09-怎么能避免写出慢SQL？

你好，我是李玥。

通过上节课的案例，我们知道，一个慢SQL就可以直接让MySQL瘫痪。今天这节课，我们一起看一下，怎么才能避免写出危害数据库的慢SQL。

所谓慢SQL，就是执行特别慢的SQL语句。什么样的SQL语句是慢SQL？多慢才算是慢SQL？并没有一个非常明确的标准或者说是界限。但并不是说，我们就很难区分正常的SQL和慢SQL，在大多数实际的系统中，慢SQL消耗掉的数据库资源，往往是正常SQL的几倍、几十倍甚至几百倍，所以还是非常容易区分的。

但问题是，我们不能等着系统上线，慢SQL吃光数据库资源之后，再找出慢SQL来改进，那样就晚了。那么，怎样才能在开发阶段尽量避免写出慢SQL呢？

定量认识MySQL

我们回顾一下上节课的案例，那个系统第一次全站宕机发生在圣诞节平安夜，故障之前的一段时间，系统并没有更新过版本，这个时候，其实慢SQL已经存在了，直到平安夜那天，访问量的峰值比平时增加一些，正是增加的这部分访问量，引发了数据库的雪崩。

这说明，**慢SQL对数据库的影响，是一个量变到质变的过程，对“量”的把握，就很重要**。作为一个合格的程序员，你需要对数据库的能力，有一个定量的认识。

影响MySQL处理能力的因素很多，比如：服务器的配置、数据库中的数据量大小、MySQL的一些参数配置、数据库的繁忙程度等等。但是，通常情况下，这些因素对于MySQL性能和处理能力影响范围，大概在几倍的性能差距。所以，我们不需要精确的性能数据，只要掌握一个大致的量级，就足够指导我们的开发工作了。

一台MySQL数据库，大致处理能力的极限是，每秒一万条左右的简单SQL，这里的“简单SQL”，指的是类似于主键查询这种不需要遍历很多条记录的SQL。根据服务器的配置高低，可能低端的服务器只能达到每秒几千条，高端的服务器可以达到每秒钟几万条，所以这里给出的一万TPS是中位数的经验值。考虑到正常的系统不可能只有简单SQL，所以实际的TPS还要打很多折扣。

我的经验数据，一般一台MySQL服务器，平均每秒钟执行的SQL数量在几百左右，就已经是非常繁忙了，即使看起来CPU利用率和磁盘繁忙程度没那么高，你也需要考虑给数据库“减负”了。

另外一个重要的定量指标是，到底多慢的SQL才算慢SQL。这里面这个“慢”，衡量的单位本来是执行时长，但是时长这个东西，我们在编写SQL的时候并不好去衡量。那我们可以用执行SQL查询时，需要遍历的数据行数替代时间作为衡量标准，因为查询的执行时长基本上是和遍历的数据行数正相关的。

你在编写一条查询语句的时候，可以依据你要查询数据表的数据总量，估算一下这条查询大致需要遍历多少行数据。如果遍历行数在百万以内的，只要不是每秒钟都要执行几十上百次的频繁查询，可以认为是安全的。遍历数据行数在几百万的，查询时间最少也要几秒钟，你就要仔细考虑有没有优化的办法。遍历行数达到千万量级和以上的，我只能告诉你，这种查询就不应该出现在你的系统中。当然我们这里说的都是在线交易系统，离线分析类系统另说。

遍历行数在千万左右，是MySQL查询的一个坎儿。MySQL中单个表数据量，也要尽量控制在一千万条以

下，最多不要超过二三千万这个量级。原因也很好理解，对一个千万级别的表执行查询，加上几个WHERE条件过滤一下，符合条件的数据最多可能在几十万或者百万量级，这还可以接受。但如果再和其他的表做一个联合查询，遍历的数据量很可能就超过千万级别了。所以，每个表的数据量最好小于千万级别。

如果数据库中的数据量就是很多，而且查询业务逻辑就需要遍历大量数据怎么办？

使用索引避免全表扫描

使用索引可以有效地减少执行查询时遍历数据的行数，提高查询性能。

数据库索引的原理也很简单，我举个例子你就明白了。比如说，有一个无序的数组，数组的每个元素都是一个用户对象。如果说我们要把所有姓李的用户找出来。比较笨的办法是，用一个循环把数组遍历一遍。

