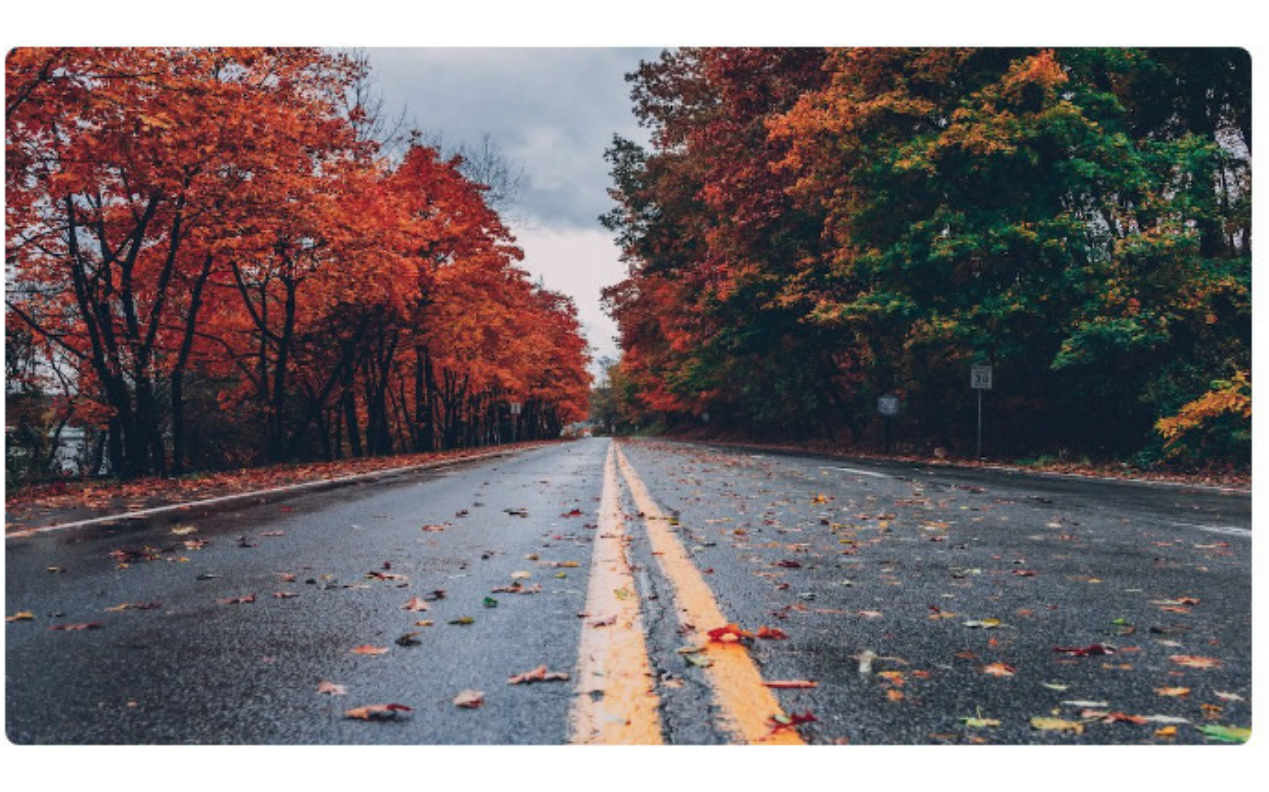
13 | 临时表：复杂查询，如何保存中间结果？

朱晓峰 2021-04-06



你好，我是朱晓峰。今天，我来和你聊一聊临时表。

当我们遇到一些复杂查询的时候，经常无法一步到位，或者是一步到位会导致查询语句太过复杂，开发和维护的成本过高。这个时候，就可以使用临时表。

下面，我就结合实际的项目来讲解一下，怎么拆解一个复杂的查询，通过临时表来保存中间结果，从而把一个复杂查询变得简单而且容易实现。

临时表是什么？

临时表是一种特殊的表，用来存储查询的中间结果，并且会随着当前连接的结束而自动删除。**MySQL 中有 2 种临时表，分别是内部临时表和外部临时表**：

- 内部临时表主要用于性能优化，由系统自动产生，我们无法看到；
- 外部临时表通过 SQL 语句创建，我们可以使用。

因为我们不能使用内部临时表，所以我就不多讲了。今天，我来重点讲一讲我们可以创建和使用的外部临时表。

首先，你要知道临时表的创建语法结构：

```
CREATE TEMPORARY TABLE 表名
(
字段名 字段类型,
...
);
```

跟普通表相比，临时表有 3 个不同的特征：

1. 临时表的创建语法需要用到关键字 TEMPORARY；
2. 临时表创建完成之后，只有当前连接可见，其他连接是看不到的，具有连接隔离性；
3. 临时表在当前连接结束之后，会被自动删除。

