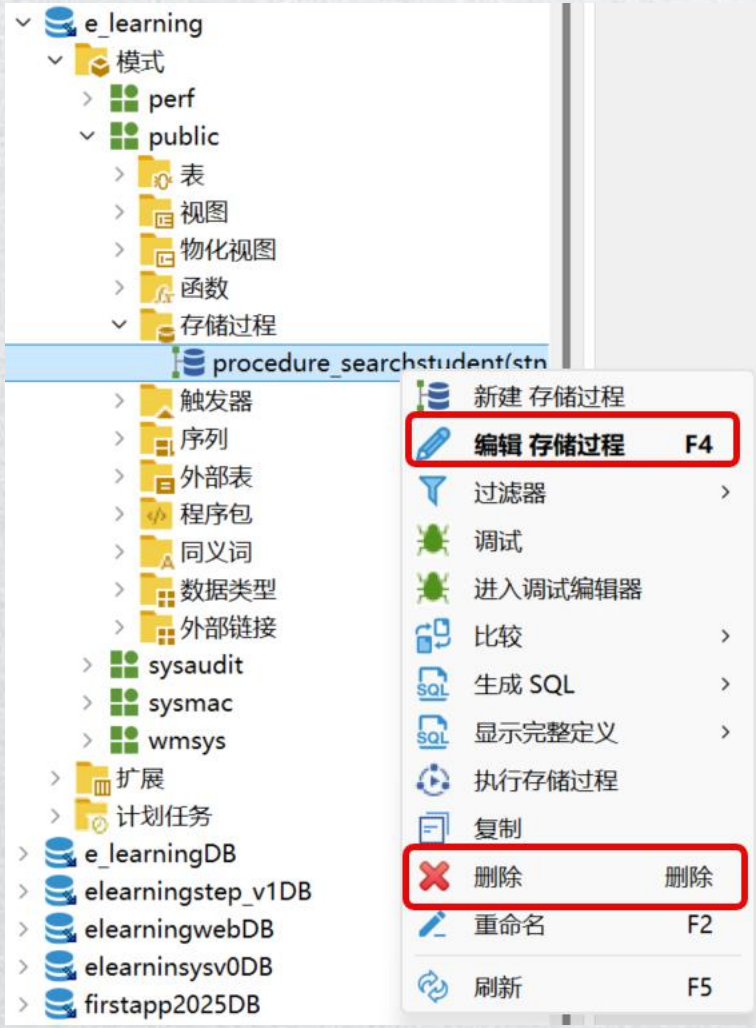
```
CREATE PROC proc_UpdateGrade(stcode int, cscode char(4), sc float)
AS
BEGIN
    UPDATE courseenroll
    SET Score=sc          --以sc值修改成绩
    WHERE StudentCode=stcode AND CourseCode=cscode ;
END;
```

调用存储过程，将1001号学生的C001课程成绩改为80分。

```
CALL proc_updatescore(1001,'C001',80);
```

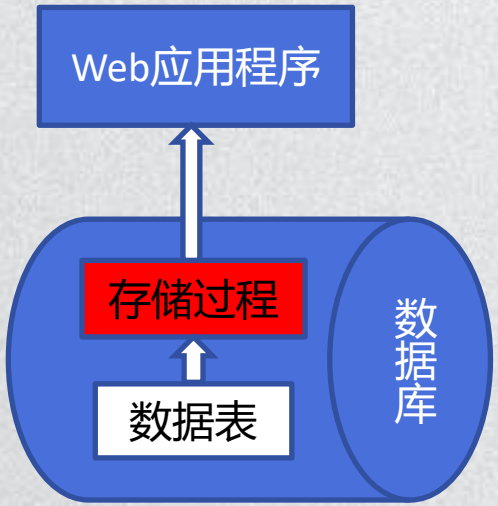
维护存储过程

- ✓ 查看
- ✓ 修改
- ✓ 删除



小结：存储过程

存储过程是完成特定功能的一组SQL语句的集合，
它是数据库的一种数据对象。



优点：

- ✓ **① 可以在一个存储过程中执行多条PL/SQL语句；**
- ✓ **② 可以带有输入参数调用存储过程动态执行；**
- ✓ **③ 存储过程在创建时就在服务器端进行了编译，节省PL/SQL语句的运行时间；**
- ✓ **④ 提供了一种安全机制，可以限制用户执行PL/SQL语句，只允许其访问存储过程。**