有没有更好的办法？很多办法是吧？比如说，我们用一个Map(在有些编程语言中是Dictionary)来给数组做一个索引，Key保存姓氏，值是所有这个姓氏的用户对象在数组中序号的集合。这样再查找的时候，就不用去遍历数组，先在Map中查找，然后再直接用序号去数组中拿用户数据，这样查找速度就快多了。

这个例子对应到数据库中，存放用户数据的数组就是表，我们构建的Map就是索引。实际上数据库的索引，和编程语言中的Map或者Dictionary，它们的数据结构都是差不多的，基本上就是各种B树和HASH表。

绝大多数情况下，我们编写的查询语句，都应该使用索引，避免去遍历整张表，也就是通常说的，避免全表扫描。你在每次开发新功能，需要给数据库增加一个新的查询时，都要评估一下，是不是有索引可以支撑新的查询语句，如果有必要的话，需要新建索引来支持新增的查询。

但是，增加索引付出的代价是，会降低数据插入、删除和更新的性能。这个也很好理解，增加了索引，在数据变化的时候，不仅要变更数据表里的数据，还要去变更每个索引。所以，对于更新频繁并且对更新性能要求较高的表，可以尽量少建索引。而对于查询较多更新较少的表，可以根据查询的业务逻辑，适当多建一些索引。

怎么写SQL能更好地使用索引，查询效率更高，这是一门手艺，需要丰富的经验，不是通过一节课的学习能练成的。但是，我们是有方法，可以评估写出来的SQL的查询性能怎么样，是不是一个潜在的“慢SQL”。

逻辑不是很复杂的单表查询，我们可能还可以分析出来，查询会使用哪个索引。但如果是比较复杂的多表联合查询，我们单看SQL语句本身，就很难分析出查询到底会命中哪些索引，会遍历多少行数据。MySQL和大部分数据库，都提供一个帮助我们分析查询功能：执行计划。

分析SQL执行计划

在MySQL中使用执行计划也非常简单，只要在你的SQL语句前面加上**EXPLAIN**关键字，然后执行这个查询语句就可以了。

举个例子说明，比如有一个用户表，包含用户ID、姓名、部门编号和状态这几个字段：

```
mysql> desc user;
+-----------------+----------------------+------+-----+---------+----------------+
| Field           | Type                 | Null | Key | Default | Extra          |
+-----------------+----------------------+------+-----+---------+----------------+
| id              | bigint(19) unsigned  | NO   | PRI | NULL    | auto_increment |
| name            | varchar(50)          | NO   |     | NULL    |                |
| department_code | varchar(50)          | NO   | MUL | NULL    |                |
| status          | tinyint(4)           | NO   |     | NULL    |                |
+-----------------+----------------------+------+-----+---------+----------------+
```

我们希望查询某个二级部门下的所有人，查询条件就是，部门代号以00028开头的所有人。下面这两个SQL，他们的查询结果是一样的，都满足要求，但是，哪个查询性能更好呢？

```
SELECT * FROM user WHERE left(department_code, 5) = '00028';
SELECT * FROM user WHERE department_code LIKE '00028%';
```

