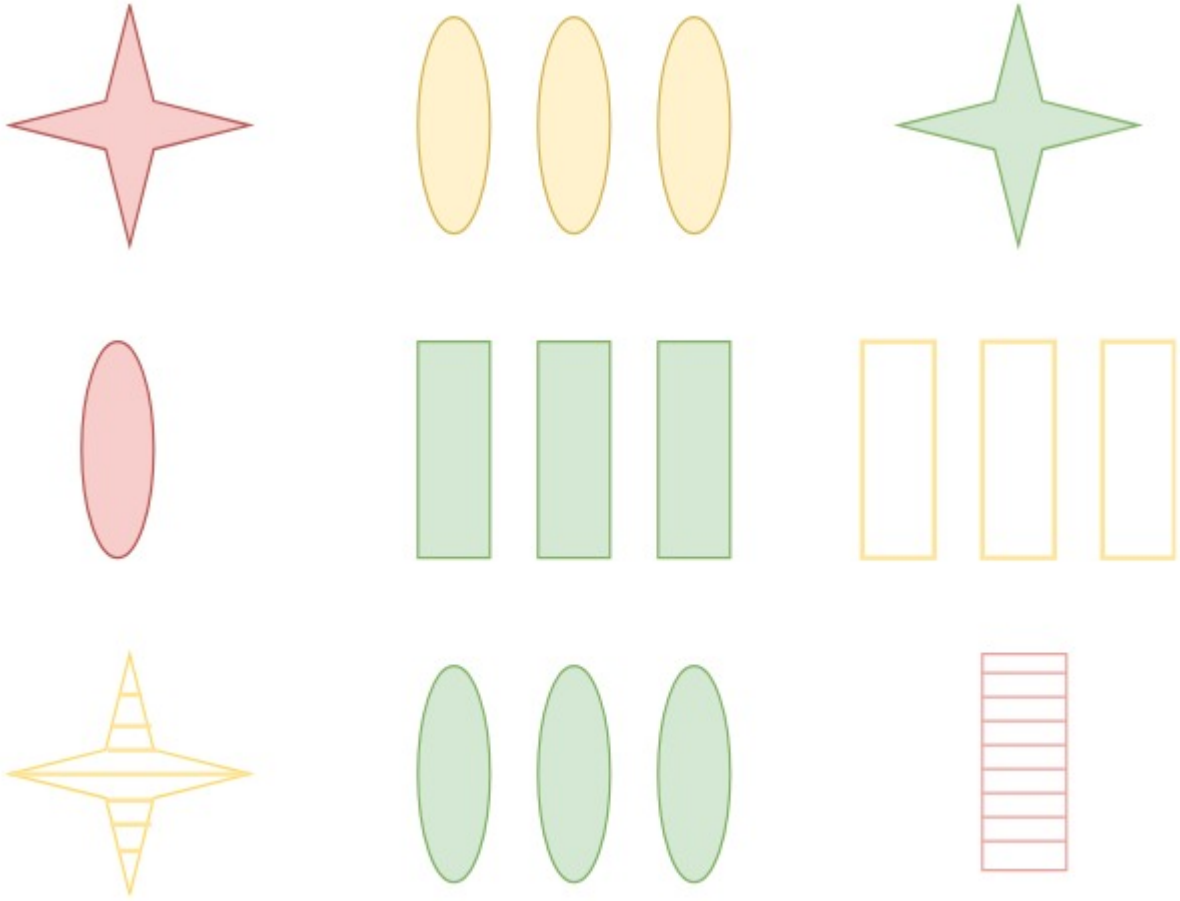
# 16丨游标：当我们需要逐条处理数据时，该怎么做？

2019-07-17 陈旸

我们在编写SQL语句的时候通常是面向集合进行思考，这种思考方式更让我们关注结果集的特征，而不是具体的实现过程。面向集合的思考方式与面向过程的思考方式各有特点，我们该如何理解它们呢？

我们用下面这张图开启今天的学习。这张图中一共有9个图形，每个图形有不同的特征，包括形状、纹理、颜色和个数等。



当我们看到这张图时，有时候会不由自主地按照某个属性进行分类，比如说按照红色分类，那么1、4、9就是一类。这实际上就是属于同一个条件下的查询结果集。或者我们也可以按照物体的个数来划分，比如都有3个物体的，那么对应的就是2、5、6、8，这就是对应着“都包括3个物体”的查询结果集。

