

## 14 | count(\*)这么慢，我该怎么办？

2018-12-14 林晓斌

MySQL实战45讲

[进入课程 >](#)



讲述：林晓斌

时长 15:20 大小 14.05M



在开发系统的时候，你可能经常需要计算一个表的行数，比如一个交易系统的所有变更记录总数。这时候你可能会想，一条 `select count(*) from t` 语句不就解决了吗？

但是，你会发现随着系统中记录数越来越多，这条语句执行得也会越来越慢。然后你可能就想了，MySQL 怎么这么笨啊，记个总数，每次要查的时候直接读出来，不就好了吗。

那么今天，我们就来聊聊 `count(*)` 语句到底是怎样实现的，以及 MySQL 为什么会这么实现。然后，我会再和你说说，如果应用中有这种频繁变更并需要统计表行数的需求，业务设计上可以怎么做。

### count(\*) 的实现方式

你首先要明确的是，在不同的 MySQL 引擎中，count(\*) 有不同的实现方式。

MyISAM 引擎把一个表的总行数存在了磁盘上，因此执行 count(\*) 的时候会直接返回这个数，效率很高；

而 InnoDB 引擎就麻烦了，它执行 count(\*) 的时候，需要把数据一行一行地从引擎里面读出来，然后累积计数。

这里需要注意的是，我们在这篇文章里讨论的是没有过滤条件的 count(\*)，如果加了 where 条件的话，MyISAM 表也是不能返回得这么快的。

在前面的文章中，我们一起分析了为什么要使用 InnoDB，因为不论是在事务支持、并发能力还是在数据安全方面，InnoDB 都优于 MyISAM。我猜你的表也一定是用了 InnoDB 引擎。这就是当你的记录数越来越多的时候，计算一个表的总行数会越来越慢的原因。

**那为什么 InnoDB 不跟 MyISAM 一样，也把数字存起来呢？**

这是因为即使是在同一个时刻的多个查询，由于多版本并发控制（MVCC）的原因，InnoDB 表“应该返回多少行”也是不确定的。这里，我用一个算 count(\*) 的例子来为你解释一下。

假设表 t 中现在有 10000 条记录，我们设计了三个用户并行的会话。

会话 A 先启动事务并查询一次表的总行数；

会话 B 启动事务，插入一行后记录后，查询表的总行数；

会话 C 先启动一个单独的语句，插入一行记录后，查询表的总行数。

我们假设从上到下是按照时间顺序执行的，同一行语句是在同一时刻执行的。

会话A	会话B	会话C
begin;		
select count(*) from t;		
		insert into t (插入一行);
	begin;	
	insert into t (插入一行);	
select count(*) from t; (返回10000)	select count(*) from t; (返回10002)	select count(*) from t; (返回10001)

图 1 会话 A、B、C 的执行流程

你会看到，在最后一个时刻，三个会话 A、B、C 会同时查询表 t 的总行数，但拿到的结果却不同。

这和 InnoDB 的事务设计有关系，可重复读是它默认的隔离级别，在代码上就是通过多版本并发控制，也就是 MVCC 来实现的。每一行记录都要判断自己是否对这个会话可见，因此对于 count(\*) 请求来说，InnoDB 只好把数据一行一行地读出依次判断，可见的行才能够用于计算“基于这个查询”的表的总行数。

备注：如果你对 MVCC 记忆模糊了，可以再回顾下第 3 篇文章[《事务隔离：为什么你改了我还看不见？》](#)和第 8 篇文章[《事务到底是隔离的还是不隔离的？》](#)中的相关内容。

当然，现在这个看上去笨笨的 MySQL，在执行 count(\*) 操作的时候还是做了优化的。

你知道的，InnoDB 是索引组织表，主键索引树的叶子节点是数据，而普通索引树的叶子节点是主键值。所以，普通索引树比主键索引树小很多。对于 count(\*) 这样的操作，遍历哪个索引树得到的结果逻辑上都是一样的。因此，MySQL 优化器会找到最小的那棵树来遍历。**在保证逻辑正确的前提下，尽量减少扫描的数据量，是数据库系统设计的通用法则之一。**

