https://blog.csdn.net/heng\_yan/article/details/78324176

**mysql的sql执行计划详解（非常有用）**

2017年10月23日 22:21:40 [heng\_yan](https://me.csdn.net/heng_yan" \t "_blank) 阅读数：41880

引言：

实际项目开发中，由于我们不知道实际查询的时候数据库里发生了什么事情，数据库软件是怎样扫描表、怎样使用索引的，因此，我们能感知到的就只有

sql语句运行的时间，在数据规模不大时，查询是瞬间的，因此，在写sql语句的时候就很少考虑到性能的问题。但是当数据规模增大，如千万、亿的时候，我们运

行同样的sql语句时却发现迟迟没有结果，这个时候才知道数据规模已经限制了我们查询的速度。所以，查询优化和索引也就显得很重要了。

问题：

当我们在查询前能否预先估计查询究竟要涉及多少行、使用哪些索引、运行时间呢？答案是能的，mysql提供了相应的功能和语法来实现该功能。

分析：

MySql提供了EXPLAIN语法用来进行查询分析，在SQL语句前加一个"EXPLAIN"即可。比如我们要分析如下SQL语句：

explain select \* from table where table.id = 1 

运行上面的sql语句后你会看到，下面的表头信息：

table | type | possible\_keys | key | key\_len | ref | rows | Extra

EXPLAIN列的解释

table   
显示这一行的数据是关于哪张表的

type   
这是重要的列，显示连接使用了何种类型。从最好到最差的连接类型为const、eq\_reg、ref、range、indexhe和ALL

说明：不同连接类型的解释（按照效率高低的顺序排序）

system：表只有一行：system表。这是const连接类型的特殊情况。

const ：表中的一个记录的最大值能够匹配这个查询（索引可以是主键或惟一索引）。因为只有一行，这个值实际就是常数，因为MYSQL先读这个值然后把它当做常数来对待。

eq\_ref：在连接中，MYSQL在查询时，从前面的表中，对每一个记录的联合都从表中读取一个记录，它在查询使用了索引为主键或惟一键的全部时使用。

ref：这个连接类型只有在查询使用了不是惟一或主键的键或者是这些类型的部分（比如，利用最左边前缀）时发生。对于之前的表的每一个行联合，全部记录都将从表中读出。这个类型严重依赖于根据索引匹配的记录多少—越少越好。

range：这个连接类型使用索引返回一个范围中的行，比如使用>或<查找东西时发生的情况。

index：这个连接类型对前面的表中的每一个记录联合进行完全扫描（比ALL更好，因为索引一般小于表数据）。

ALL：这个连接类型对于前面的每一个记录联合进行完全扫描，这一般比较糟糕，应该尽量避免。

possible\_keys   
显示可能应用在这张表中的索引。如果为空，没有可能的索引。可以为相关的域从WHERE语句中选择一个合适的语句

key   
实际使用的索引。如果为NULL，则没有使用索引。很少的情况下，MYSQL会选择优化不足的索引。这种情况下，可以在SELECT语句中使用USE INDEX（indexname）来强制使用一个索引或者用IGNORE INDEX（indexname）来强制MYSQL忽略索引

key\_len   
使用的索引的长度。在不损失精确性的情况下，长度越短越好

ref   
显示索引的哪一列被使用了，如果可能的话，是一个常数

rows   
MYSQL认为必须检查的用来返回请求数据的行数

Extra   
关于MYSQL如何解析查询的额外信息。将在表4.3中讨论，但这里可以看到的坏的例子是Using temporary和Using filesort，意思MYSQL根本不能使用索引，结果是检索会很慢

说明：extra列返回的描述的意义

Distinct ：一旦mysql找到了与行相联合匹配的行，就不再搜索了。

Not exists ：mysql优化了LEFT JOIN，一旦它找到了匹配LEFT JOIN标准的行，就不再搜索了。

Range checked for each Record（index map:#） ：没有找到理想的索引，因此对从前面表中来的每一个行组合，mysql检查使用哪个索引，并用它来从表中返回行。这是使用索引的最慢的连接之一。

Using filesort ：看到这个的时候，查询就需要优化了。mysql需要进行额外的步骤来发现如何对返回的行排序。它根据连接类型以及存储排序键值和匹配条件的全部行的行指针来排序全部行。

Using index ：列数据是从仅仅使用了索引中的信息而没有读取实际的行动的表返回的，这发生在对表的全部的请求列都是同一个索引的部分的时候。

Using temporary ：看到这个的时候，查询需要优化了。这里，mysql需要创建一个临时表来存储结果，这通常发生在对不同的列集进行ORDER BY上，而不是GROUP BY上。

Where used ：使用了WHERE从句来限制哪些行将与下一张表匹配或者是返回给用户。如果不想返回表中的全部行，并且连接类型ALL或index，这就会发生，或者是查询有问题。

因此，弄明白了explain语法返回的每一项结果，我们就能知道查询大致的运行时间了，如果查询里没有用到索引、或者需要扫描的行过多，那么可以感到明显的延迟。因此需要改变查询方式或者新建索引。mysql中的explain语法可以帮助我们改写查询，优化表的结构和索引的设置，从而最大地提高查询效率。当然，在大规模数据量时，索引的建立和维护的代价也是很高的，往往需要较长的时间和较大的空间，如果在不同的列组合上建立索引，空间的开销会更大。因此索引最好设置在需要经常查询的字段中。