**MySQL EXPLAIN详解**

[](http://www.jianshu.com/u/2766e4cfc391)



image.png

MySQL EXPLAIN命令是查询性能优化不可缺少的一部分，该文主要讲解explain命令的使用及相关参数说明。

**EXPLAIN Output Columns**

| **列名** | **说明** |
| --- | --- |
| id | 执行编号，标识select所属的行。如果在语句中没子查询或关联（left join 或者right join）查询，只有唯一的select，每行都将显示1。否则，内层的select语句一般会顺序编号，对应于其在原始语句中的位置 |
| select\_type | 显示本行是简单（simple）或复杂select。如果查询有任何复杂的子查询，则最外层标记为PRIMARY（DERIVED、UNION、UNION RESUlT） |
| table | 访问引用哪个表（引用某个查询，如“derived3”） |
| partitions |  |
| type | 数据访问/读取操作类型（ALL、index、range、ref、eq\_ref、const/system、NULL） |
| possible\_keys | 揭示哪一些索引可能有利于高效的查找 |
| key | 显示mysql决定采用哪个索引来优化查询 |
| key\_len | 显示mysql在索引里使用的字节数 |
| ref | 显示了之前的表在key列记录的索引中查找值所用的列或常量 |
| rows | 为了找到所需的行而需要读取的行数，估算值，不精确。通过把所有rows列值相乘，可粗略估算整个查询会检查的行数 |
| Extra | 额外信息，如using index、filesort等 |

**单条查询语句：**

mysql> explain select name from thing\_device;

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| 1 | SIMPLE | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11016 | 100.00 | NULL |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

**联合查询：**

mysql> explain select a.name,b.name from thing\_device a left join thing\_group b on a.group\_id = b.id limit 10;

+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+-------+----------+-------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+-------+----------+-------+

| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL | NULL | 11016 | 100.00 | NULL |

| 1 | SIMPLE | b | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | htiiot.a.group\_id | 1 | 100.00 | NULL |

+----+-------------+-------+------------+--------+---------------+---------+---------+-------------------+-------+----------+-------+

**子查询：**

mysql> explain select a.name from thing\_device a where id in (select b.id from thing\_device b where tenant\_id=1) limit 10;

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------------+

| 1 | SIMPLE | b | NULL | ref | PRIMARY,idx\_thing\_device\_user\_id\_type,idx\_thing\_device\_serialno,idx\_thing\_device\_tenant\_id\_deleted\_type | idx\_thing\_device\_user\_id\_type | 4 | const | 772 | 100.00 | Using index |

| 1 | SIMPLE | a | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | htiiot.b.id | 1 | 100.00 | NULL |

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------------+

**Union查询**

mysql> explain select name from thing\_device union select name from thing\_group;

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| 1 | PRIMARY | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11016 | 100.00 | NULL |

| 2 | UNION | thing\_group | NULL | ALL | NULL | NULL | NULL | NULL | 1710 | 100.00 | NULL |

| NULL | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

**使用order by Extra显示filesort:**

mysql> explain select a.name from thing\_device a where id in (select b.id from thing\_device b where tenant\_id=1) order by a.id desc;

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+----------------------------------------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+----------------------------------------------+

| 1 | SIMPLE | b | NULL | ref | PRIMARY,idx\_thing\_device\_user\_id\_type,idx\_thing\_device\_serialno,idx\_thing\_device\_tenant\_id\_deleted\_type | idx\_thing\_device\_user\_id\_type | 4 | const | 772 | 100.00 | Using index; Using temporary; Using filesort |

| 1 | SIMPLE | a | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | htiiot.b.id | 1 | 100.00 | NULL |

+----+-------------+-------+------------+--------+---------------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+----------------------------------------------+

2 rows in set, 1 warning (0.01 sec)

**id**

id是用来顺序标识整个查询中SELELCT 语句的，在嵌套查询中id越大的语句越先执行。该值可能为NULL，如果为NULL则表示这一行用来说明的是其他行的联合结果（如使用union时会最后的union操作作为一个select，其id值为null）。

**select\_type**

表示查询的类型

| **类型** | **说明** |
| --- | --- |
| simple | 简单子查询，不包含子查询和union |
| primary | 包含union或者子查询，最外层的部分标记为primary |
| subquery | 一般子查询中的子查询被标记为subquery，也就是位于select列表中的查询 |
| derived | 派生表——该临时表是从子查询派生出来的，位于form中的子查询 |
| union | 位于union中第二个及其以后的子查询被标记为union，第一个就被标记为primary如果是union位于from中则标记为derived |
| union result | 用来从匿名临时表里检索结果的select被标记为union result |
| dependent union | 顾名思义，首先需要满足UNION的条件，及UNION中第二个以及后面的SELECT语句，同时该语句依赖外部的查询 |
| subquery | 子查询中第一个SELECT语句 |
| dependent subquery | 和DEPENDENT UNION相对UNION一样 |