你能看出来集合思维更像是从整体的角度来考虑，然后把整个数据集按照不同的属性进行划分，形成不同的子集合。面向集合的思考方式，让我们关注“获取什么”，而不是“如何获取”，这也可以说是SQL与传统编程最大的区别之一，因为SQL本身是以关系模型和集合论为基础的。

然而也有一些情况，我们不需要对查询结果集中的所有数据行都采用相同的处理方式，需要每次处理一行或者一部分行，这时就需要面向过程的编程方法了。游标就是这种编程方式的体现。如果你之前已经有了一些面向过程的编程经验，那么对于游标的理解也会比较容易。

关于游标，你需要掌握以下几个方面的内容：

1. 什么是游标？我们为什么要使用游标？
2. 如何使用游标？使用游标的常用步骤都包括哪些？
3. 如何使用游标来解决一些常见的问题？

## 什么是游标？

在数据库中，游标是个重要的概念，它提供了一种灵活的操作方式，可以让我们从数据结果集中

每次提取一条数据记录进行操作。游标让SQL这种面向集合的语言有了面向过程开发的能力。可以说，游标是面向过程的编程方式，这与面向集合的编程方式有所不同。

在SQL中，游标是一种临时的数据库对象，可以指向存储在数据库表中的数据行指针。这里游标充当了指针的作用，我们可以通过操作游标来对数据行进行操作。

比如我们查询了heros数据表中最大生命值大于8500的英雄都有哪些：

```
SELECT id, name, hp_max FROM heros WHERE hp_max > 8500
```

查询结果（4条数据）：


这里我们就可以通过游标来操作数据行，如图所示此时游标所在的行是“白起”的记录，我们也可以在结果集上滚动游标，指向结果集中的任意一行。

### 如何使用游标？

游标实际上是一种控制数据集的更加灵活的处理方式。

如果我们想要使用游标，一般需要经历五个步骤。不同DBMS中，使用游标的语法可能略有不同。

第一步，定义游标。

```
DECLARE cursor_name CURSOR FOR select_statement
```

这个语法适用于MySQL，SQL Server，DB2和MariaDB。如果是用Oracle或者PostgreSQL，需要写成：

```
DECLARE cursor_name CURSOR IS select_statement
```

要使用SELECT语句来获取数据结果集，而此时还没有开始遍历数据，这里select_statement代表的是SELECT语句。

下面我用MySQL举例讲解游标的使用，如果你使用的是其他的RDBMS，具体的游标语法可能略有差异。我们定义一个能够存储heros数据表中的最大生命值的游标，可以写为：

```
DECLARE cur_hero CURSOR FOR
 SELECT hp_max FROM heros;
```

第二步，打开游标。

```
OPEN cursor_name
```

当我们定义好游标之后，如果想要使用游标，必须先打开游标。打开游标的时候SELECT语句的查询结果集就会送到游标工作区。

第三步，从游标中取得数据。

```
FETCH cursor_name INTO var_name ...
```

这句的作用是使用cursor_name这个游标来读取当前行，并且将数据保存到var_name这个变量中，游标指针指到下一行。如果游标读取的数据行有多个列名，则在INTO关键字后面赋值给多个变量名即可。

第四步，关闭游标。

```
CLOSE cursor_name
```

有OPEN就会有CLOSE，也就是打开和关闭游标。当我们使用完游标后需要关闭掉该游标。关闭游标之后，我们就不能再检索查询结果中的数据行，如果需要检索只能再次打开游标。

最后一步，释放游标。

```
DEALLOCATE cursor_namec
```

有DECLARE就需要有DEALLOCATE，DEALLOCATE的作用是释放游标。我们一定要养成释放游标的习惯，否则游标会一直存在于内存中，直到进程结束后才会自动释放。当你不需要使用游标的时候，释放游标可以减少资源浪费。

