EXPLAIN输出信息详解
- id
- select_type
- table
- type
- possible_keys
- key
- key_len
- ref
- rows
- Extra

EXPLAIN输出信息详解

EXPLAIN的每个输出行提供一个表的相关信息，并且每个行包括下面的列

```
mysql> explain select * from mysql.user\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: user
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 5
        Extra:
1 row in set (0.00 sec)
```

Column	Meaning
`id`	The `SELECT` identifier
`select_type`	The `SELECT` type
`table`	The table for the output row
`partitions`	The matching partitions
`type`	The join type
`possible_keys`	The possible indexes to choose
`key`	The index actually chosen
`key_len`	The length of the chosen key
`ref`	The columns compared to the index
`rows`	Estimate of rows to be examined
`filtered`	Percentage of rows filtered by table condition
`Extra`	Additional information

id

SELECT识别符。这是SELECT的查询序列号。

select_type

SELECT类型，可以为以下任何一种

```
1. SIMPLE              ## 简单SELECT(不使用UNION或子查询)
2. PRIMARY             ## 最外面的SELECT
3. UNION               ## UNION中的第二个或后面的SELECT语句
4. DEPENDENT UNION     ## UNION中的第二个或后面的SELECT语句，取决于外面的查询
5. UNION RESULT        ## UNION的结果。
6. SUBQUERY            ## 子查询中的第一个SELECT
7. DEPENDENT SUBQUERY ## 子查询中的第一个SELECT，取决于外面的查询
8. DERIVED             ## 导出表的SELECT(FROM子句的子查询)
```

table

输出的行所引用的表。

type

> 联接类型。下面给出各种联接类型，按照从最佳类型到最坏类型进行排序

1. system 表仅有一行(=系统表)。这是const联接类型的一个特例。
2. const

表最多有一个匹配行，它将在查询开始时被读取。因为仅有一行，在这行的列值可被优化器剩余部分认为是常数。const表很快，因为它们只读取一次！ const用于用常数值比较PRIMARY KEY或UNIQUE索引的所有部分时。

在下面的查询中，tbl_name可以用于const表：

```
SELECT * from tbl_name WHERE primary_key=1;
SELECT * from tbl_name WHERE primary_key_part1=1和 primary_key_part2=2;
```

3. eq_ref

对于每个来自于前面的表的行组合，从该表中读取一行。这可能是最好的联接类型，除了const类型。它用在一个索引的所有部分被联接使用并且索引是UNIQUE或PRIMARY KEY。
eq_ref可以用于使用= 操作符比较的带索引的列。比较值可以为常量或一个使用在该表前面所读取的表的列的表达式。

在下面的例子中，MySQL可以使用eq_ref联接来处理ref_tables：

```
SELECT * FROM ref_table,other_table WHERE ref_table.key_column=other_table.column;
SELECT * FROM ref_table,other_table WHERE ref_table.key_column_part1=other_table.column AND
ref_table.key_column_part2=1;
```

4. ref

对于每个来自于前面的表的行组合，所有有匹配索引值的行将从这张表中读取。如果联接只使用键的最左边的前缀，或如果键不是UNIQUE或PRIMARY KEY（换句话说，如果联接不能基于关键字选择单个行的话），则使用ref。
如果使用的键仅仅匹配少量行，该联接类型是不错的。
ref可以用于使用=或<=>操作符的带索引的列。

在下面的例子中，MySQL可以使用ref联接来处理ref_tables：

```
SELECT * FROM ref_table WHERE key_column=expr;
SELECT * FROM ref_table,other_table WHERE ref_table.key_column=other_table.column;
SELECT * FROM ref_table,other_table WHERE ref_table.key_column_part1=other_table.column AND
ref_table.key_column_part2=1;
```

5. ref_or_null

该联接类型如同ref，但是添加了MySQL可以专门搜索包含NULL值的行。在解决子查询中经常使用该联接类型的优化。
在下面的例子中，MySQL可以使用ref_or_null联接来处理ref_tables：

```
SELECT * FROM ref_table WHERE key_column=expr OR key_column IS NULL;
```

6. index_merge

该联接类型表示使用了索引合并优化方法。在这种情况下，key列包含了使用的索引的清单，key_len包含了使用的索引的最长的关键元素。详细信息参见7.2.6节，“索引合并优化”。

7. unique_subquery

该类型替换了下面形式的IN子查询的ref：

```
value IN (SELECT primary_key FROM single_table WHERE some_expr)
```

unique_subquery是一个索引查找函数，可以完全替换子查询，效率更高。

8. index_subquery

该联接类型类似于unique_subquery。可以替换IN子查询，但只适合下列形式的子查询中的非唯一索引：

```
value IN (SELECT key_column FROM single_table WHERE some_expr)
```

