15 | 存储过程：如何提高程序的性能和安全性？

朱晓峰 2021-04-13

讲述：朱晓峰 大小：9.25M 时长：10:05

你好，我是朱晓峰。今天呢，我们来聊一聊 MySQL 的存储过程。

在我们的超市项目中，每天营业结束后，超市经营者都要计算当日的销量，核算成本和毛利等营业数据，这也就意味着每天都要做重复的数据统计工作。其实，这种数据量大，而且计算过程复杂的场景，就非常适合使用存储过程。

简单来说呢，存储过程就是把一系列 SQL 语句预先存储在 MySQL 服务器上，需要执行的时候，客户端只需要向服务器端发出调用存储过程的命令，服务器端就可以把预先存储好的这一系列 SQL 语句全部执行。

这样一来，不仅执行效率非常高，而且客户端不需要把所有的 SQL 语句通过网络发给服务器，减少了 SQL 语句暴露在网上的风险，也提高了数据查询的安全性。

今天，我就借助真实的超市项目，给你介绍一下如何创建和使用存储过程，帮助你提升查询的效率，并且让你开发的应用更加简洁安全。

如何创建存储过程？

在创建存储过程的时候，我们需要用到关键字 CREATE PROCEDURE。具体的语法结构如下：

```
CREATE PROCEDURE 存储过程名 （[ IN | OUT | INOUT] 参数名称 类型）程序体
```

接下来，我以超市的日结计算为例，给你讲一讲怎么创建存储过程。当然，为了方便你理解，我对计算的过程进行了简化。

假设在日结计算中，我们需要统计每天的单品销售，包括销售数量、销售金额、成本、毛利、毛利率等。同时，我们还要把计算出来的结果存入单品统计表中。

这个计算需要用到几个数据表，我分别来展示下这些表的基本信息。

销售单明细表（demo.transactiondetails）中包括了每笔销售中的商品编号、销售数量、销售价格和销售金额。

```
mysql> SELECT *
-> FROM demo.transactiondetails;
+---------------+------------+----------+------------+------------+
| transactionid | itemnumber | quantity | salesprice | salesvalue |
+---------------+------------+----------+------------+------------+
| 1 | 1 | 1.000 | 89.00 | 89.00 |
| 1 | 2 | 2.000 | 5.00 | 10.00 |
| 2 | 1 | 2.000 | 89.00 | 178.00 |
| 3 | 2 | 10.000 | 5.00 | 50.00 |
| 3 | 3 | 3.000 | 15.00 | 45.00 |
+---------------+------------+----------+------------+------------+
5 rows in set (0.00 sec)
```

销售单头表（demo.transactionhead）中包括流水单号、收款机编号、会员编号、操作员编号、交易时间。

```
mysql> SELECT *
-> FROM demo.transactionhead;
+---------------+------------------+-----------+----------+------------+------
| transactionid | transactionno | cashierid | memberid | operatorid | transdat
+---------------+------------------+-----------+----------+------------+------
| 1 | 0120201201000001 | 1 | 1 | 1 | 2020-12-01 00:00:00 |
| 2 | 0120201201000002 | 1 | NULL | 1 | 2020-12-01 00:00:00 |
| 3 | 0120201202000001 | 1 | NULL | 1 | 2020-12-02 00:00:00 |
+---------------+------------------+-----------+----------+------------+------
3 rows in set (0.00 sec)
```

商品信息表（demo.goodsmaster）中包括商品编号、商品条码、商品名称、规格、单位、售价和平均进价。

```
mysql> SELECT *
-> FROM demo.goodsmaster;
+------------+---------+-----------+---------------+------+------------+------
| itemnumber | barcode | goodsname | specification | unit | salesprice | avgim
+------------+---------+-----------+---------------+------+------------+------
| 1 | 0001 | 书 | NULL | 本 | 89.00 | 33.50 |
| 2 | 0002 | 笔 | NULL | 支 | 5.00 | 3.50 |
| 3 | 0003 | 胶水 | NULL | 瓶 | 15.00 | 11.00 |
+------------+---------+-----------+---------------+------+------------+------
3 rows in set (0.00 sec)
```