如果你用过 show table status 命令的话，就会发现这个命令的输出结果里面也有一个 TABLE\_ROWS 用于显示这个表当前有多少行，这个命令执行挺快的，那这个 TABLE\_ROWS 能代替 count(\*) 吗？

你可能还记得在第 10 篇文章 [《MySQL 为什么有时候会选错索引？》](#) 中我提到过，索引统计的值是通过采样来估算的。实际上，TABLE\_ROWS 就是从这个采样估算得来的，因此它也很不准。有多不准呢，官方文档说误差可能达到 40% 到 50%。**所以，show table status 命令显示的行数也不能直接使用。**

到这里我们小结一下：

MyISAM 表虽然 count(\*) 很快，但是不支持事务；

show table status 命令虽然返回很快，但是不准确；

InnoDB 表直接 count(\*) 会遍历全表，虽然结果准确，但会导致性能问题。

那么，回到文章开头的问题，如果你现在有一个页面经常要显示交易系统的操作记录总数，到底应该怎么办呢？答案是，我们只能自己计数。

接下来，我们讨论一下，看看自己计数有哪些方法，以及每种方法的优缺点有哪些。

这里，我先和你说一下这些方法的基本思路：你需要自己找一个地方，把操作记录表的行数存起来。

## 用缓存系统保存计数

对于更新很频繁的库来说，你可能会第一时间想到，用缓存系统来支持。

你可以用一个 Redis 服务来保存这个表的总行数。这个表每被插入一行 Redis 计数就加 1，每被删除一行 Redis 计数就减 1。这种方式下，读和更新操作都很快，但你再想一下这种方式存在什么问题吗？

没错，缓存系统可能会丢失更新。

Redis 的数据不能永久地留在内存里，所以你会找一个地方把这个值定期地持久化存储起来。但即使这样，仍然可能丢失更新。试想如果刚刚在数据表中插入了一行，Redis 中保存的值也加了 1，然后 Redis 异常重启了，重启后你要从存储 redis 数据的地方把这个值读回来，而刚刚加 1 的这个计数操作却丢失了。

当然了，这还是有解的。比如，Redis 异常重启以后，到数据库里面单独执行一次 count(\*) 获取真实的行数，再把这个值写回到 Redis 里就可以了。异常重启毕竟不是经常出现的情况，这一次全表扫描的成本，还是可以接受的。

但实际上，**将计数保存在缓存系统中的方式，还不只是丢失更新的问题。即使 Redis 正常工作，这个值还是逻辑上不精确的。**

你可以设想一下有这么一个页面，要显示操作记录的总数，同时还要显示最近操作的 100 条记录。那么，这个页面的逻辑就需要先到 Redis 里面取出计数，再到数据表里面取数据记录。

我们是这么定义不精确的：

- 1. 一种是，查到的 100 行结果里面有最新插入记录，而 Redis 的计数里还没加 1；
- 2. 另一种是，查到的 100 行结果里没有最新插入的记录，而 Redis 的计数里已经加了 1。

这两种情况，都是逻辑不一致的。

我们一起来看看这个时序图。

时刻	会话A	会话B
T1		
T2	插入一行数据R;	
T3		读Redis计数; 查询最近100条记录;
T4	Redis 计数加1;	

图 2 会话 A、B 执行时序图

图 2 中，会话 A 是一个插入交易记录的逻辑，往数据表里插入一行 R，然后 Redis 计数加 1；会话 B 就是查询页面显示时需要的数据。

在图 2 的这个时序里，在 T3 时刻会话 B 来查询的时候，会显示出新插入的 R 这个记录，但是 Redis 的计数还没加 1。这时候，就会出现我们说的数据不一致。

你一定会说，这是因为我们执行新增记录逻辑时候，是先写数据表，再改 Redis 计数。而读的时候是先读 Redis，再读数据表，这个顺序是相反的。那么，如果保持顺序一样的话，是不是就没问题了？我们现在把会话 A 的更新顺序换一下，再看看执行结果。