因为临时表有连接隔离性，不同连接创建相同名称的临时表也不会产生冲突，适合并发程序的运行。而且，连接结束之后，临时表会自动删除，也不用担心大量无用的中间数据会残留在数据库中。因此，我们就可以利用这些特点，用临时表来存储 SQL 查询的中间结果。

如何用临时表简化复杂查询？

刚刚提到，临时表可以简化复杂查询，具体是怎么实现的呢？我来介绍一下。

举个例子，超市经营者想要查询 2020 年 12 月的一些特定商品销售数量、进货数量、返厂数量，那么，我们就要先把销售、进货、返厂这 3 个模块分开计算，用临时表来存储中间计算的结果，最后合并在一起，形成超市经营者想要的结果集。

首先，我们统计一下在 2020 年 12 月的商品销售数据。

假设我们的销售流水表（mysales）如下所示：

transid (流水单号)	itemnumber (商品编号)	quantity (销售数量)	salesvalue (销售金额)	transdate (交易时间)
5897	1	2	176.22	2020-12-02
5897	2	5	24.75	2020-12-02
5898	1	3	234.96	2020-12-03

我们可以用下面的 SQL 语句，查询出每个单品的销售数量和销售金额，并存入临时表：

```
mysql> CREATE TEMPORARY TABLE demo.mysales
-> SELECT                              -- 用查询的结果直接生成临时表
-> itemnumber,
-> SUM(quantity) AS QUANTITY,
-> SUM(salesvalue) AS salesvalue
-> FROM
-> demo.transactiondetails
-> GROUP BY itemnumber
-> ORDER BY itemnumber;
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM demo.mysales;
+------------+----------+------------+
| itemnumber | QUANTITY | salesvalue |
+------------+----------+------------+
| 1 | 5.000 | 411.18 |
| 2 | 5.000 | 24.75 |
+------------+----------+------------+
2 rows in set (0.01 sec)
```

需要注意的是，这里我是直接用查询结果来创建的临时表。因为创建临时表就是为了存放某个查询的中间结果。直接用查询语句创建临时表比较快捷，而且连接结束后临时表就会被自动删除，不需要过多考虑表的结构设计问题（比如冗余、效率等）。

到这里，我们就有了一个存储单品销售统计的临时表。接下来，我们计算一下 2020 年 12 月的进货信息。

我们的进货数据包括进货单头表（importhead）和进货单明细表（importdetails）。

进货单头表包括进货单编号、供货商编号、仓库编号、操作员编号和验收日期：

listnumber (进货单编号)	supplierid (供货商编号)	stockid (仓库编号)	operatorid (操作员编号)	confitmationdate (验收日期)
4587	1	1	1	2020-12-02
4588	2	1	1	2020-12-03

进货单明细表包括进货单编号、商品编号、进货数量、进货价格和进货金额：

listnumber (进货单编号)	itemnumber (商品编号)	quantity (进货数量)	importprice (进货价格)	importvalue (进货金额)
4587	1	2	55	110
4587	2	5	3	15
4587	3	8	5	40
4588	1	3	60	180

我们用下面的 SQL 语句计算进货数据，并且保存在临时表里面：

```
mysql> CREATE TEMPORARY TABLE demo.myimport
-> SELECT b.itemnumber,SUM(b.quantity) AS quantity,SUM(b.importvalue) AS impor
-> FROM demo.importhead a JOIN demo.importdetails b
-> ON (a.listnumber=b.listnumber)
-> GROUP BY b.itemnumber;
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM demo.myimport;
+------------+----------+-------------+
| itemnumber | quantity | importvalue |
+------------+----------+-------------+
| 1 | 5.000 | 290.00 |
| 2 | 5.000 | 15.00 |
| 3 | 8.000 | 40.00 |
+------------+----------+-------------+
3 rows in set (0.00 sec)
```

这样，我们又得到了一个临时表 demo.myimport，里面保存了我们需要的进货数据。

接着，我们来查询单品返厂数据，并且保存到临时表。

我们的返厂数据表有 2 个，分别是返厂单头表（returnhead）和返厂单明细表（returndetails）。

返厂单头表包括返厂单编号、供货商编号、仓库编号、操作员编号和验收日期：

listnumber (返厂单编号)	supplierid (供货商编号)	stockid (仓库编号)	operatorid (操作员编号)	confirmationdate (验收日期)
654	1	1	1	2020-12-02
655	2	1	1	2020-12-03