缺点：可移植性差

4.2.6 用户自定义函数

用户自定义函数

在KingbaseES中，函数也是存储在数据库服务器上的可被调用的PL/SQL命名块。
与存储过程不同，函数主要用于返回特定的数据，必须有返回值。

创建:

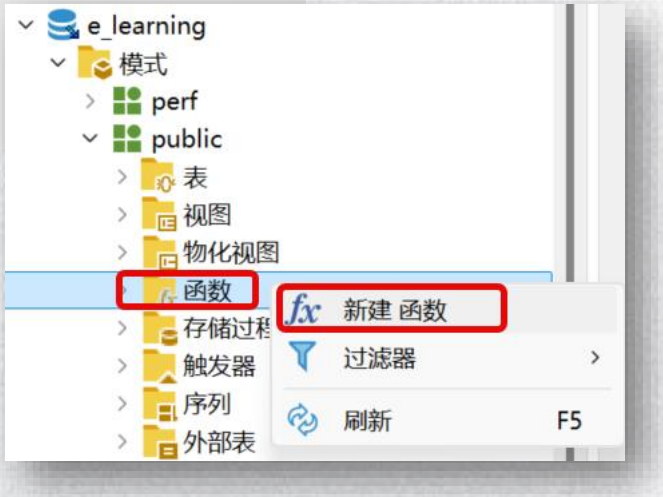
```
CREATE [OR REPLACE] FUNCTION 函数名( [形式参数列表])
RETURNS 返回数据类型
AS
[DECLARE 变量声明]          -- 可选
BEGIN
    <函数体>                -- PL/SQL语句序列
    RETURN 返回值;
    [EXCEPTION 异常处理]    -- 可选
END ;
```

参数名 参数类型 [DEFAULT 默认值]

- 多个参数之间用逗号分隔
- 有默认值的参数，调用时可以不提供实参值。

调用:

```
CALL 函数名(实参值列表);
或 SELECT 函数名(实参值列表);  -- 返回值
   SELECT * FROM 函数名        --返回表
```



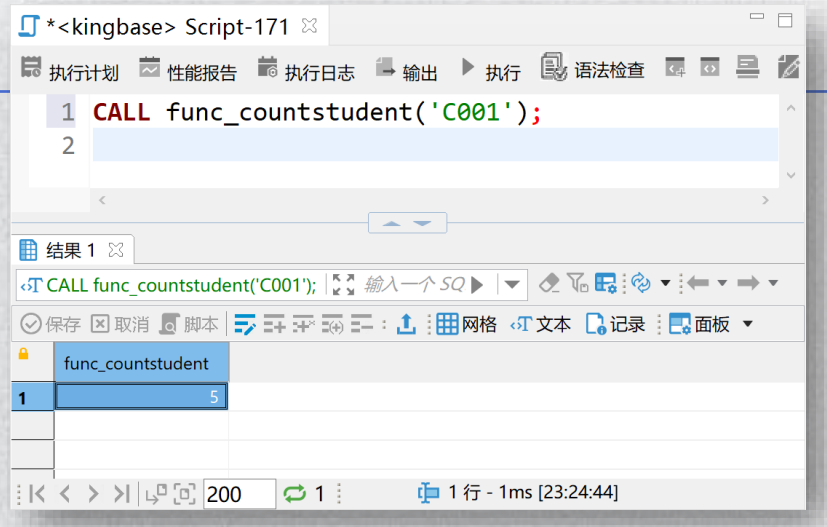
举例1：函数（有参数、返回值）

【例】创建函数func_countstudent，根据课程号统计选课人数，返回选课人数。

```
CREATE FUNCTION func_countstudent(cscode char(4))  --参数cscode表示课程号
RETURNS INTEGER
AS
DECLARE stnumber INTEGER;  --存储选课人数
BEGIN
    SELECT COUNT(StudentCode) into stnumber FROM courseenroll
    WHERE Score IS NOT NULL AND CourseCode=cscode; -- 查询结果存stnumber参数
    RETURN stnumber;
END;
```