上面就是5个常用的游标步骤。我来举一个简单的例子，假设我想用游标来扫描heros数据表中的数据行，然后累计最大生命值，那么该怎么做呢？

我先创建一个存储过程calc_hp_max，然后在存储过程中定义游标cur_hero，使用FETCH获取每一行的具体数值，然后赋值给变量hp，再用变量hp_sum做累加求和，最后再输出hp_sum，代码如下：

```
CREATE PROCEDURE `calc_hp_max`()
BEGIN
    -- 创建接收游标的变量
    DECLARE hp INT;
    -- 创建总数变量
    DECLARE hp_sum INT DEFAULT 0;
    -- 创建结束标志变量
    DECLARE done INT DEFAULT false;
    -- 定义游标
    DECLARE cur_hero CURSOR FOR SELECT hp_max FROM heros;

    OPEN cur_hero;
    read_loop:LOOP
    FETCH cur_hero INTO hp;
    SET hp_sum = hp_sum + hp;
    END LOOP;
    CLOSE cur_hero;
    SELECT hp_sum;
END
```

你会发现执行call calc_hp_max()这一句的时候系统会提示1329错误，也就是在LOOP中当游标没有取到数据时会报的错误。

当游标溢出时（也就是当游标指向到最后一行数据后继续执行会报的错误），我们可以定义一个

continue的事件，指定这个事件发生时修改变量done的值，以此来判断游标是否已经溢出，即：

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;
```

同时在循环中我们需要加上对done的判断，如果游标的循环已经结束，就需要跳出read_loop循环，完善的代码如下：

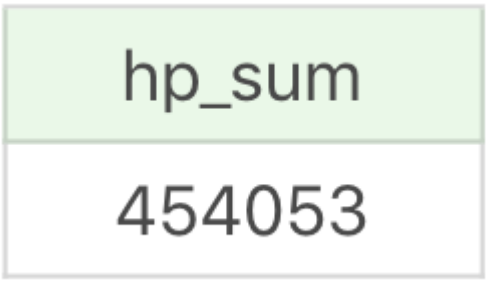
```
CREATE PROCEDURE `calc_hp_max`()
BEGIN
    -- 创建接收游标的变量
    DECLARE hp INT;

    -- 创建总数变量
    DECLARE hp_sum INT DEFAULT 0;
    -- 创建结束标志变量
   DECLARE done INT DEFAULT false;
    -- 定义游标
    DECLARE cur_hero CURSOR FOR SELECT hp_max FROM heros;
    -- 指定游标循环结束时的返回值
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;

    OPEN cur_hero;
    read_loop:LOOP
    FETCH cur_hero INTO hp;
    -- 判断游标的循环是否结束
    IF done THEN
            LEAVE read_loop;
    END IF;

    SET hp_sum = hp_sum + hp;
    END LOOP;
    CLOSE cur_hero;
    SELECT hp_sum;
END
```

运行结果（1行数据）：




在游标中的循环中，除了使用LOOP循环以外，你还可以使用REPEAT...UNTIL..以及WHILE循环。它们同样需要设置CONTINUE事件来处理游标溢出的情况。

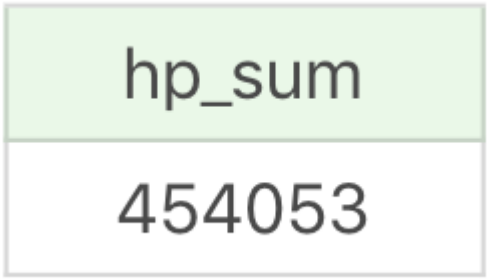
所以你能看出，使用游标可以让我们对SELECT结果集中的每一行数据进行相同或者不同的操作，从而很精细化地管理结果集中的每一条数据。

**使用游标来解决一些常见的问题**

我刚才讲了一个简单的使用案例，实际上如果想要统计hp_sum，完全可以通过SQL语句来完成，比如：

```
SELECT SUM(hp_max) FROM heros
```

运行结果（1行数据）：




那么游标都有什么用呢？

当你需要处理一些复杂的数据行计算的时候，游标就会起到作用了。我举个例子，还是针对heros数据表，假设我们想要对英雄的物攻成长（对应attack_growth）进行升级，在新版本中大范围提升英雄的物攻成长数值，但是针对不同的英雄情况，提升的幅度也不同，具体提升的方式如下。

如果这个英雄原有的物攻成长小于5，那么将在原有基础上提升7%-10%。如果物攻成长的提升空间（即最高物攻attack_max-初始物攻attack_start）大于200，那么在原有的基础上提升10%；如果物攻成长的提升空间在150到200之间，则提升8%；如果物攻成长的提升空间不足150，则提升7%。

如果原有英雄的物攻成长在5– 10之间，那么将在原有基础上提升5%。

如果原有英雄的物攻成长大于10，则保持不变。

以上所有的更新后的物攻成长数值，都需要保留小数点后3位。

你能看到上面这个计算的情况相对复杂，实际工作中你可能会遇到比这个更加复杂的情况，这时你可以采用面向过程的思考方式来完成这种任务，也就是说先取出每行的数值，然后针对数值的不同情况采取不同的计算方式。

针对上面这个情况，你自己可以用游标来完成转换，具体的代码如下：

