MySQL EXPLAINN 执行计划

作者：Hassan
原文链接：https://ld246.com/article/1471600801730
来源网站：链滴

EXPLAIN 用来查看MySQL执行一个SQL语句的执行计划。
<h2>语法</h2>

<pre class="brush: sql">{EXPLAIN | DESCRIBE | DESC}
    tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC}
[explain_type]
{explainable_stmt | FOR CONNECTION connection_id}

explain_type: {
EXTENDED
| PARTITIONS
| FORMAT = format_name
}

format_name: {
TRADITIONAL
| JSON
}

explainable_stmt: {
SELECT statement
| DELETE statement
| INSERT statement
| REPLACE statement
| UPDATE statement
}

</pre>

简单说一下这个语法怎么看，有些初学者可能会看不懂。 语法里，{}表示一个语句块；| 表示或意思，就是说可以用EXPLAIN 或 DESCRIBE 或 DESC的意思；[]这个表示可选项，就是可有可无的意。 其中，explain_type，explainable_stmt格式要怎么写，可以写什么内容，分别参考下面给出的explainable_stmt：{}等对应的包裹的格式。 比如：EXPLAIN table_name_1; 再比如：EXPLAIN EXTENED SELECT * FROM table_name_2; 再或者：EXPLAIN FORMAT = JSON SELECT * FROM table_name_3;
EXPLAIN和DESCRIBE 、DESC 是同义词。执行结果是一样的。但是习惯上，会使用DESCRIBE DESC 来查询表结构信息；用EXPLAIN来查看MySQL执行计划。（In practice, the DESCRIBE keyword is more often used to obtain information about table structure, whereas EXPLAIN is used to obtain a query execution plan）。
<h2>使用</h2>
获取表结构信息：

<pre class="brush: bash">mysql> DESCRIBE City;
+-----------+----------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |

```
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | NO   |     |         |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | NO   |     | 0       |                |
+------------+----------+------+-----+---------+----------------+</pre>
</div>
<p>查看MySQL执行计划：</p>
<div class="highlight highlight-source-shell">
<pre class="brush: bash">mysql&gt; EXPLAIN SELECT * FROM city ;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+-
--------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered |
xtra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+-
--------+-------+
|  1 | SIMPLE      | city  | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 4188 |   100.00 |
ULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+-
--------+-------+
1 row in set, 1 warning (0.00 sec)

mysql&gt; EXPLAIN EXTENDED SELECT * FROM city ;

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

| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered |
xtra |

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

|  1 | SIMPLE      | city  | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 4188 |   100.00 |
ULL  |

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

1 row in set, 2 warnings (0.00 sec)</pre>

</div>
<p>EXPLAIN 加上 EXTENDED 关键字的意思的在执行计划结果字段里显示：filtered 字段（什么意
后面会继续说）。但是细看2中的例子，会发现，没有加EXTENDED关键字也显示了filtered字段。这
因为在5.7.3以后，EXPLAIN的默认处理就是按存在EXTENDED关键字来处理的。所以不用加EXTEND
D也会显示filtered字段。 在细心的看一下2中查看MySQL执行计划的两次结果提示，含有EXTENDE
的查询有两个warnings，而不含EXTENDED的只有一个warnings。 我们用SHOW WARNINGS; 查
一下提示信息：</p>
<div class="highlight highlight-source-shell">
<pre class="brush: bash">mysql&gt; SHOW WARNINGS;
+---------+------+--------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
------------------------------------------+
| Level   | Code | Message
                                                                           |
+---------+------+--------------------------------------------------------------------------------
```