时刻	会话A	会话B
T1		
T2	Redis 计数加1;	
T3		读Redis计数; 查询最近100条记录;
T4	插入一行数据R;	
T5		

图 3 调整顺序后，会话 A、B 的执行时序图

你会发现，这时候反过来了，会话 B 在 T3 时刻查询的时候，Redis 计数加了 1 了，但还查不到新插入的 R 这一行，也是数据不一致的情况。

在并发系统里面，我们是无法精确控制不同线程的执行时刻的，因为存在图中的这种操作序列，所以，我们说即使 Redis 正常工作，这个计数值还是逻辑上不精确的。

### 在数据库保存计数

根据上面的分析，用缓存系统保存计数有丢失数据和计数不精确的问题。那么，**如果我们把这个计数直接放到数据库里单独的一张计数表 C 中，又会怎么样呢？**

首先，这解决了崩溃丢失的问题，InnoDB 是支持崩溃恢复不丢数据的。

备注：关于 InnoDB 的崩溃恢复，你可以再回顾一下第 2 篇文章[《日志系统：一条 SQL 更新语句是如何执行的？》](#)中的相关内容。



然后，我们再看看能不能解决计数不精确的问题。

你会说，这不一样吗？无非就是把图 3 中对 Redis 的操作，改成了对计数表 C 的操作。只要出现图 3 的这种执行序列，这个问题还是无解的吧？

这个问题还真不是无解的。

我们这篇文章要解决的问题，都是由于 InnoDB 要支持事务，从而导致 InnoDB 表不能把 count(\*) 直接存起来，然后查询的时候直接返回形成的。

所谓以子之矛攻子之盾，现在我们就利用“事务”这个特性，把问题解决掉。

时刻	会话A	会话B
T1		
T2	begin; 表C中计数值加1;	
T3		begin; 读表C计数值; 查询最近100条记录; commit;
T4	插入一行数据R commit;	

图 4 会话 A、B 的执行时序图

我们来看下现在的执行结果。虽然会话 B 的读操作仍然是在 T3 执行的，但是因为这时候更新事务还没有提交，所以计数值加 1 这个操作对会话 B 还不可见。

因此，会话 B 看到的结果里，查计数值和“最近 100 条记录”看到的结果，逻辑上就是一致的。

## 不同的 count 用法

在前面文章的评论区，有同学留言问到：在 `select count(?) from t` 这样的查询语句里面，`count(*)`、`count(主键 id)`、`count(字段)` 和 `count(1)` 等不同用法的性能，有哪些差别。今天谈到了 `count(*)` 的性能问题，我就借此机会和你详细说明一下这几种用法的性能差别。

需要注意的是，下面的讨论还是基于 InnoDB 引擎的。

这里，首先你要弄清楚 `count()` 的语义。`count()` 是一个聚合函数，对于返回的结果集，一行行地判断，如果 `count` 函数的参数不是 `NULL`，累计值就加 1，否则不加。最后返回累计值。

所以，`count(*)`、`count(主键 id)` 和 `count(1)` 都表示返回满足条件的结果集的总行数；而 `count(字段)`，则表示返回满足条件的数据行里面，参数“字段”不为 `NULL` 的总个数。

至于分析性能差别的时候，你可以记住这么几个原则：

1. server 层要什么就给什么；
2. InnoDB 只给必要的值；
3. 现在的优化器只优化了 `count(*)` 的语义为“取行数”，其他“显而易见”的优化并没有做。

这是什么意思呢？接下来，我们就一个个地来看看。

**对于 `count(主键 id)` 来说**，InnoDB 引擎会遍历整张表，把每一行的 `id` 值都取出来，返回给 server 层。server 层拿到 `id` 后，判断是不可能为空的，就按行累加。

**对于 `count(1)` 来说**，InnoDB 引擎遍历整张表，但不取值。server 层对于返回的每一行，放一个数字“1”进去，判断是不可能为空的，按行累加。

单看这两个用法的差别的话，你能对比出来，`count(1)` 执行得要比 `count(主键 id)` 快。因为从引擎返回 `id` 会涉及到解析数据行，以及拷贝字段值的操作。