```
CREATE PROCEDURE `alter_attack_growth`()
BEGIN
    -- 创建接收游标的变量
    DECLARE temp_id INT;
    DECLARE temp_growth, temp_max, temp_start, temp_diff FLOAT;

    -- 创建结束标志变量
    DECLARE done INT DEFAULT false;
    -- 定义游标
    DECLARE cur_hero CURSOR FOR SELECT id, attack_growth, attack_max, attack_start FROM heros;
    -- 指定游标循环结束时的返回值
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;

    OPEN cur_hero;
    FETCH cur_hero INTO temp_id, temp_growth, temp_max, temp_start;
    REPEAT
            IF NOT done THEN
                SET temp_diff = temp_max - temp_start;
                IF temp_growth < 5 THEN
                    IF temp_diff > 200 THEN
                        SET temp_growth = temp_growth * 1.1;
                    ELSEIF temp_diff >= 150 AND temp_diff <=200 THEN
                        SET temp_growth = temp_growth * 1.08;
                    ELSEIF temp_diff < 150 THEN
                        SET temp_growth = temp_growth * 1.07;
                    END IF;
                ELSEIF temp_growth >=5 AND temp_growth <=10 THEN
                    SET temp_growth = temp_growth * 1.05;
                END IF;
                UPDATE heros SET attack_growth = ROUND(temp_growth,3) WHERE id = temp_id;
            END IF;
    FETCH cur_hero INTO temp_id, temp_growth, temp_max, temp_start;
    UNTIL done = true END REPEAT;

    CLOSE cur_hero;
END
```

这里我创建了alter_attack_growth这个存储过程，使用了REPEAT..UNTIL..的循环方式，针对不同的情况计算了新的物攻成长temp_growth，然后对原有的attack_growth进行了更新，最后调用call alter_attack_growth();执行存储过程。

有一点需要注意的是，我们在对数据表进行更新前，需要备份之前的表，我们可以将备份后的表命名为heros_copy1。更新完heros数据表之后，你可以看下两张表在attack_growth字段上的对比，我们使用SQL进行查询：