我们分别查看一下这两个SQL的执行计划：

```
mysql> EXPLAIN SELECT * FROM user WHERE left(department_code, 5) = '00028';
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | user  | ALL  | NULL          | NULL | NULL    | NULL | 4534 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM user WHERE department_code LIKE '00028%';
+----+-------------+-------+-------+--------------------------+--------------------------+---------+------+------+-------------+
| id | select_type | table | type  | possible_keys            | key                      | key_len | ref  | rows | Extra       |
+----+-------------+-------+-------+--------------------------+--------------------------+---------+------+------+-------------+
|  1 | SIMPLE      | user  | range | idx_user_department_code | idx_user_department_code | 152     | NULL |    8 | Using where |
+----+-------------+-------+-------+--------------------------+--------------------------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

我带你一起来分析一下这两个SQL的执行计划。首先来看rows这一列，rows的含义就是，MySQL预估执行这个SQL可能会遍历的数据行数。第一个SQL遍历了四千多行，这就是整个User表的数据条数；第二个SQL只有8行，这8行其实就是符合条件的8条记录。显然第二个SQL查询性能要远远好于第一个SQL。

为什么第一个SQL需要全表扫描，第二个SQL只遍历了很少的行数呢？注意看type这一列，这一列表示这个查询的访问类型。ALL代表全表扫描，这是最差的情况。range代表使用了索引，在索引中进行范围查找，因为SQL语句的WHERE中有一个LIKE的查询条件。如果直接命中索引，type这一列显示的是index。如果使用了索引，可以在key这一列中看到，实际上使用了哪个索引。

通过对比这两个SQL的执行计划，就可以看出来，第二个SQL虽然使用了普遍认为低效的LIKE查询条件，但是仍然可以用到索引的范围查找，遍历数据的行数远远少于第一个SQL，查询性能更好。

小结

在开发阶段，衡量一个SQL查询语句查询性能的手段是，估计执行SQL时需要遍历的数据行数。遍历行数在百万以内，可以认为是安全的SQL，百万到千万这个量级则需要仔细评估和优化，千万级别以上则是非常危险的。为了减少慢SQL的可能性，每个数据表的行数最好控制在千万以内。

索引可以显著减少查询遍历数据的数量，所以提升SQL查询性能最有效的方式就是，让查询尽可能多的命中索引，但索引也是一把双刃剑，它在提升查询性能的同时，也会降低数据更新的性能。

对于复杂的查询，最好使用SQL执行计划，事先对查询做一个分析。在SQL执行计划的结果中，可以看到查询预估的遍历行数，命中了哪些索引。执行计划也可以很好地帮助你优化你的查询语句。

思考题

课后请你想一下，在讲解SQL执行计划那个例子中的第一个SQL，为什么没有使用索引呢？

```
SELECT * FROM user WHERE left(department_code, 5) = '00028';
```

欢迎你在留言区与我讨论，如果你觉得今天学到的知识对你有帮助，也欢迎把它分享给你的朋友。

精选留言：

- 冯玉鹏 2020-03-17 01:26:26
  innodb 的索引是用索引关联列以b+树的形式 管理，其中主键索性和数据的物理顺序一致，也叫聚集索引。非主键索引实际上是指向主键索引。
  文末的问题对 department_code 列 left 运算后，MySQL 认为运算后的结果不可与原数据列内容匹配，故采用全表扫描， 而第二个语句like '00028%' 可以使用到索引 是因为索引的最左匹配选择，如果%在前面也将无法使用索引。PS:在这里MySQL的查询优化器在使用了left函数无法匹配索引可以认为有偷懒的嫌疑，哈哈~ 类似的场景还有 where 列 +1 = val 查询优化器也完全可以改写成 where 列=val - 1。 [2赞]

  作者回复2020-03-17 10:08:22
  👍👍👍

- 肥low 2020-03-17 09:35:54
  因为第一个用到了函数 如果在建立索引的时候就left 在SQL中不显式的left就能用到了

- 攻城拔寨 2020-03-17 09:32:45
  索引使用函数会让索引失效，因为必须拿所有索引去计算才能得到结果

- 刘楠 2020-03-17 08:55:46
  LEFT()函数是一个字符串函数，它返回具有指定长度的字符串的左边部分。

  LEFT(Str,length);

  接收两个参数：

  str：一个字符串；

  length：想要截取的长度，是一个正整数；
  left函数，会扫描所有的行，并试图找到所有能与匹配的记录，是值班表扫描

- 夜空中最亮的星（华仔） 2020-03-17 08:52:08
  运算不能使用索引

- 0x12FD16B 2020-03-17 08:47:07
  MySQL 执行器去执行查询语句时会先去查询所有的 user 信息，然后对 department_code列执行函数，再和 00028 比较。

- 滴流乱转小胖子 2020-03-17 08:29:52
  使用left(department_code, 5)函数对department_code列所有值，进行了操作，只能进行全表扫描。 对不对老师?

- 撒旦的堕落 2020-03-17 08:26:23
  应该是where条件后使用了函数导致无法使用索引

- 饭团 2020-03-17 08:25:55
  因为where条件左值使用了函数！

- webmin 2020-03-17 08:16:44
  left(department_code, 5)经过了函数计算，与索引建立时条件不一至，如果要让left(department_code, 5)使用索引，可以使用函数索引机制来处理。

- myrfy 2020-03-17 07:57:06
  left是一个函数，而索引存的是原始数据，必须要通过把原始数据依次交给函数去执行才能拿到函数的值，所以需要查所有数据

- 每天晒白牙 2020-03-17 07:14:42
  索引进行运算会失效

- 火车日记 2020-03-17 00:21:07
  where中列使用了函数，优化器无法用到索引