存储过程会用刚刚的三个表中的数据进行计算，并且把计算的结果存储到下面的这个单品统计表中。

```
mysql> DESCRIBE demo.dailystatistics;
+-------------+---------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+----------------+
| id | int | NO | PRI | NULL | auto_increment |
| itemnumber | int | YES | MUL | NULL | |
| quantity | decimal(10,3) | YES | | NULL | |
| actualvalue | decimal(10,2) | YES | | NULL | |
| cost | decimal(10,2) | YES | | NULL | |
| profit | decimal(10,2) | YES | | NULL | |
| profitratio | decimal(10,4) | YES | | NULL | |
| salesdate | datetime | YES | MUL | NULL | |
+-------------+---------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

我们现在就来创建一个存储过程，完成单品销售统计的计算。我来讲一讲具体的思路。

第一步，我们把 SQL 语句的分隔符改为"//"。因为存储过程中包含很多 SQL 语句，如果不修改分隔符的话，MySQL 会在读到第一个 SQL 语句的分隔符";"的时候，认为语句结束并且执行，这样就会导致错误。

第二步，我们来创建存储过程，把要处理的日期作为一个参数传入（关于参数，下面我会具体讲述）。同时，用 BEGIN 和 END 关键字把存储过程中的 SQL 语句包裹起来，形成存储过程的程序体。

第三步，在程序体中，先定义 2 个数据类型为 DATETIME 的变量，用来记录要计算数据的起始时间和截止时间。

第四步，删除保存结果数据的单品统计表中相同时间段的数据，目的是防止数据重复。

第五步，计算起始时间和截止时间内单品的销售数量合计、销售金额合计、成本合计、毛利和毛利率，并且把结果存储到单品统计表中。

这五个步骤，我们就可以用下面的代码来实现。