Simple:

**mysql> explain select name from thing\_device limit 10;**

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| 1 | SIMPLE | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11020 | 100.00 | NULL |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

1 row in set, 1 warning (0.01 sec)

**mysql> explain select count(\*) from thing\_dev**ice;

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+------------------------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+------------------------------+

| 1 | SIMPLE | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | Select tables optimized away |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+------------------------------+

1 row in set, 1 warning (0.01 sec)

Rows为null说明一个表的count是被数据库保存在表的信息里面的

**mysql> explain select name from thing\_device union select name from thing\_group limit 10;**

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| 1 | PRIMARY | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11020 | 100.00 | NULL |

| 2 | UNION | thing\_group | NULL | ALL | NULL | NULL | NULL | NULL | 1710 | 100.00 | NULL |

| NULL | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

3 rows in set, 1 warning (0.02 sec)

**mysql> explain select name from (select id,name from thing\_device limit 10) a;**

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

| 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL | NULL | 10 | 100.00 | NULL |

| 2 | DERIVED | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11020 | 100.00 | NULL |

+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+

2 rows in set, 1 warning (0.01 sec)

**table**

对应行正在访问哪一个表，表名或者别名（me：有别名的话就用别名）

* 关联优化器会为查询选择关联顺序，左侧深度优先
* 当from中有子查询的时候，表名是derivedN的形式，N指向子查询，也就是explain结果中的下一列
* 当有union result的时候，表名是union 1,2等的形式，1,2表示参与union的query id

mysql> explain select name from thing\_device union select name from thing\_group limit 10;

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

| 1 | PRIMARY | thing\_device | NULL | ALL | NULL | NULL | NULL | NULL | 11020 | 100.00 | NULL |

| 2 | UNION | thing\_group | NULL | ALL | NULL | NULL | NULL | NULL | 1710 | 100.00 | NULL |

| NULL | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |

+----+--------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+

注意：MySQL对待这些表和普通表一样，但是这些“临时表”是没有任何索引的。

有别名的时候使用别名

**mysql> explain select a.id,a.name,b.name from thing\_device a,thing\_group b limit 10;**

+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+---------------------------------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+---------------------------------------+

| 1 | SIMPLE | b | NULL | ALL | NULL | NULL | NULL | NULL | 1710 | 100.00 | NULL |

| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL | NULL | 11020 | 100.00 | Using join buffer (Block Nested Loop) |

+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+---------------------------------------+

2 rows in set, 1 warning (0.01 sec)

**type**

type显示的是访问类型，是较为重要的一个指标，结果值从好到坏依次是：  
system > const > eq\_ref > ref > fulltext > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL ，一般来说，得保证查询至少达到range级别，最好能达到ref。

| **类型** | **说明** |
| --- | --- |
| All | 最坏的情况,全表扫描 |
| index | 和全表扫描一样。只是扫描表的时候按照索引次序进行而不是行（不是使用索引扫描全表）。主要优点就是避免了排序, 但是开销仍然非常大。如在Extra列看到Using index，说明正在使用覆盖索引，只扫描索引的数据，它比按索引次序全表扫描的开销要小很多 |
| range | 范围扫描，一个有限制的索引扫描。key 列显示使用了哪个索引。当使用=、 <>、>、>=、<、<=、IS NULL、<=>、BETWEEN 或者 IN 操作符,用常量比较关键字列时,可以使用 range（me：当使用了范围比较where条件时） |
| ref | 一种索引访问，它返回所有匹配某个单个值的行。此类索引访问只有当使用非唯一性索引或唯一性索引非唯一性前缀时才会发生。这个类型跟eq\_ref不同的是，它用在关联操作只使用了索引的最左前缀，或者索引不是UNIQUE和PRIMARY KEY。ref可以用于使用=或<=>操作符的带索引的列。 |
| eq\_ref | 最多只返回一条符合条件的记录。使用唯一性索引或主键查找时会发生 （高效） |
| const | 当确定最多只会有一行匹配的时候，MySQL优化器会在查询前读取它而且只读取一次，因此非常快。当主键放入where子句时，mysql把这个查询转为一个常量（高效） |
| system | 这是const连接类型的一种特例，表仅有一行满足条件。 |
| Null | 意味说mysql能在优化阶段分解查询语句，在执行阶段甚至用不到访问表或索引（高效） |