返厂单明细表包括返厂单编号、商品编号、返厂数量、返厂价格和返厂金额：

listnumber (返厂单编号)	itemnumber (商品编号)	quantity (返厂数量)	importprice (返厂价格)	importvalue (返厂金额)
654	1	1	55	55
654	2	1	3	3
655	3	1	5	5
655	1	1	60	60

我们可以使用下面的 SQL 语句计算返厂信息，并且保存到临时表中。

```
mysql> CREATE TEMPORARY TABLE demo.myreturn
-> SELECT b.itemnumber,SUM(b.quantity) AS quantity,SUM(b.returnvalue) AS retur
-> FROM demo.returnhead a JOIN demo.returndetails b
-> ON (a.listnumber=b.listnumber)
-> GROUP BY b.itemnumber;
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM demo.myreturn;
+------------+----------+-------------+
| itemnumber | quantity | returnvalue |
+------------+----------+-------------+
| 1 | 2.000 | 115.00 |
| 2 | 1.000 | 3.00 |
| 3 | 1.000 | 5.00 |
+------------+----------+-------------+
3 rows in set (0.00 sec)
```

这样，我们就获得了单品的返厂信息。

有了前面计算出来的数据，现在，我们就可以把单品的销售信息、进货信息和返厂信息汇总到一起了。

如果你跟着实际操作的话，你可能会有这样一个问题：我们现在有 3 个临时表，分别存储单品的销售信息、进货信息和返厂信息。那么，能不能把这 3 个表相互关联起来，把这些信息都汇总到对应的单品呢？

答案是不行，不管是用内连接、还是用外连接，都不可以。因为无论是销售信息、进货信息，还是返厂信息，都存在商品信息缺失的情况。换句话说，就是在指定时间段内，某些商品可能没有销售，某些商品可能没有进货，某些商品可能没有返厂。如果仅仅通过这 3 个表之间的连接进行查询，我们可能会丢失某些数据。

为了解决这个问题，我们可以引入商品信息表。因为商品信息表包含所有的商品，因此，把商品信息表放在左边，与其他的表进行左连接，就可以确保所有的商品都包含在结果集中。凡是不存在的数值，都设置为 0，然后再筛选一下，把销售、进货、返厂都是 0 的商品去掉，这样就能得到我们最终希望的查询结果：2020 年 12 月的商品销售数量、进货数量和返厂数量。

代码如下所示：

```
mysql> SELECT
-> a.itemnumber,
-> a.goodsname,
-> ifnull(b.quantity,0) as salesquantity,    -- 如果没有销售记录，销售数量设置为0
-> ifnull(c.quantity,0) as importquantity,   -- 如果没有进货，进货数量设为0
-> ifnull(d.quantity,0) as returnquantity    -- 如果没有返厂，返厂数量设为0
-> FROM
-> demo.goodsmaster a               -- 商品信息表放在左边进行左连接，确保所有的商品都
-> LEFT JOIN demo.mysales b
-> ON (a.itemnumber=b.itemnumber)
-> LEFT JOIN demo.myimport c
-> ON (a.itemnumber=c.itemnumber)
-> LEFT JOIN demo.myreturn d
-> ON (a.itemnumber=d.itemnumber)
-> HAVING salesquantity>0 OR importquantity>0 OR returnquantity>0; -- 在结果集中
+------------+-----------+---------------+----------------+----------------+
| itemnumber | goodsname | salesquantity | importquantity | returnquantity |
+------------+-----------+---------------+----------------+----------------+
| 1 | 书 | 5.000 | 5.000 | 2.000 |
| 2 | 笔 | 5.000 | 5.000 | 1.000 |
| 3 | 橡皮 | 0.000 | 8.000 | 1.000 |
+------------+-----------+---------------+----------------+----------------+
3 rows in set (0.00 sec)
```

总之，通过临时表，我们就可以把一个复杂的问题拆分成很多个前后关联的步骤，把中间的运行结果存储起来，用于之后的查询。这样一来，就把面向集合的 SQL 查询变成了面向过程的编程模式，大大降低了难度。

内存临时表和磁盘临时表

由于采用的存储方式不同，临时表也可分为内存临时表和磁盘临时表，它们有着各自的优缺点，下面我来解释下。

关于内存临时表，有一点你要注意的是，你可以通过指定引擎类型（比如 ENGINE=MEMORY），来告诉 MySQL 临时表存储在内存中。

好了，现在我们先来创建一个内存中的临时表：

```
mysql> CREATE TEMPORARY TABLE demo.mytrans
-> (
-> itemnumber int,
-> groupnumber int,
-> branchnumber int
-> ) ENGINE = MEMORY;  (临时表数据存在内存中)
Query OK, 0 rows affected (0.00 sec)
```