调用函数，查询“C001”课的选修人数。

```
CALL func_countstudent('C001');
 -- 或 SELECT func_countstudent('C001');
```



举例2：函数（有参数、**返回表**）

【例】创建函数func_searchstudent，根据输入的学生姓名模糊查询一些学生的信息，包括学号、姓名、性别、电话，并将查询结果以表的形式返回。

```
CREATE FUNCTION func_searchstudent(stname varchar(16))
RETURNS TABLE( studentCode INT,
    studentName VARCHAR(16),
    gender CHAR(1),
    email VARCHAR(30))   --RETURNS TABLE定义返回的表结构
AS
BEGIN
   RETURN QUERY  --返回查询结果
   SELECT studentCode, studentName,Gender,Email FROM student
   WHERE StudentName LIKE '%' || stname || '%';
END;
```

返回的表结构与此处的字段名称和类型一致

这是一个句子！

||表示字符串拼接

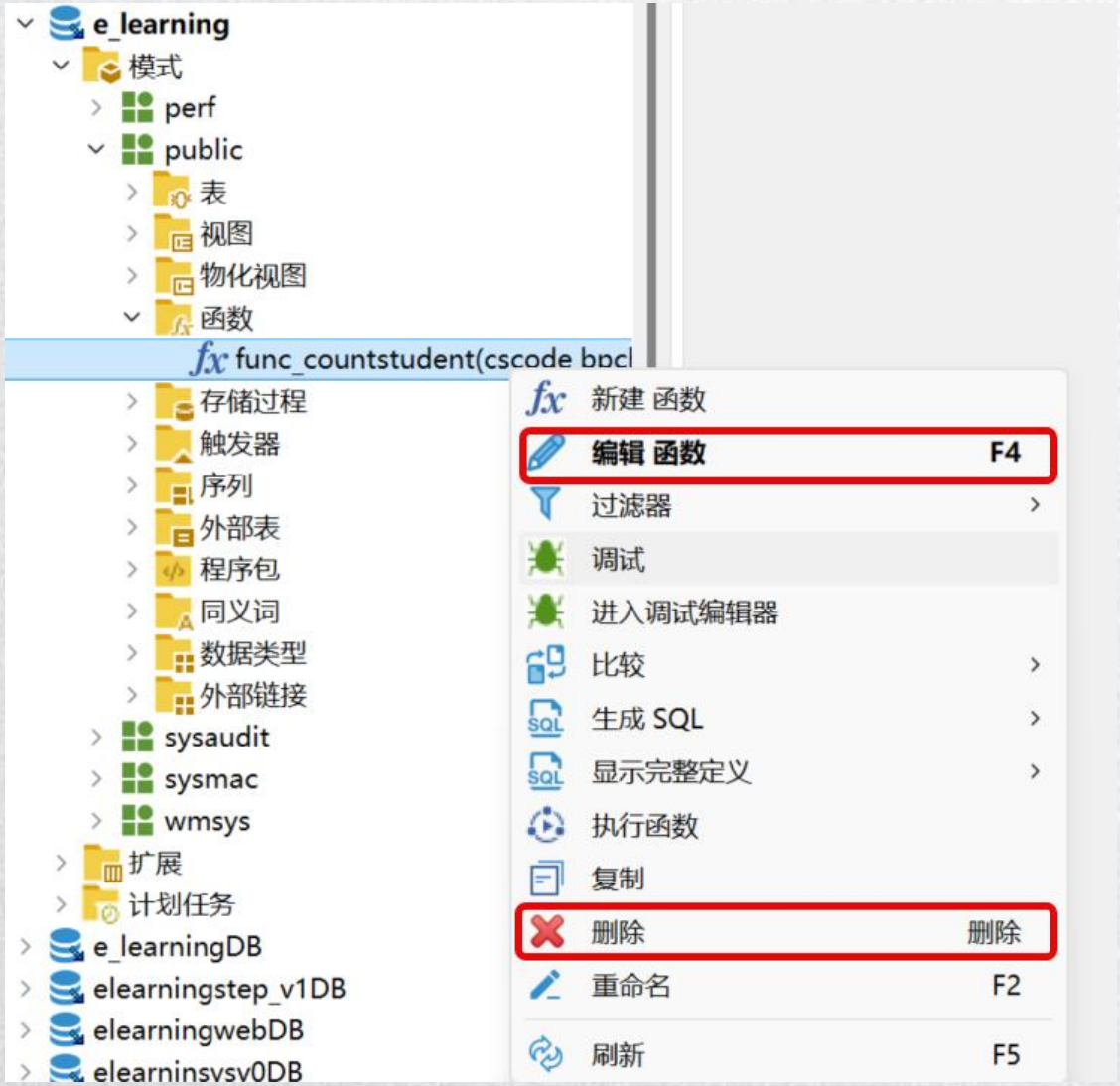
调用函数

```
SELECT * FROM func_searchstudent('');        -- 查询所有学生的信息
SELECT * FROM func_searchstudent('张');      -- 查询姓张的学生的信息
SELECT * FROM func_searchstudent('张小宝');        -- 查询学生张小宝的
```