9. range

只检索给定范围的行，使用一个索引来选择行。key列显示使用了哪个索引。key_len包含所使用索引的最长关键元素。在该类型中ref列为NULL。

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：

```
SELECT * FROM tbl_name WHERE key_column = 10;
SELECT * FROM tbl_name WHERE key_column BETWEEN 10 and 20;
SELECT * FROM tbl_name WHERE key_column IN (10,20,30);
SELECT * FROM tbl_name WHERE key_part1= 10 AND key_part2 IN (10,20,30);
```

10. index

该联接类型与ALL相同，除了只有索引树被扫描。这通常比ALL快，因为索引文件通常比数据文件小。

当查询只使用作为单索引一部分的列时，MySQL可以使用该联接类型。

11. ALL

```
对于每个来自于先前的表的行组合，进行完整的表扫描。如果表是第一个没标记const的表，这通常不好，并且通常在它情况下很差。通常可以增加更多的索引而不要使用ALL，使得行能基于前面的表中的常数值或列值被检索出。
```

possible_keys

possible_keys列指出MySQL能使用哪个索引在该表中找到行。注意，该列完全独立于EXPLAIN输出所示的表的次序。这意味着在possible_keys中的某些键实际上不能按生成的表次序使用。

如果该列是NULL，则没有相关的索引。在这种情况下，可以通过检查WHERE子句看是否它引用某些列或适合索引的列来提高你的查询性能。如果是这样，创造一个适当的索引并且再次用EXPLAIN检查查询。

为了看清一张表有什么索引，使用SHOW INDEX FROM tbl_name。

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。参见13.2.7节，“SELECT语法”。

对于MyISAM和BDB表，运行ANALYZE TABLE可以帮助优化器选择更好的索引。对于MyISAM表，可以使用myisamchk --analyze。

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。注意通过key_len值我们可以确定MySQL将实际使用一个多部关键字的几个部分。

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。

Extra

> 该列包含MySQL解决查询的详细信息。下面解释了该列可以显示的不同的文本字符串：

1. Distinct

   MySQL发现第1个匹配行后，停止为当前的行组合搜索更多的行。

2. Not exists

   MySQL能够对查询进行LEFT JOIN优化，发现1个匹配LEFT JOIN标准的行后，不再为前面的的行组合在该表内检查更多的行。

   下面是一个可以这样优化的查询类型的例子：
   ```
   SELECT * 从t1 LEFT JOIN t2 ON t1.id=t2.id
   WHERE t2.id IS NULL;
   ```

   假定t2.id定义为NOT NULL。在这种情况下，MySQL使用t1.id的值扫描t1并查找t2中的行。
   如果MySQL在t2中发现一个匹配的行，它知道t2.id绝不会为NULL，并且不再扫描t2内有相同的id值的行。
   换句话说，对于t1的每个行，MySQL只需要在t2中查找一次，无论t2内实际有多少匹配的行。

3. range checked for each record (index map: #)

> MySQL没有发现好的可以使用的索引，但发现如果来自前面的表的列值已知，可能部分索引可以使用。对前面的表的每个行组合，MySQL检查是否可以使用range或index_merge访问方法来索取行。关于适用性标准的描述参见7.2.5节，“范围优化”和7.2.6节，“索引合并优化”，不同的是前面表的所有列值已知并且认为是常量。
>
> 这并不很快，但比执行没有索引的联接要快得多。

4. Using filesort

> MySQL需要额外的一次传递，以找出如何按排序顺序检索行。通过根据联接类型浏览所有行并为所有匹配WHERE子句的行保存排序关键字和行的指针来完成排序。然后关键字被排序，并按排序顺序检索行。

5. Using index

> 从只使用索引树中的信息而不需要进一步搜索读取实际的行来检索表中的列信息。当查询只使用作为单一索引一部分的列时，可以使用该策略。

6. Using temporary

> 为了解决查询，MySQL需要创建一个临时表来容纳结果。典型情况如查询包含可以按不同情况列出列的GROUP BY和ORDER BY子句时。

7. Using where

> WHERE子句用于限制哪一个行匹配下一个表或发送到客户。除非你专门从表中索取或检查所有行，如果Extra值不为Using where并且表联接类型为ALL或index，查询可能会有一些错误。
>
> 如果想要使查询尽可能快，应找出Using filesort 和Using temporary的Extra值。

8. Using sort_union(...), Using union(...), Using intersect(...)

> 这些函数说明如何为index_merge联接类型合并索引扫描。

9. Using index for group-by

> 类似于访问表的Using index方式，Using index for group-by表示MySQL发现了一个索引，可以用来查询GROUP BY或DISTINCT查询的所有列，而不要额外搜索硬盘访问实际的表。并且，按最有效的方式使用索引，以便对于每个组，只读取少量索引条目。
>
> 通过相乘EXPLAIN输出的rows列的所有值，你能得到一个关于一个联接如何的提示。这应该粗略地告诉你MySQL必须检查多少行以执行查询。当你使用max_join_size变量限制查询时，也用这个乘积来确定执行哪个多表SELECT语句。