**对于 `count(字段)` 来说：**

1. 如果这个“字段”是定义为 `not null` 的话，一行行地从记录里面读出这个字段，判断不能为 `null`，按行累加；



2. 如果这个“字段”定义允许为 null，那么执行的时候，判断到有可能是 null，还要把值取出来再判断一下，不是 null 才累加。

也就是前面的第一条原则，server 层要什么字段，InnoDB 就返回什么字段。

**但是 count(\*) 是例外**，并不会把全部字段取出来，而是专门做了优化，不取值。count(\*) 肯定不是 null，按行累加。

看到这里，你一定会说，优化器就不能自己判断一下吗，主键 id 肯定非空啊，为什么不能按照 count(\*) 来处理，多么简单的优化啊。

当然，MySQL 专门针对这个语句进行优化，也不是不可以。但是这种需要专门优化的情况太多了，而且 MySQL 已经优化过 count(\*) 了，你直接使用这种用法就可以了。

所以结论是：按照效率排序的话，count(字段)<count(主键 id)<count(1)≈count(\*)，所以我建议你，尽量使用 count(\*)。

## 小结

今天，我和你聊了聊 MySQL 中获得表行数的两种方法。我们提到了在不同引擎中 count(\*) 的实现方式是不一样的，也分析了用缓存系统来存储计数值存在的问题。

其实，把计数放在 Redis 里面，不能够保证计数和 MySQL 表里的数据精确一致的原因，**是这两个不同的存储构成的系统，不支持分布式事务，无法拿到精确一致的视图**。而把计数值也放在 MySQL 中，就解决了一致性视图的问题。

InnoDB 引擎支持事务，我们利用好事务的原子性和隔离性，就可以简化在业务开发时的逻辑。这也是 InnoDB 引擎备受青睐的原因之一。

最后，又到了今天的思考题时间了。

在刚刚讨论的方案中，我们用了事务来确保计数准确。由于事务可以保证中间结果不被别的事务读到，因此修改计数值和插入新记录的顺序是不影响逻辑结果的。但是，从并发系统性能的角度考虑，你觉得在这个事务序列里，应该先插入操作记录，还是应该先更新计数表呢？

你可以把你的思考和观点写在留言区里，我会在下一篇文章的末尾给出我的参考答案。感谢你的收听，也欢迎你把这篇文章分享给更多的朋友一起阅读。

## 上期问题时间

上期我给你留的问题是，什么时候使用 `alter table t engine=InnoDB` 会让一个表占用的空间反而变大。

在这篇文章的评论区里面，大家都提到了一个点，就是这个表，本身就已经没有空洞的了，比如说刚刚做过一次重建表操作。

在 DDL 期间，如果刚好有外部的 DML 在执行，这期间可能会引入一些新的空洞。

@飞翔 提到了一个更深刻的机制，是我们在文章中说没说的。在重建表的时候，InnoDB 不会把整张表占满，每个页留了 1/16 给后续的更新用。也就是说，其实重建表之后不是“最”紧凑的。

假如是这么一个过程：

1. 将表 t 重建一次；
2. 插入一部分数据，但是插入的这些数据，用掉了一部分的预留空间；
3. 这种情况下，再重建一次表 t，就可能会出现这个问题中的现象。