返回表的调用只能用SELECT语句

维护自定义函数

- ✓ 查看
- ✓ 修改
- ✓ 删除



4.2.7 触发器

为什么需要触发器?

例 当某生选了某门课程后，需对数据库中的数据做以下修改：

- 选课表中插入一条记录
- 将该课程的选课人数增加1

选课人数的更新能不能自动完成呢?

数据库中的部分关系模式：

学生（学号，姓名，...）

选课（学号，课程号，成绩）

课程（课程号，课程名，选课人数，...）

触发器一般用来监视数据库，以便及时进行一些数据维护工作。

如，

- ✓ 实现比数据完整性约束**更为复杂**的其他限制；
- ✓ **级联**修改数据库中相关表的数据；
- ✓ 及时取消不合适的更新操作，防止恶意或错误地使用数据库。

触发器的定义

触发器是一种特殊的存储过程。既不能被程序调用，也不能接收参数，只能由数据库的特定事件来触发（触发后**自动执行，无需写调用语句**）。

更新操作
- Insert
- Update
- Delete

设置触发器需要指明以下内容：

1. 建立在哪个数据库对象上？
2. 什么条件下触发？
3. 触发后执行的动作

使用 SQL语句创建触发器

```
CREATE TRIGGER 触发器名
{BEFORE|AFTER |INSTEAD OF} {INSERT|UPDATE|DELETE}  ON 表名
[FOR EACH ROW| FOR EACH STATEMENT]
[WHEN (条件)]
{EXECUTE {FUNCTION| PROCEDURE}函数名(参数)  |  AS < PL/SQL语句块> };
```