```
SELECT heros.id, heros.attack_growth, heros_copy1.attack_growth FROM heros JOIN heros_copy1 WHERE her
```

运行结果（69条记录）：

| id | attack_growth | attack_growth(1) |
| --- | --- | --- |
| 10000 | 11.57 | 11.57 |
| 10001 | 11 | 11 |
| 10002 | 10.57 | 10.57 |
| 10003 | 8.775 | 8.357 |
| ..... | ..... | ..... |
| 10068 | 15.86 | 15.86 |

通过前后两张表的attack_growth对比你也能看出来，存储过程通过游标对不同的数据行进行了更新。

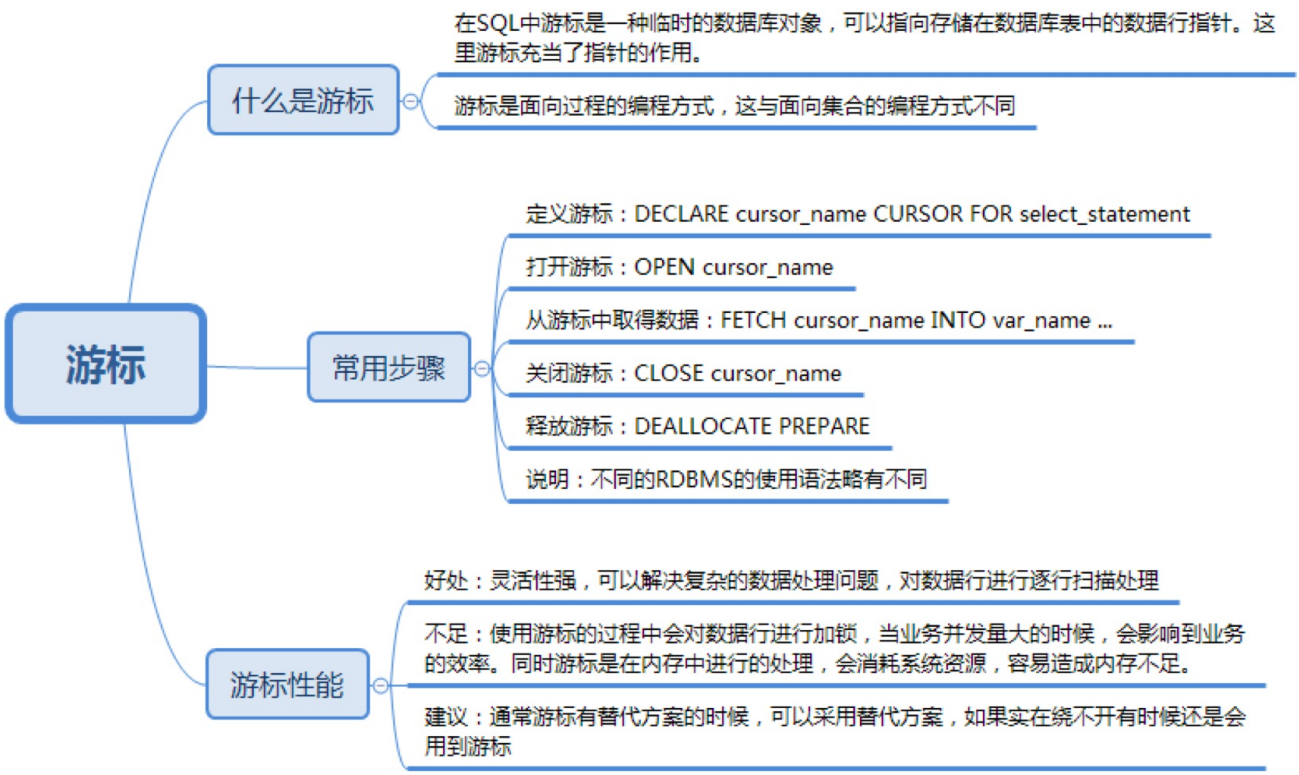
需要说明的是，以上代码适用于MySQL，如果在SQL Server或Oracle中，使用方式会有些差别。

## 总结

今天我们讲解了如何在SQL中使用游标，游标实际上是面向过程的思维方式，与面向集合的思维方式不同的地方在于，游标更加关注“如何执行”。我们可以通过游标更加精细、灵活地查询和管理想要的数据行。

有的时候，我们需要找特定数据，用SQL查询写起来会比较困难，比如两表或多表之间的嵌套循环查找，如果用JOIN会非常消耗资源，效率也可能不高，而用游标则会比较高效。

虽然在处理某些复杂的数据情况下，使用游标可以更灵活，但同时也会带来一些性能问题，比如在使用游标的过程中，会对数据行进行加锁，这样在业务并发量大的时候，不仅会影响业务之间的效率，还会消耗系统资源，造成内存不足，这是因为游标是在内存中进行的处理。如果有游标的替代方案，我们可以采用替代方案。




我们今天讲解了游标，你能用自己的语言介绍下游标的作用吗？另外，我们之前提到过，SQL本身是一门结构化查询语言，但我们也可以在SQL的基础上进行面向过程的开发，完成较为复杂的功能，你能说一下面向过程和面向集合这两种编程方式的区别吗？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

精选留言

ahazxc 5

想问下老师，在第二个例子中，第一次游标获取数据后，按条件更新数据后，为啥还要再fetch一遍呢？

2019-07-17

ABC 3

老师这些例子是在MySQL的客户端命令行里面运行吗?在navicat里面运行好几个例子都没反应..

2019-07-18

阿恺 3

例子运行会报告：ERROR 1243 (HY000): Unknown prepared statement handler (cur_hero) given to DEALLOCATE PREPARE。 在MYSQL官网上的例子（https://dev.mysql.com/doc/refman/5.7/en/cursors.html）并没有 DEALLOCATE PREPARE 这一步。我理解DECLARE只是做一个变量声明，就如同DECLARE hp INT。过程结束，应该就是生命周期的结束。精简例子，只有声明CURSOR和释放CURSOR，同样报错。

2019-07-18

Y024 3

面向集合思维方式的训练，有个在线的游戏，可以帮忙大家：
www.setgame.com/puzzle/set.htm

关于面向集合的思考，《Oracle SQL高级编程》一书里有个专门的章节介绍：
http://m.ituring.com.cn/article/472

2019-07-17

林彦 1

面向集合是通过属性来划分数据获取结果，面向过程是通过对每个/条数据进行处理。
请问老师文中提到的游标的比较常见的，或能避免其缺点的替代方式有哪些？有哪些场景又必须要用游标？
谢谢。

2019-07-17

Danpier 0

DEALLOCATE PREPARE 这个查了官方文档5.5和8.0，都说明这个语句是用于释放 PREPARE 语句，而没有提到释放游标，谷歌也没查到相关用法，应该是说错了吧？
第二个例子 FETCH 放在repeat内部第一行就行，不知为什么要在循环外和循环尾部都加一句？