评论区留言点赞板：

@W\_T 等同学提到了数据表本身紧凑的情况；

@undifined 提了一个好问题，@帆帆帆帆帆帆帆 同学回答了这个问题；

@陈飞 @郜 @wang chen wen 都提了很不错的问题，大家可以去看看。

# MySQL 实战 45 讲

从原理到实战，丁奇带你搞懂 MySQL

林晓斌

网名丁奇  
前阿里资深技术专家



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 13 | 为什么表数据删掉一半，表文件大小不变？

下一篇 15 | 答疑文章（一）：日志和索引相关问题

## 精选留言 (102)

写留言



阿建 置顶

2018-12-14

92

从并发系统性能的角度考虑，应该先插入操作记录，再更新计数表。

知识点在《行锁功过：怎么减少行锁对性能的影响？》

因为更新计数表涉及到行锁的竞争，先插入再更新能最大程度地减少了事务之间的锁等待，提升了并发度。

展开 ▾

作者回复: 好几个同学说对，你第一个标明出处👍





倪大人 置顶

2018-12-15

4

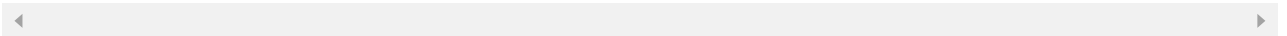
看到有同学说会话A是幻读，其实图一的会话B才是幻读吧？

作者回复: 这些都不叫幻读，幻读的意思是“用一个事务里面，后一个请求看到的比之前相同请求看到的，多了记录出来”。

改了不算

大家关注一下这个问题。

好问题



发条橙子 ... 置顶

2018-12-15

3

老师，我这边有几个问题：

1. 看到老师回复评论说 count(id) 也是走普通索引，那是不是也算是优化了，我以为 count(字段) 是走的聚集索引。老师的意思是 count(字段) 是走二级索引，但是不一定是数据最少的索引树的意思么 ...

展开 v

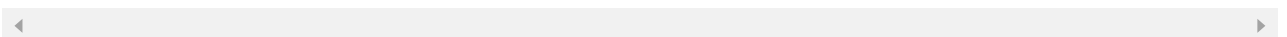
作者回复: 1. 如果有索引用到这个字段的话，比较大可能会用到这个索引，比主键索引小

2. 索引字段就算是NULL，上面的id也不是的

3. 进了Buffer pool 的原因吧

4. 嗯，rc用得挺多的，但是原因可能只是因为“以前是这么用的”。使用rc可能有问题，也可能没问题。但是我觉得DBA不知道为什么这么选，这个是问题。

rc本身的问题其实前面我们说过一些，比如不是一致性读。后面也会有文章说到。



果然如此 置顶

2018-12-14

2

一、请问计数用这个MySQL+redis方案如何：

1.开启事务（程序中的事务）

2.MySQL插入数据

3.原子更新redis计数

4.如果redis更新成功提交事务，如果redis更新失败回滚事务。...

展开 ∨

作者回复: 1. 好问题，不会还是没解决我们说的一致性问题。如果在3、4之间插入了 Session B的逻辑呢

2. 我估计就是启动事务（执行begin),结束时提交（执行commit)吧，没有了解过所有框架，不确定哈



北天魔狼

2018-12-14

👍 14

老师说过：事务开启后，更新操作放到最后。较少锁等待时间的影响



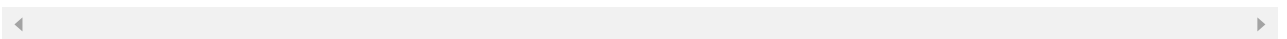
三木子

2018-12-15

👍 7

一直以为带\*查询效率是最差的，平时查询特意加了 count(ID) 查询。罪过啊。

作者回复: 😊 来得及来得及



某、人

2018-12-14

👍 7

老师我先问个本章之外的问题:

1.rr模式下,一张表上没有主键和唯一键,有二级索引c.如果是一张大表,删除一条数据delete t where c=1.

在主库上利用二级索引,在根据虚拟的主键列回表删除还挺快.但是在备库上回放特别慢,而且状态是system lock,是因为binlog event里没有包含虚拟主键列.导致在备库回放的时候...

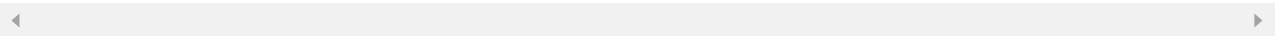
展开 ∨

作者回复: 1. 对，这个是个bug, 从库上会全表扫描。MariaDB 的版本有解决这个问题。生产上我们最好不允许没有主键的表

2. 按照你问的,gap锁没问题了。delete 被锁是因为行锁吧。从库重放就是因为走全表扫描按行锁下来触发的

3. 出现这个问题肯定是binlog设置了row格式。

这样binlog里面有所有值。如果你有主键的话，就是主键查，没有的话...就是全表了



某、人

2018-12-16

👍 6

谈谈自己的理解,有不对之处还请老师指出:

数据一致性问题目前来说主要分为三类

1.主从不一致

解决办法:半同步复制after\_commit,after\_sync,MGR(after\_prepare)。但是都不能完成满足完全实时一致,由于等待的ack点不同,相对来说一致性的强度是递增....