原文链接：MySQL EXPLAINN 执行计划

```
-----------------------------------------------------------------------------------------------------------------
----------------------------------------------+
| Warning | 1681 | 'EXTENDED' is deprecated and will be removed in a future release.
                                                                                                        |
| Note    | 1003 | /* select#1 */ select `world`.`city`.`ID` AS `ID`,`world`.`city`.`Name` AS `Name`,`world`.`city`.`CountryCode` AS `CountryCode`,`world`.`city`.`District` AS `District`,`world`.`city`.`Population` AS `Population` from `world`.`city` |
+---------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
----------------------------------------------+
2 rows in set (0.00 sec)</pre>
</div>
<p>'EXTENDED' 在未来版本将被弃用。so...</p>
<p>另外，2中的两次查询都有 1003这个warning。</p>
<p>这里再试试：</p>
<div class="highlight highlight-source-shell">
<pre class="brush: bash">mysql&gt;  SELECT * FROM city WHERE ID=1;
+----+-------+-------------+----------+------------+
| ID | Name  | CountryCode | District | Population |
+----+-------+-------------+----------+------------+
|  1 | Kabul | AFG         | Kabol    |    1780000 |
+----+-------+-------------+----------+------------+
1 row in set (0.00 sec)
```

```
mysql&gt; EXPLAIN SELECT * FROM city WHERE ID=1;

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

| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra |

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

|  1 | SIMPLE      | city  | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | NULL  |

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

1 row in set, 1 warning (0.00 sec)

mysql&gt; SHOW WARNINGS;

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

| Level | Code | Message                                                                                                                                                  |

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

| Note  | 1003 | /* select#1 */ select '1' AS ID,'Kabul' AS Name,'AFG' AS CountryCode,'Kabol' AS District,'1780000' AS Population from world.city where 1 |

+-------+------+------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

原文链接：MySQL EXPLAINN 执行计划

1 row in set (0.00 sec)</pre>

同样可以看见1003这个warning，再看Message字段的内容，sql语句条件变成了where 1，并且elect字段也变成了“结果值 AS 字段名”的样子。 这个警告信息，只是起提示作用的，诉你一些信息，比如说：执行计划使用“const” 访问类型，将列值转换为常量来处理。 Message的语句并不等同于原始SQL语句，也不等同于被优化器执行的语句，并且不一定是能被行的语句。 （原文：Because the statement displayed by SHOW WARNINGS may contain speci l markers to provide information about query rewriting or optimizer actions, the statement is ot necessarily valid SQL and is not intended to be executed. The output may also include rows with Message values that provide additional non-SQL explanatory notes about actions taken y the optimizer.）
扩展阅读：https://www.percona.com/blog/2006/07/24/extended-explain/
<h2>EXPLAIN 输出的列说明</h2>
<table>
<thead>
<tr><th>列名</th><th>JSON 名称</th></tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>select_id</td>
</tr>
<tr>
<td>select_type</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>table_name</td>
</tr>
<tr>
<td>partitions</td>
<td>partitions</td>
</tr>
<tr>
<td>type</td>
<td>access_type</td>
</tr>
<tr>
<td>possible_keys</td>
<td>possible_keys</td>
</tr>
<tr>
<td>key</td>
<td>key</td>
</tr>
<tr>
<td>key_len</td>
<td>key_length</td>
</tr>
<tr>
<td>ref</td>

<td>ref</td>
</tr>
<tr>
<td>rows</td>
<td>rows</td>
</tr>
<tr>
<td>filtered</td>
<td>filtered</td>
</tr>
<tr>
<td>Extra</td>
<td>None</td>
</tr>
</tbody>
</table>

<h4>id</h4>
Optimizer 执行计划中查询的序列号。表示查询中执行 select 子句或操作表的顺序，id 值越大先级越高，越先被执行。id 相同，执行顺序由上至下。
如果行数据通过union关联了其他的行数据，则该值为NULL。这个时候table 列的值会设置成 &lt unionM,N>形式，表示id为M和N的结果进行了union。
<h4>select_type</h4>
通过这个类型来标识一条语句中不同位置的SQL语句。
SIMPLE--简单的SELECT查询，没有使用UNION或子查询。
PRIMARY--嵌套查询时，最外层的SELECT语句；在UNION查询时，最前面的SELECT语句。</p

UNION--UNION中第二个以及后面的SELECT语句。
UNION RESULT--一个UNION查询的结果。
SUBQUERY--子查询中第一个SELECT语句。
DERIVED--FROM子查询中的SELECT语句。
DEPENDENT SUBQUERY--子查询中的第一个SELECT语句，并且是依赖外部查询的。
DEPENDENT UNION--UNION中的第二个以及之后的SELECT语句，并且是依赖外部查询的。<
p>
DEPENDENT的这两个类型有必要写个sql来看看，先看一下下面这个sql的执行计划：</
pan>
<pre class="brush: bash">mysql> explain select a.articleTitle from b3_solo_article a where
.oid in
>(select commentOnId from b3_solo_comment where oid='1471598068326'
nion select commentOnId from b3_solo_comment where oid='1471622574286');
+----+-------------------+----------------+-------+--------------+---------+---------+-------+
-----+-----------------+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra
     |
+----+-------------------+----------------+-------+--------------+---------+---------+-------+
-----+-----------------+
| 1 | PRIMARY | a | ALL | NULL | NULL | NULL | NULL | 4 | Using w
ere |
| 2 | DEPENDENT SUBQUERY | b3_solo_comment | const | PRIMARY | PRIMARY | 767 | c
nst | 1 | NULL |
| 3 | DEPENDENT UNION | b3_solo_comment | const | PRIMARY | PRIMARY | 767 | con
t | 1 | NULL |
| NULL | UNION RESULT | <union2,3> | ALL | NULL | NULL | NULL | NULL
 | NULL | Using temporary |
+----+-------------------+----------------+-------+--------------+---------+---------+-------+

-----+-----------------+
4 rows in set (0.00 sec)</pre>
可以看到，
id为3的select_type是DEPENDENT UNION，所以3对应的就是select commentOnId from b3_olo_comment where oid='1471622574286'；
id为2的select_type是DEPENDENT SUBQUERY，所以2对于的就是select commentOnId from b3_solo_comment where oid='1471598068326'；
table <union2,3>，就表示id为2和3的结果进行了union。
你肯能注意到了，上面DEPENDENT类型的描述“并且是依赖外部查询的”，但是和3的查询看起来并不依赖于外部。
这是因为Optimizer对实际执行的sql语句做优化，把in中的非关联的子查询转成了关联子查询（a href="http://dev.mysql.com/doc/refman/5.7/en/correlated-subqueries.html" target="_blank>Correlated Subqueries）。
SELECT ... FROM t1 WHERE t1.a IN (SELECT b FROM t2);
转成了
SELECT ... FROM t1 WHERE EXISTS (SELECT 1 FROM t2 WHERE t2.b = t1.a);
所以转换以后，就变成子查询依赖外部查询了。
<h4>table</h4>
标识查询的表信息，值可能是：
表名
指定的别名
<unionM,N> 上面已经说到过
<derivedN> 使用id为N的表作为此衍生表
<subqueryN> 使用id为N的子查询作为次衍生表
<h4>partitions</h4>
使用EXPLAIN PARTITIONS 关键字会显示出partitions列，显示表的分区信息。参考：http://devmysql.com/doc/refman/5.6/en/partitioning-info.html
<h4>type</h4>
system-- 系统表，一行匹配，const的特例
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一次，因此非常快。const只会用在将常量和主键或唯一索引进行比较时，而且是比较所有的索引字。
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL
<h4>possible_keys</h4>

<h4>key</h4>

<h4>key_len</h4>

<h4>ref</h4>

<h4>rows</h4>

<h4>filtered</h4>

<h4>Extra</h4>

更详细的信息参考：http://dev.mysql.com/doc/refman/5.7/en/explain-output.html