Ref：

**mysql> explain select \* from thing\_device where tenant\_id=1 and type=0;**

+----+-------------+--------------+------------+------+-------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+--------------+------------+------+-------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------+

| 1 | SIMPLE | thing\_device | NULL | ref | idx\_thing\_device\_user\_id\_type,idx\_thing\_device\_serialno,idx\_thing\_device\_tenant\_id\_deleted\_type | idx\_thing\_device\_user\_id\_type | 5 | const,const | 663 | 100.00 | NULL |

+----+-------------+--------------+------------+------+-------------------------------------------------------------------------------------------------+-------------------------------+---------+-------------+------+----------+-------+

1 row in set, 1 warning (0.01 sec)

Range

**mysql> explain select \* from thing\_device where id between 0 and 100;**

+----+-------------+--------------+------------+-------+---------------+---------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+--------------+------------+-------+---------------+---------+---------+------+------+----------+-------------+

| 1 | SIMPLE | thing\_device | NULL | range | PRIMARY | PRIMARY | 4 | NULL | 12 | 100.00 | Using where |

+----+-------------+--------------+------------+-------+---------------+---------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.01 sec)

**possible\_keys**

显示查询使用了哪些索引，表示该索引可以进行高效地查找，但是列出来的索引对于后续优化过程可能是没有用的

**key**

key列显示MySQL实际决定使用的键（索引）。如果没有选择索引，键是NULL。要想强制MySQL使用或忽视possible\_keys列中的索引，在查询中使用FORCE INDEX、USE INDEX或者IGNORE INDEX。

**key\_len**

key\_len列显示MySQL决定使用的键长度。如果键是NULL，则长度为NULL。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**

ref列显示使用哪个列或常数与key一起从表中选择行。

**rows**

rows列显示MySQL认为它执行查询时必须检查的行数。注意这是一个预估值。

**Extra**

Extra是EXPLAIN输出中另外一个很重要的列，该列显示MySQL在查询过程中的一些详细信息，MySQL查询优化器执行查询的过程中对查询计划的重要补充信息。

| **类型** | **说明** |
| --- | --- |
| Using filesort | MySQL有两种方式可以生成有序的结果，通过排序操作或者使用索引，当Extra中出现了Using filesort 说明MySQL使用了后者，但注意虽然叫filesort但并不是说明就是用了文件来进行排序，只要可能排序都是在内存里完成的。大部分情况下利用索引排序更快，所以一般这时也要考虑优化查询了。使用文件完成排序操作，这是可能是ordery by，group by语句的结果，这可能是一个CPU密集型的过程，可以通过选择合适的索引来改进性能，用索引来为查询结果排序。 |
| Using temporary | 用临时表保存中间结果，常用于GROUP BY 和 ORDER BY操作中，一般看到它说明查询需要优化了，就算避免不了临时表的使用也要尽量避免硬盘临时表的使用。 |
| 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。 |
| Using index | 说明查询是覆盖了索引的，不需要读取数据文件，从索引树（索引文件）中即可获得信息。如果同时出现using where，表明索引被用来执行索引键值的查找，没有using where，表明索引用来读取数据而非执行查找动作。这是MySQL服务层完成的，但无需再回表查询记录。 |
| Using index condition | 这是MySQL 5.6出来的新特性，叫做“索引条件推送”。简单说一点就是MySQL原来在索引上是不能执行如like这样的操作的，但是现在可以了，这样减少了不必要的IO操作，但是只能用在二级索引上。 |
| Using where | 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。**注意**：Extra列出现Using where表示MySQL服务器将存储引擎返回服务层以后再应用WHERE条件过滤。 |
| Using join buffer | 使用了连接缓存：**Block Nested Loop**，连接算法是块嵌套循环连接;**Batched Key Access**，连接算法是批量索引连接 |
| impossible where | where子句的值总是false，不能用来获取任何元组 |
| select tables optimized away | 在没有GROUP BY子句的情况下，基于索引优化MIN/MAX操作，或者对于MyISAM存储引擎优化COUNT(\*)操作，不必等到执行阶段再进行计算，查询执行计划生成的阶段即完成优化。 |
| 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 |

**参考资料**

高性能mysql第三版  
<https://dev.mysql.com/doc/refman/5.7/en/explain-output.html>