展开 ▾



崔根禄

2018-12-14

👍 4

老师：

1.count(\*) 不取值，

InnoDB还做遍历表的操作吗，也不用给server层返回值吗？

2.count(1) 不取值，

但是要遍历表。原文中：...

展开 ▾



斜面镜子 ...

2018-12-14

👍 4

先插入操作纪录，再更新计数表，因为计数表相当于热点行，加锁时间需要考虑足够短！



菜鸡一只

2019-02-01

👍 3

count ( id ) 和count ( 这段 ) 都是要把每一行的该字段值取出来，然后判断是否为空，那为什么count ( id ) 的效率要高？

展开 ▾

作者回复: count(id)可能会选择最小的索引来遍历

而count(字段)的话，如果字段上没有索引，就只能选主键索引



萧萧木叶

2018-12-26

👍 2

歪个楼请教个业务中遇到的问题：

表结构如下：

```
CREATE TABLE `tablename` (  
  `id` int(10) NOT NULL AUTO_INCREMENT,  
  `uid` bigint(20) NOT NULL DEFAULT '0',...
```

展开 ▾



WL

2018-12-16

👍 2

把该讲内容总结为几个问题, 大家复习的时候可以先尝试回答这些问题检查自己的掌握程度:

1.

count(\*)的实现方式在MySAM引擎和InnoDB引擎的实现方式各是怎么样的? 为什么会有这种不同?

2. ...

展开 ▾



Bin

2018-12-14

👍 2

答：先插入操作记录，再更新计数表。

在InnoDB事务中，行锁是在需要的时候才加上, 等到事务结束(commit)的时候才释放。

案例中的更新操作很多时候都是更新同一个数据对象, 如果是先更新计数表，那么持有的锁时间会更长。

展开 ▾



帆帆帆帆帆...

2018-12-14

👍 2

如果字段上有索引，且字段非空，count(字段)的效率就不是最差的了吧。

作者回复: 还是的。

注意：count(id)也是可以使用普通索引的





po

2019-01-24

1

老师你说Myisam引擎的总行数是写在磁盘上的，但是如果更新了数据，这个总行数的值是会实时改变，还是他有什么其他的策略。

展开

作者回复: 实时改变



Zzz

2019-01-22

1

老师，对于自增主键，批量插入是否存在阻塞的情况？

作者回复: 为什么会呢？不会的



小动物很困

2019-01-22

1

我记得有一个并发插入的方法,就是说计数表对同一个表开很多行,然后计数增加对这些数据随机做加法,当做count操作的时候取sum,这样对行锁竞争会削弱.

展开

作者回复: 是的，咱们专栏07篇也有类似的介绍😁



Ruian

2018-12-28

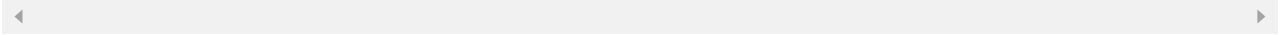
1

老师，我有一个问题请教下您。我的mysql数据库 有一张表project\_des (850万数据) count(1) 只要0.05s 和从表project ( 6500条数据 ) 关联后需要15s 查看执行计划发现都走索引了。不知道为啥这么慢？

查询语句 select count(1) from project\_des a,project b where a.project\_id=b.id 我换 exists 和in 也这么慢

展开 ▾

作者回复: 你这个是笛卡尔积肯定慢的...你的业务需求是啥, 这个语句的业务意义是什么



不忘初心

2018-12-26

👍 1

对于 count(主键 id), server层拿到ID, 判断ID是不可能为空的按行累加。这个地方, 是不是又点问题, 既然是主键ID, 是一定不会为空的, 这个server层还需要判断不为空吗

作者回复: 嗯, 代码就是这么写的 我也觉得可以优化一下... 不过现在就这样😞