```
mysql> DELIMITER // -- 设置分割符为//
-> CREATE PROCEDURE demo.dailyoperation(transdate TEXT)
-> BEGIN -- 开始程序体
-> DECLARE startdate,enddate DATETIME; -- 定义变量
-> SET startdate = date_format(transdate,'%Y-%m-%d'); -- 给起始时间赋值
-> SET enddate = date_add(startdate,INTERVAL 1 DAY); -- 截止时间赋值为1天以后
-> -- 删除原有数据
-> DELETE FROM demo.dailystatistics
-> WHERE
-> salesdate = startdate;
-> -- 插入新计算的数据
-> INSERT into dailystatistics
-> (
-> salesdate,
-> itemnumber,
-> quantity,
-> actualvalue,
-> cost,
-> profit,
-> profitratio
-> )
-> SELECT
-> LEFT(b.transdate,10),
-> a.itemnumber,
-> SUM(a.quantity), -- 数量总计
-> SUM(a.salesvalue), -- 金额总计
-> SUM(a.quantity*c.avgimportprice), -- 计算成本
-> SUM(a.salesvalue-a.quantity*c.avgimportprice), -- 计算毛利
-> CASE sum(a.salesvalue) WHEN 0 THEN 0
-> ELSE round(sum(a.salesvalue-a.quantity*c.avgimportprice)/sum(a.salesvalue),
-> FROM
-> demo.transactiondetails AS a
-> JOIN
-> demo.transactionhead AS b
-> ON (a.transactionid = b.transactionid)
-> JOIN
-> demo.goodsmaster c
-> ON (a.itemnumber=c.itemnumber)
-> WHERE
-> b.transdate>startdate AND b.transdate<enddate
-> GROUP BY
-> LEFT(b.transdate,10),a.itemnumber
-> ORDER BY
-> LEFT(b.transdate,10),a.itemnumber;
-> END
-> // -- 语句结束，执行语句
Query OK, 0 rows affected (0.01 sec)
-> DELIMITER ; -- 恢复分隔符为;
```


这样，我们的存储过程就创建成功了。

在这个存储过程中，我们用到了存储过程的参数定义和程序体，这些具体是什么意思呢？我们来学习下。

存储过程的参数定义

存储过程可以有参数，也可以没有参数。一般来说，当我们通过客户端或者应用程序调用存储过程的时候，如果需要与存储过程进行数据交互，比如，存储过程需要根据输入的数值为基础进行某种数据处理和计算，或者需要把某个计算结果返回给调用它的客户端或者应用程序，就需要设置参数。否则，就不用设置参数。

参数有 3 种，分别是 IN、OUT 和 INOUT。

- IN 表示输入的参数，存储过程只是读取这个参数的值。如果没有定义参数种类，默认就是 IN，表示输入参数。
- OUT 表示输出的参数，存储过程在执行的过程中，把某个计算结果值赋给这个参数，执行完成之后，调用这个存储过程的客户端或者应用程序就可以读取这个参数返回的值了。
- INOUT 表示这个参数既可以作为输入参数，又可以作为输出参数使用。

除了定义参数种类，还要对参数的数据类型进行定义。在这个存储过程中，我定义了一个参数 transdate 的数据类型是 TEXT。这个参数的用处是告诉存储过程，我要处理的是哪一天的数据。我没有指定参数种类是 IN、OUT 或者 INOUT，这是因为在 MySQL 中，**如果不指定参数的种类，默认就是 IN，表示输入参数。**

知道了参数，下面我具体讲解一下这个存储过程的程序体。存储过程的具体操作步骤都包含在程序体里面，我们来分析一下程序体中 SQL 操作的内容，就可以知道存储过程到底在做什么。

存储过程的程序体

程序体中包含的是存储过程需要执行的 SQL 语句，一般通过关键字 BEGIN 表示 SQL 语句的开始，通过 END 表示 SQL 语句的结束。

在程序体的开始部分，我定义了 2 个变量，分别是 startdate 和 enddate。它们都是 DATETIME 类型，作用是根据输入参数 transdate，计算出需要筛选的数据的时间区间。

后面的代码分 3 步完成起始时间和截止时间的计算，并且分别赋值给变量 startdate 和 enddate。

第一步，使用 DATE_FROMAT（）函数，把输入的参数，按照 YYYY 年 MM 月 DD 日的格式转换成了日期时间类型数据，比如输入参数是"2020-12-01"，那么，转换成的日期时间值是"2020-12-01 00:00:00"，表示 2020 年 12 月 01 日 00 点 00 分 00 秒。

第二步，把第一步中计算出的值，作为起始时间赋值给变量 startdate。

第三步，把第一步中计算出的值，通过 DATE_ADD（）函数，计算出 1 天以后的时间赋值给变量 enddate。

这样，我就获得了需要计算的销售时段。计算出了起始时间和截止时间之后，我们先删除需要计算日期的单品统计数据，以防止数据重复。接着，我们重新计算单品的销售统计，并且把计算的结果插入到单品统计表。

在计算单品销售统计的时候，也分为 3 步：

1. 按照"成本 = 销售数量×平均价格"的方式计算成本；
2. 按照"毛利 = 销售金额 - 成本"的方式计算毛利；
3. 按照"毛利率 = 毛利 ÷销售金额"的方式计算毛利率。

需要注意的是，这里我使用 CASE 函数来解决销售金额为 0 时计算毛利的问题。这是为了防止计算出现被 0 除而报错的情况。不要以为销售金额就一定大于 0，在实际项目运行的过程中，会出现因为优惠而导致实际销售金额为 0 的情况。我建议你在实际工作中，把这些极端情况都考虑在内，提前进行防范，这样你的代码才能稳定可靠。

存储过程通过开始时定义的分隔符"//"结束，MySQL 执行这段 SQL 语句，就创建出了一个存储过程：demo.dailyoperation. 最后，你不要忘了把分隔符改回到";"。

创建完之后，怎么知道我们创建的存储过程是否成功了呢？下面我介绍一下查看存储过程的方法。

如何查看存储过程？

我们可以通过 SHOW CREATE PROCEDURE 存储过程名称，来查看刚刚创建的存储过程：

```
mysql> SHOW CREATE PROCEDURE demo.dailyoperation \G
*************************** 1. row ***************************
Procedure: dailyoperation       -- 存储过程名
sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE `dailyoperation`
BEGIN                             -- 开始程序体
DECLARE startdate,enddate DATETIME;
SET startdate = date_format(transdate,'%Y-%m-%d');
SET enddate = date_add(startdate,INTERVAL 1 DAY);
DELETE FROM demo.dailystatistics   -- 删除重复
WHERE
salesdate = startdate;
INSERT into dailystatistics        -- 将计算结果插入每日统计表
(
salesdate,
itemnumber,
quantity,
actualvalue,
cost,
profit,
profitratio
)
SELECT
LEFT(b.transdate,10),
a.itemnumber,
SUM(a.quantity),
SUM(a.salesvalue),
SUM(a.quantity*c.avgimportprice),
SUM(a.salesvalue-a.quantity*c.avgimportprice),
CASE sum(a.salesvalue) WHEN 0 THEN 0
ELSE round(sum(a.salesvalue-a.quantity*c.avgimportprice)/sum(a.salesvalue),4)
FROM
demo.transactiondetails AS a
JOIN
demo.transactionhead AS b
ON (a.transactionid = b.transactionid)
JOIN
demo.goodsmaster c
ON (a.itemnumber=c.itemnumber)
WHERE
b.transdate>startdate AND b.transdate<enddate
GROUP BY
LEFT(b.transdate,10),a.itemnumber
ORDER BY
LEFT(b.transdate,10),a.itemnumber;
END                              -- 结束程序体
character_set_client: gbk         -- 采用的字符集gbk
collation_connection: gbk_chinese_ci  -- 连接校对采用的字符集
Database Collation: utf8mb4_0900_ai_ci -- 数据校对字符集
1 row in set (0.00 sec)
```

如何调用存储过程？

下面我们来尝试调用一下这个存储过程，并且给它传递一个参数"2020-12-01"，也就是计算 2020 年 12 月 01 日的单品统计数据：