2019-07-25

Ronnyz 0

在Navicat运行例子出现报错ERROR 1243，解决办法就是去掉 DEALLOCATE PREPARE cur_hero

2019-07-24


大斌 0

我理解游标的作用：
在进行一些复杂计算的时候可以使用游标，因为自带的没有这么复杂的计算，需要自己进行定制，

我理解的面向过程和面向集合的编程方式的区别：
面向过程是结构化编程，是对解决问题步骤的分析
面向集合是属于抽象式的，把解决步骤模块化来复用

2019-07-22

自主 0

我想问问游标溢出，什么是游标溢出，标准是什么？

2019-07-22

linus 0

第一个示例报如下错，麻烦大佬瞅一下，给个回复，谢谢
[SQL]call calc_hp_max(10)

[Err] 1318 - Incorrect number of arguments for PROCEDURE help.calc_hp_max; expected 0, g

ot 1

2019-07-19

华夏

0

IF temp_growth < 5 THEN
IF temp_diff > 200 THEN
SET temp_growth = temp_growth * 1.1;
ELSEIF temp_diff >= 150 THEN
SET temp_growth = temp_growth * 1.08;
ELSEIF temp_diff < 150 THEN
SET temp_growth = temp_growth * 1.07;
END IF;
ELSEIF temp_growth < 10 THEN
SET temp_growth = temp_growth * 1.05;
条件判断也可以写成这样哈。因为上文if>200，elseif就默认<=200。后面同理。

2019-07-19

算盘Man

0

DECLARE cur_hero CURSOR FOR
SELECT hp_max FROM heros
> 1064 - You have an error in your SQL syntax; check the manual that corresponds to your My SQL server version for the right syntax to use near 'DECLARE cur_hero CURSOR FOR SELECT hp_max FROM heros' at line 1
> 时间: 0s
请教陈老师 为什么复制您的代码会在Navicat里会报错啊 谢谢

2019-07-19

ttttt

0

运行最后一个例子alter_attack_growth时，结果运行正确。看着逻辑也对。
报这个错误。
1243 - Unknown prepared statement handler (cur_hero) given to DEALLOCATE PREPARE, Time: 0.070000s

2019-07-18

liuyyy

0

运行第二个例子时，为啥affected rows为0呢

2019-07-18

夜路破晓

0

简单理解就是把操作鼠标的动作用存储过程封装起来从而实现面对大批量数据重复操作的自动化。
这种情况我选择用编程比如python来解决，除非是那种有定期更新要求的定制宽表，可以考虑在定制宽表的基础加个游标以便实现自动化定期更新的要求。

2019-07-17

时间是最真的答案

0

使用游标会造成性能问题，导致目前很少有公司在线上的系统中使用游标吧，反正我没有遇到使用游标的，可能是我见识不够，有公司使用游标的朋友吗，可以讨论一下游标的使用场景等

2019-07-17

江南皮革厂研发中心保安队长

0

老师，declare游标是不是只能在存储过程中使用啊，我在外面declare都报语法错误。

2019-07-17

NO.9

0

这个跟exec sql curser有什么区别？

2019-07-17

我行我素

0

不使用游标，使用case when then 也能完成物攻的那个sql吧

2019-07-17

cricket1981

0

游标面向过程处理是否可以类比流处理，而sql面向集合处理类比批处理？

2019-07-17