接下来，我们在磁盘上创建一个同样结构的临时表。在磁盘上创建临时表时，只要我们不指定存储引擎，MySQL 会默认存储引擎是 InnoDB，并且把表存放在磁盘上。

```
mysql> CREATE TEMPORARY TABLE demo.mytransdisk
-> (
-> itemnumber int,
-> groupnumber int,
-> branchnumber int
-> );
Query OK, 0 rows affected (0.00 sec)
```

现在，我们向刚刚的两张表里都插入同样数量的记录，然后再分别做一个查询：

```
mysql> SELECT COUNT(*) FROM demo.mytrans;
+----------+
| count(*) |
+----------+
| 4355 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM demo.mytransdisk;
+----------+
| count(*) |
+----------+
| 4355 |
+----------+
1 row in set (0.21 sec)
```

可以看到，区别是比较明显的。对于同一条查询，内存中的临时表执行时间不到 10 毫秒，而磁盘上的表却用掉了 210 毫秒。显然，内存中的临时表查询速度更快。

不过，内存中的临时表也有缺陷。因为数据完全在内存中，所以，一旦断电，数据就消失了，无法找回。不过临时表只保存中间结果，所以还是可以用的。

我画了一张图，汇总了内存临时表和磁盘临时表的优缺点：

类别	优点	缺点
内存临时表	查询速度快	一旦断电，全部丢失，数据无法找回
磁盘临时表	数据不易丢失	速度相对较慢

总结

这节课，我们学习了临时表的概念，以及使用临时表来存储中间结果以拆分复杂查询的方法。临时表可以存储在磁盘中，也可以通过指定引擎的办法存储在内存中，以加快存取速度。

其实，临时表有很多好处，除了可以帮助我们把复杂的 SQL 查询拆分成多个简单的 SQL 查询，而且，因为临时表是连接隔离的，不同的连接可以使用相同的临时表名称，相互之间不会受到影响。除此之外，临时表会在连接结束的时候自动删除，不会占用磁盘空间。

当然，临时表也有不足，比如会挤占空间。我建议你，**在使用临时表的时候，要从简化查询和挤占资源两个方面综合考虑，既不能过度加重系统的负担，同时又能够通过存储中间结果，最大限度地简化查询。**

思考题

我们有这样的一个销售流水表：

branchnumber (门店编号)	cashiernumber (收款机编号)	itemnumber (商品编号)	salesvalue (销售金额)
1	1	1	10
1	2	3	20

假设有多个门店，每个门店有多台收款机，每台收款机销售多种商品，请问如何查询每个门店、每台收款机的销售金额占所属门店的销售金额的比率呢？

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(9)

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，我们学习了事务。下面是思考题的答案：

这种说法是不对的，事务会确保事务处理中的操作要么全部执行，要么全部不执行，执行中遇到错误，是继续还是回滚，则需要程序员来处理。

2021-04-21

未命名

据了解，临时表的开销很大，不建议在高访问量的线上系统中使用。离线备份库或供数据分析所用的数据库上可以考虑有限制的使用。
毁，会不会影响后面再次从连接池取出该连接使用的情况？

2021-04-18 1

星空下

数据库一般是性能瓶颈点，用临时表太占用数据库资源吧

2021-04-06 1

Harry

当引擎类型为 memory 时，如果去掉 temporary 那么表还会存储在内存中吗？

2021-04-06 1 1

Harry

最后左连接的代码中，为什么要使用 having 而不使用 where 呢？

2021-04-06 1 1

菜鸟要先飞

创建门店销售额临时表
create temporary table branchsales
select branchnumber,sum(salesvalue) as salesvalue from transdetails group by branchnumber;

创建每个门店各自收款机销售额的临时表
create temporary table cashiernumber
select branchnumber,cashiernumber, salesvalue as salesvalue from transdetails group by branchnumber,cashiernumber;

计算占比：
select a.branchnumber,b.cashiernumber, b.salesvalue/a.salesvalue as '占比' from branchsales as a join cashiernumber as b on(a.branchnumber = b.branchnumber);

2021-05-11

---

lesserror

之前对临时表的认识停留在对系统性能开销大的理解上。

开发中也没有尝试过使用临时表。看了这节内容，对临时表如何存储中间结果来简化查询有了一定的认识。

之后开发中如果有适合临时表的场景，会来再来看看这篇的内容。并分享一下使用心得。

2021-04-06

洛奇

临时表的数据是不是易丢失，这不重要吧？断电后，连接也断了，这时候有去找回临时表的数据的必要吗？

2021-04-06