```
mysql> CALL demo.dailyoperation('2020-12-01');
Query OK, 2 rows affected (0.03 sec)
```

存储过程执行结果提示"Query OK"，表示执行成功了。"2 rows affected"表示执行的结果影响了 2 条数据记录。

我们用 SELECT 语句来查看一下单品统计表，看看有没有把单品统计的结果存入单品统计表中。

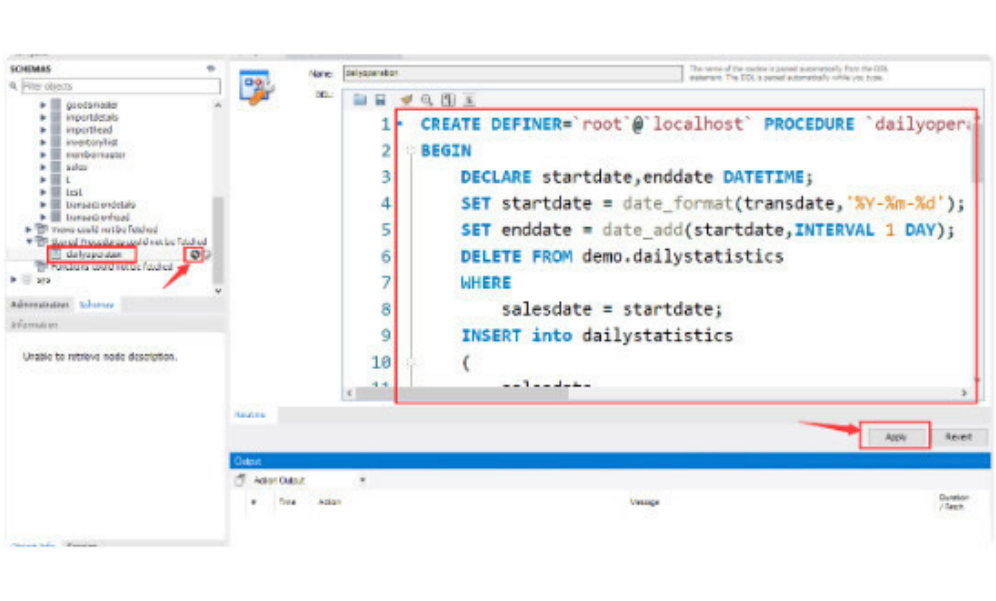
```
mysql> SELECT *                 -- 查询单品统计表中的数据
-> FROM demo.dailystatistics;
+----+------------+----------+-------------+--------+--------+-------------+----
| id | itemnumber | quantity | actualvalue | cost | profit | profitratio | sal
+----+------------+----------+-------------+--------+--------+-------------+----
| 13 | 1 | 3.000 | 267.00 | 100.50 | 166.50 | 0.6236 | 2020-12-01 00:00:00 |
| 14 | 2 | 2.000 | 10.00 | 7.00 | 3.00 | 0.3000 | 2020-12-01 00:00:00 |
+----+------------+----------+-------------+--------+--------+-------------+----
2 rows in set (0.00 sec)
```

看到了吗？我们已经能够在单品统计表中，查询到 2020 年 12 月 01 日的单品统计结果了。这也就意味着我们的存储过程被执行了，它计算出了我们需要的单品统计结果，并且把统计结果存入了单品统计表中。

如何修改和删除存储过程？

如果你需要修改存储过程的内容，我建议你在 Workbench 中操作。这是因为你可以在里面直接修改存储过程，而如果用 SQL 命令来修改存储过程，就必须删除存储过程再重新创建，相比之下，在 Workbench 中修改比较简单。

具体的做法是：在左边的导航栏，找到数据库 demo，展开之后，找到存储过程 stored procedure，然后找到我们刚刚创建的 dailyoperation，点击右边的设计按钮，就可以在右边的工作区进行修改了。



修改完成之后，点击工作区右下方的按钮"Apply"，保存修改。

在 MySQL 中，存储过程不像普通的编程语言（比如 VC++、Java 等）那样有专门的集成开发环境。因此，你可以通过 SELECT 语句，把程序执行的中间结果查询出来，来调试一个 SQL 语句的正确性。调试成功之后，把 SELECT 语句后移到下一个 SQL 语句之后，再调试下一个 SQL 语句。这样逐步推进，就可以完成对存储过程中所有操作的调试了。当然，你也可以把存储过程中的 SQL 语句复制出来，逐段单独调试。

删除存储过程很简单，你知道具体的语法就行了：