触发时间

触发事件

需要建立触发器的表或视图

触发范围：行级或语句级

触发条件

触发后执行的存储过程、自定义函数或语句块

说明：

① BEFORE | AFTER | INSTEAD OF: 指定触发器在事件之前、之后或代替事件执行。视图只支持INSTEAD OF。

② INSERT | UPDATE | DELETE: 指定触发器响应的事件类型。

③ FOR EACH ROW | FOR EACH STATEMENT: 指定触发器是行级触发器还是语句级触发器。

FOR EACH ROW:行级触发。每行触发一次，即任意一条记录上的相关操作都能触发一次触发器。

FOR EACH STATEMENT:语句级触发（默认）。每条 SQL 语句触发一次，即无论语句影响多少行数据，触发器只触发一次。

④ WHEN (条件): 可选的触发条件，只有满足条件时触发器才会执行。

⑤ EXECUTE {FUNCTION| PROCEDURE}函数名(参数): 触发时执行的函数或存储过程。

⑥ PL/SQL语句块：触发动作体，说明触发后执行的语句。

提示：每个触发器只能和一个触发事件相关，每个事件只支持一个触发器。

触发器的临时表——OLD表 和 NEW 表

PL/SQL语句块中经常需要用到 *表更新前* 或 *更新后* 的数据。

触发器工作时**自动**创建和管理两个临时表 **NEW**和 **OLD**，用于记录数据变动情况。

更新操作	NEW表	OLD表
Insert	存新插入的记录	×
Delete	×	存被删的记录
Update	存修改后的记录	存修改前的记录

某列的值用“NEW.列名”和“OLD.列名”表示

提示：如果是语句级触发器，则不能在触发动作体的PL/SQL语句块中使用NEW或OLD。

举例：创建触发器

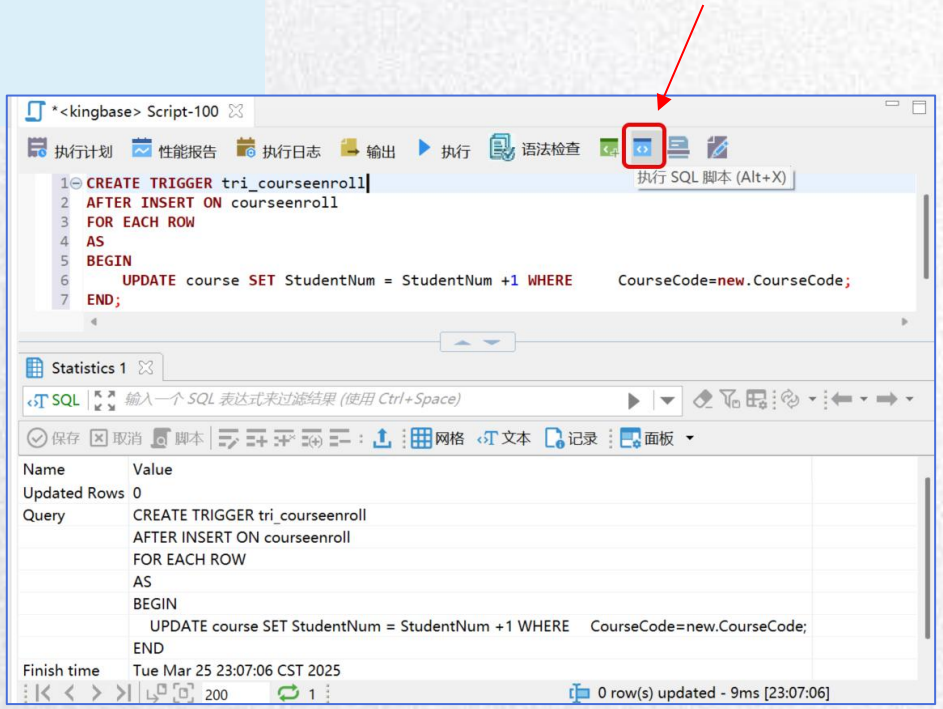
【例】在courseenroll表上建立触发器，当添加新的选课记录时，自动修改course表中的选课人数。

```
CREATE TRIGGER tri_courseenroll
AFTER INSERT ON courseenroll
FOR EACH ROW
AS
BEGIN
    UPDATE course SET StudentNum = StudentNum +1
    WHERE CourseCode=new.CourseCode;
END;
```

