链滴

Mysql explain sql 语句性能分析

作者：hongcha516

原文链接：https://ld246.com/article/1535298749533

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

使用 explain 关键字可以模拟优化器执行 sql 查询语句，通过 explain 的参数绍，我们可以得知：

1. 表的读取顺序 (id)
2. 数据读取操作的操作类型 (type)
3. 哪些索引被实际使用 (key)
4. 表之间的引用 (ref)
5. 每张表有多少行被优化器查询 (rows)

使用方式，在sql语句前面增加关键字explain，例如：

explain select * from test where id=1;

explain参数详细说明如下：

id

select 查询的序列号，包含一组可以重复的数字，表示查询中执行 sql 语句的顺序。一般有三种情况：
第一种：id 全部相同，sql 的执行顺序是由上至下；
第二种：id 全部不同，sql 的执行顺序是根据 id 大的优先执行；
第三种：id 既存在相同，又存在不同的。先根据 id 大的优先执行，再根据相同 id 从上至下的执行。

select_type

select 查询的类型，主要是用于区别普通查询，联合查询，嵌套的复杂查询
simple：简单的 select 查询，查询中不包含子查询或者 union
primary：查询中若包含任何复杂的子查询，最外层查询则被标记为 primary
subquery：在 select 或 where 列表中包含了子查询
derived：在 from 列表中包含的子查询被标记为 derived（衍生）MySQL 会递归执行这些子查询，结果放在临时表里。
union：若第二个 select 出现在 union 之后，则被标记为 union，若 union 包含在 from 子句的子询中，外层 select 将被标记为：derived
union result：从 union 表获取结果的 select

partitions

表所使用的分区，如果要统计十年公司订单的金额，可以把数据分为十个区，每一年代表一个区。这可以大大的提高查询效率。

type

这是一个非常重要的参数，连接类型，常见的有：all , index , range , ref , eq_ref , const , system , ull 八个级别。
性能从最优到最差的排序：system > const > eq_ref > ref > range > index > all
对 java 程序员来说，若保证查询至少达到 range 级别或者最好能达到 ref 则算是一个优秀而又负责程序员。
all：（full table scan）全表扫描无疑是最差，若是百万千万级数据量，全表扫描会非常慢。
index：（full index scan）全索引文件扫描比 all 好很多，毕竟从索引树中找数据，比从全表中找数要快。
range：只检索给定范围的行，使用索引来匹配行。范围缩小了，当然比全表扫描和全索引文件扫描快。sql 语句中一般会有 between，in，>，< 等查询。
ref：非唯一性索引扫描，本质上也是一种索引访问，返回所有匹配某个单独值的行。比如查询公司所

属于研发团队的同事，匹配的结果是多个并非唯一值。
eq_ref：唯一性索引扫描，对于每个索引键，表中有一条记录与之匹配。比如查询公司的 CEO，匹配结果只可能是一条记录，
const：表示通过索引一次就可以找到，const 用于比较 primary key 或者 unique 索引。因为只匹一行数据，所以很快，若将主键至于 where 列表中，MySQL 就能将该查询转换为一个常量。
system：表只有一条记录（等于系统表），这是 const 类型的特列，平时不会出现，了解即可

possible_keys

显示查询语句可能用到的索引 (一个或多个或为 null)，不一定被查询实际使用。仅供参考使用。

key

显示查询语句实际使用的索引。若为 null，则表示没有使用索引。

key_len

显示索引中使用的字节数，可通过 key_len 计算查询中使用的索引长度。在不损失精确性的情况下索长度越短越好。key_len 显示的值为索引字段的最可能长度，并非实际使用长度，即 key_len 是根据定义计算而得，并不是通过表内检索出的。

ref

显示索引的哪一列或常量被用于查找索引列上的值。

rows

根据表统计信息及索引选用情况，大致估算出找到所需的记录所需要读取的行数，值越大越不好。

extra

Using filesort： 说明 MySQL 会对数据使用一个外部的索引排序，而不是按照表内的索引顺序进行取。MySQL 中无法利用索引完成的排序操作称为“文件排序” 。出现这个就要立刻优化 sql。
Using temporary： 使用了临时表保存中间结果，MySQL 在对查询结果排序时使用临时表。常见于序 order by 和 分组查询 group by。 出现这个更要立刻优化 sql。
Using index： 表示相应的 select 操作中使用了覆盖索引（Covering index），避免访问了表的数行，效果不错！如果同时出现 Using where，表明索引被用来执行索引键值的查找。如果没有同时出 Using where，表示索引用来读取数据而非执行查找动作。
覆盖索引（Covering Index）：也叫索引覆盖，就是 select 的数据列只用从索引中就能够取得，不读取数据行，MySQL 可以利用索引返回 select 列表中的字段，而不必根据索引再次读取数据文件。
Using index condition： 在 5.6 版本后加入的新特性，优化器会在索引存在的情况下，通过符合 RAGE 范围的条数 和 总数的比例来选择是使用索引还是进行全表遍历。
Using where： 表明使用了 where 过滤
Using join buffer： 表明使用了连接缓存
impossible where： where 语句的值总是 false，不可用，不能用来获取任何元素
distinct： 优化 distinct 操作，在找到第一匹配的元组后即停止找同样值的动作

filtered

一个百分比的值，和 rows 列的值一起使用，可以估计出查询执行计划 (QEP) 中的前一个表的结果集从而确定 join 操作的循环次数。小表驱动大表，减轻连接的次数。

参考地址

https://hacpai.com/article/1535034154103#toc_h3_13

http://www.cnblogs.com/itdragon/p/8146439.html