```
DROP PROCEDURE 存储过程名称;
```

总结

这节课，我们学习了创建、查看、修改和删除存储过程的具体方法。

存储过程的优点就是执行效率高，而且更加安全，不过，它也有着自身的缺点，那就是开发和调试的成本比较高，而且不太容易维护。

在存储过程开发的过程中，虽然也有一些第三方工具可以对存储过程进行调试，但要收费。我建议你通过 SELECT 语句输出变量值的办法进行调试，虽然有点麻烦，但是成本低，而且简单可靠。如果你的存储过程需要随产品一起分发，可以考虑把脚本放在安装程序中，在产品安装的过程中创建需要的存储过程。

思考题

请写一个简单的存储过程，要求是定义 2 个参数，一个输入参数 a，数据类型是 INT；另一个输出参数是 b，类型是 INT。程序体完成的操作是：b = a + 1。

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，欢迎你把它分享给你的朋友或同事，我们下节课见。

精选留言(7)

朱晓峰

置顶

你好，我是朱晓峰，下面我就来公布一下上节课思考题的答案：

上节课，我们学习了视图。下面是思考题的答案：

第一步，创建门票信息表（demo.tickets）和类别信息表（demo.ticketkind）

```
CREATE TABLE demo.tickets (
  id INT PRIMARY KEY AUTO_INCREMENT,
  tname TEXT,
  typeid INT,
  balance INT
);

CREATE TABLE demo.ticketkind (
  typeid INT PRIMARY KEY AUTO_INCREMENT,
  opentime TIME,
  closetime TIME
);
```

第二步，创建视图：

```
CREATE VIEW demo.vTickets AS
  SELECT
    a.tname, a.balance
  FROM
    demo.tickets AS a
      JOIN
    demo.ticketkind AS b ON (a.typeid = b.typeid)
  WHERE
    NOW() > b.opentime
      AND NOW() < b.closetime;
```

2021-05-17

未命名

#不负责任的评论#

互联网公司大都不爱用存储过程，在阿里巴巴发布的Java开发手册中，明确禁止了存储过程的使用。

"【强制】禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。"

主要原因大致为：不易测试，难以调试，版本控制困难……

不过对于商用数据库和部分传统软件业务，还是值得使用的。

2021-04-13 2

floating

课后思考题：

```
mysql> drop procedure if exists demo.proc_adder
  -> ;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> delimiter //
mysql> create procedure demo.proc_adder(in a int, out b int)
  -> begin
  -> set b = a + 1;
  -> end
  -> //
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
mysql> set @a = 5;
Query OK, 0 rows affected (0.00 sec)

mysql> call demo.proc_adder(@a, @b);
| 6 |
+------+
1 row in set (0.00 sec)
```

2021-04-19 1

Harry

本节课需要掌握关于存储过程的
创建/查看/调用/修改/删除/调试
其中调试需要重点关注

2021-04-14 1

菜鸟要先飞

```
delimiter //
create procedure aAddB (in a int,out b int)
begin
set b = a+1;
end
//
delimiter ;

set @b = 0;
call aAddB(1,@b);
begin
```

2021-05-12

Sinvi

```
DELIMITER //
CREATE PROCEDURE sum_two_num(a INT, INOUT b INT)
BEGIN
  select a + 1 into b;
END //
DELIMITER ;

call sum_two_num(10, @b);
select @b;
```

2021-04-20

lesserror

一个番茄钟，补上这一讲的内容。最近实在太忙了。

另外，需要指出这里老师的存储过程代码中的参数类型是TEXT，而不是文中所说的DATETIME类型。

像视图、存储过程、游标……这些概念从伴随MySQL很久了，虽然实际开发中很少用到。但是我们也必须了解，建立起自己完整的知识体系。

2021-04-15