测试触发器执行情况：
在courseenroll表中添加一条记录，如

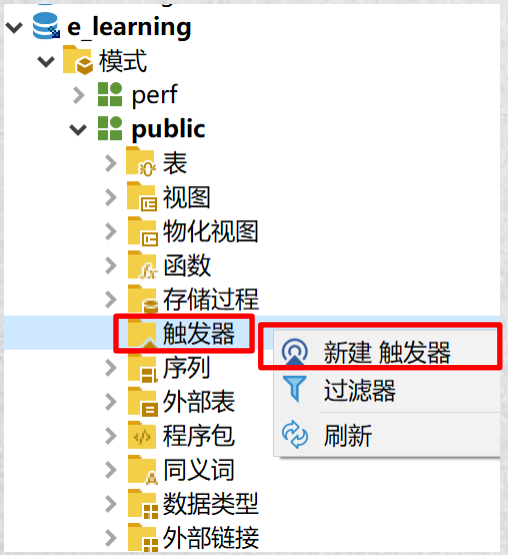
```
Insert into courseenroll(studentcode,coursecode)
values(1001, 'C004')
```

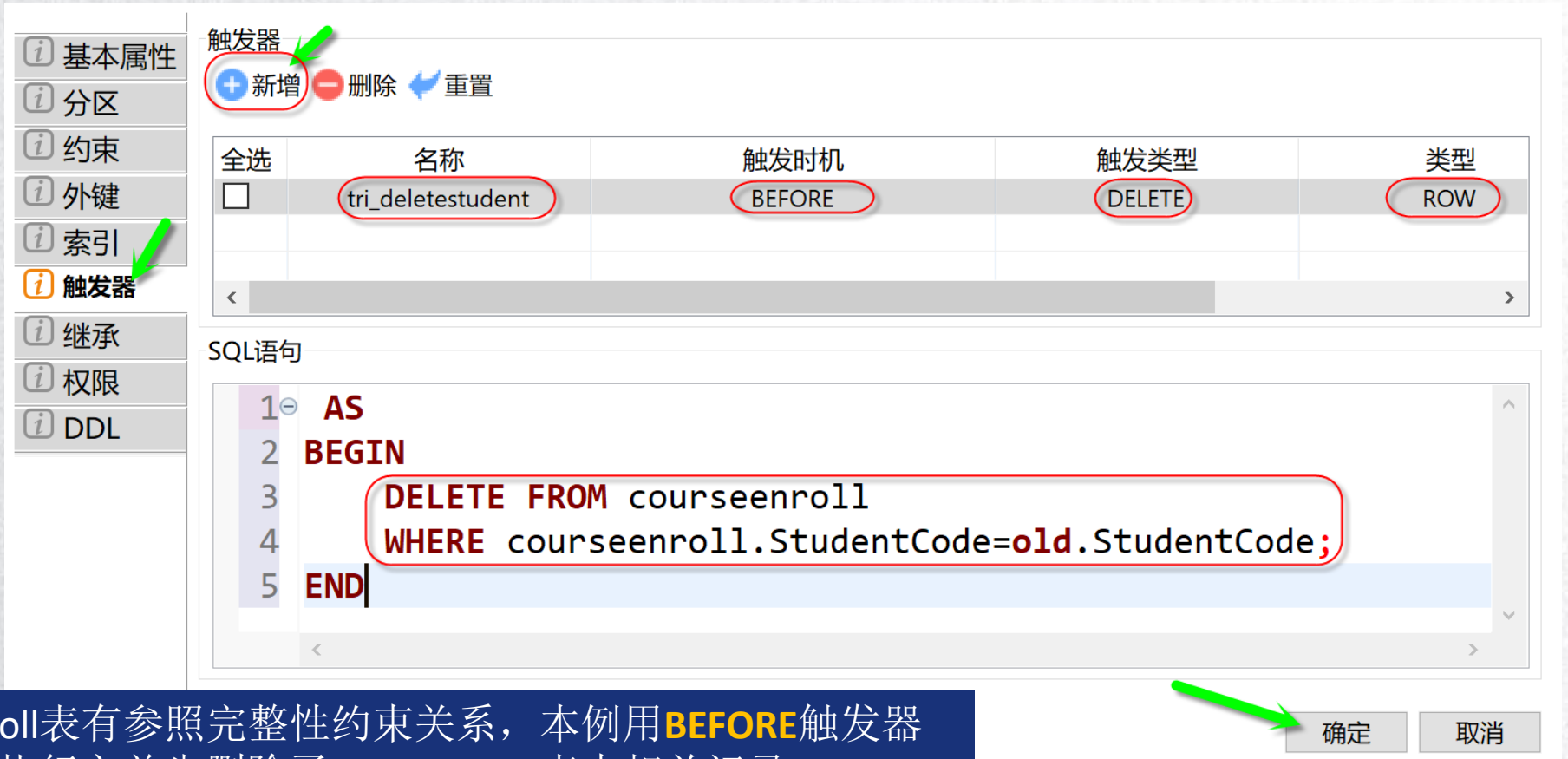
观察添加前、后course表中相关课程的选课人数。



使用KStudio数据库对象管理工具创建触发器

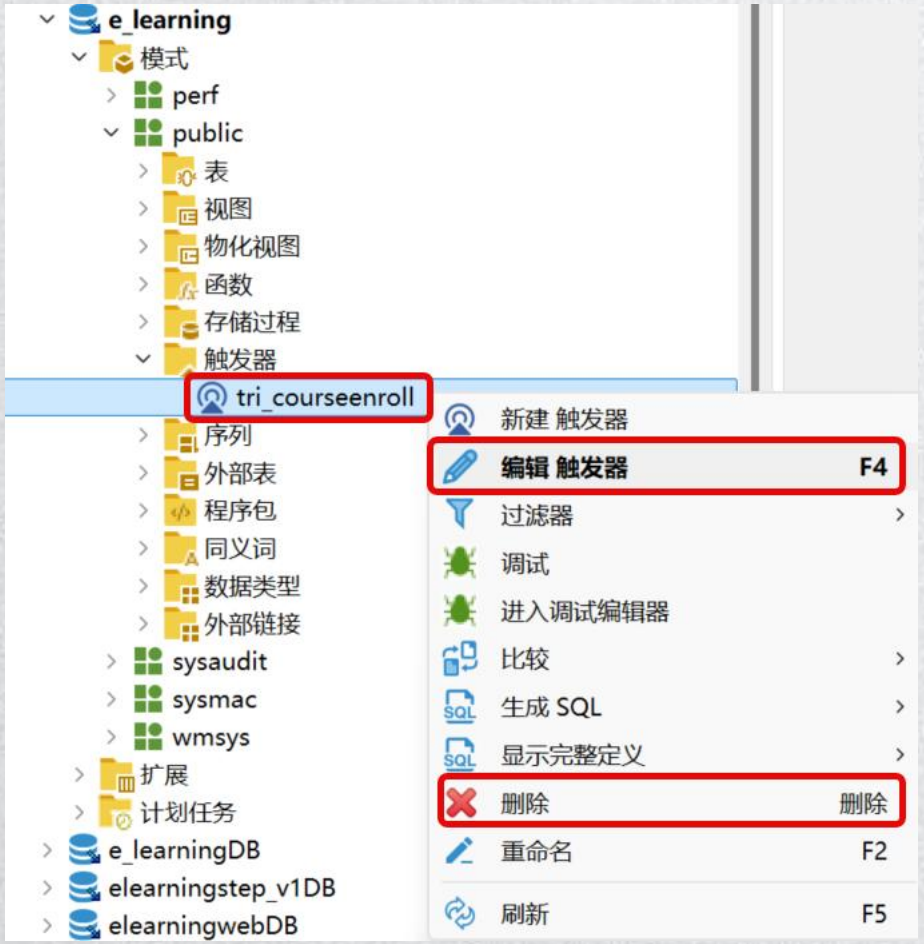
【例】在student表上创建一个触发器tri_deletestudent，当删除一条学生记录时，将该学生在courseenroll表中的选课记录全部删除。



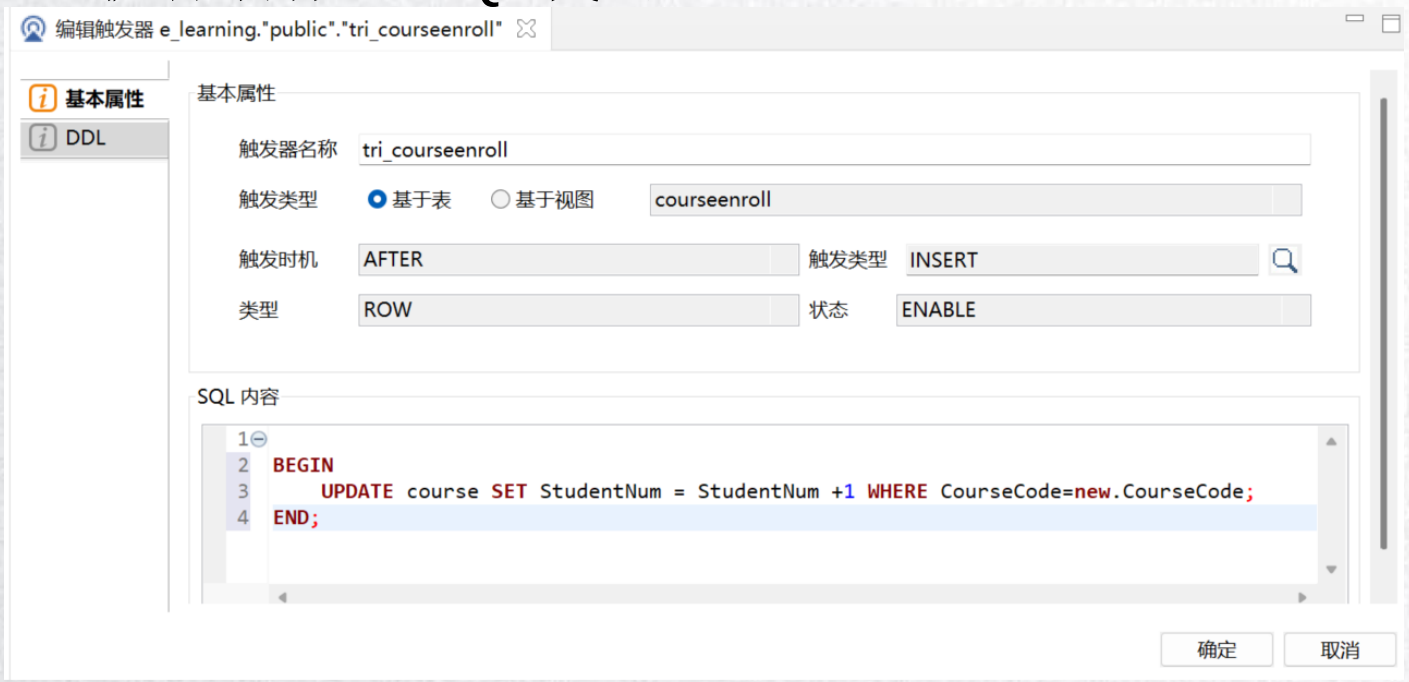


因student表和courseenroll表有参照完整性约束关系，本例用**BEFORE**触发器在student表的删除操作执行之前先删除了courseenroll表中相关记录。
若用AFTER触发器，则无法实现本例功能。

维护触发器



- ✓ 修改名称
- ✓ 触发时间
- ✓ 触发事件
- ✓ 执行动作（SQL代码）



小结：触发器

触发器是一种特殊的存储过程。当对数据库进行更新操作时被触发**自动执行**。

不需要显式调用！

更新操作
- Insert
- Update
- Delete

THANK YOU!