

MySQL 执行计划解读

胡中泉

纲要

- MySQL 执行计划调用方式
- 执行计划包含的信息
- 执行计划显示内容解读
- MySQL 执行计划的局限
- Q&A

通过执行计划可以了解什么

```
mysql> explain
-> select d1.col2, t2.col1
-> from (select col2, col3
->       from t1
->       where col1 in ('ab','ac')) d1, t2
-> where d1.col2 = t2.col1
-> group by d1.col2, t2.col1
-> order by t2.id;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	286	Using temporary; Using filesort
1	PRIMARY	t2	ref	idx_col1_col2	idx_col1_col2	195	d1.col2	45	Using where; Using index
2	DERIVED	t1	range	idx_col1_col2_col3	idx_col1_col2_col3	13	NULL	285	Using where; Using index

3 rows in set (0.00 sec)

MySQL 执行计划调用方式

- EXPLAIN SELECT

变体:

- 1. EXPLAIN EXTENDED SELECT
将执行计划“反编译”成 SELECT 语句，运行 SHOW WARNINGS
可得到被 MySQL 优化器优化后的查询语句
- 2. EXPLAIN PARTITIONS SELECT
用于分区表的 EXPLAIN

执行计划包含的信息

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
----	-------------	-------	------	---------------	-----	---------	-----	------	-------

1. id :

包含一组数字，表示查询中执行 select 子句或操作表的顺序

e.g.

```
mysql> explain select t2.*
-> from t1, t2, t3
-> where t1.id = t2.id and t1.id = t3.id
-> and t1.other_column = '';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	ref	PRIMARY,idx_t1	idx_t1	92	const	1	Using where
1	SIMPLE	t3	eq_ref	PRIMARY	PRIMARY	4	test.t1.ID	1	Using index
1	SIMPLE	t2	eq_ref	PRIMARY	PRIMARY	4	test.t1.ID	1	

3 rows in set (0.00 sec)

id 相同，执行顺序由上至下

e.g.

```
mysql> explain SELECT t2.*
-> FROM t2
-> WHERE id = (SELECT id
->             FROM t1
->             WHERE id = (SELECT t3.id
->                         FROM t3
->                         WHERE t3.other_column = ''));
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	t2	const	PRIMARY	PRIMARY	4	const	1	
2	SUBQUERY	t1	const	PRIMARY	PRIMARY	4		1	Using index
3	SUBQUERY	t3	ALL	NULL	NULL	NULL	NULL	1	Using where

3 rows in set (0.00 sec)

如果是子查询，id 的序号会递增，id 值越大优先级越高，越先被执行

e.g.

```
mysql> explain select t2.* from (  
-> select t3.id  
-> from t3  
-> where t3.other_column = '') s1, t2  
-> where s1.id = t2.id;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	system	NULL	NULL	NULL	NULL	1	
1	PRIMARY	t2	const	PRIMARY	PRIMARY	4	const	1	
2	DERIVED	t3	ALL	NULL	NULL	NULL	NULL	1	Using where

```
3 rows in set (0.00 sec)
```

id 如果相同，可以认为是一组，从上往下顺序执行；在所有组中， id 值越大，优先级越高，越先执行

2. select_type

表示查询中每个 select 子句的类型（简单 OR 复杂）

id	select_type
1	SIMPLE
2	PRIMARY
3	SUBQUERY
4	DERIVED
5	UNION
6	UNION RESULT

- SIMPLE：查询中不包含子查询或者 UNION
- 查询中若包含任何复杂的子部分，最外层查询则被标记为：PRIMARY
- 在 SELECT 或 WHERE 列表中包含了子查询，该子查询被标记为：SUBQUERY
- 在 FROM 列表中包含的子查询被标记为：DERIVED（衍生）
- 若第二个 SELECT 出现在 UNION 之后，则被标记为 UNION；若 UNION 包含在 FROM 子句的子查询中，外层 SELECT 将被标记为：DERIVED
- 从 UNION 表获取结果的 SELECT 被标记为：UNION RESULT

e.g.

```
mysql> explain select d1.name, (select id from t3) d2
-> from (select id, name from t1 where other_column = '') d1
-> union
-> (select name, id from t2);
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived3>	system	NULL	NULL	NULL	NULL	1	
3	DERIVED	t1	ALL	NULL	NULL	NULL	NULL	1	Using where
2	SUBQUERY	t3	index	NULL	PRIMARY	4	NULL	1	Using index
4	UNION	t2	ALL	NULL	NULL	NULL	NULL	1	
NULL	UNION RESULT	<union1,4>	ALL	NULL	NULL	NULL	NULL	NULL	

5 rows in set (0.01 sec)

第一行：id 列为 1，表示第一个 select，select_type 列的 primary 表示该查询为外层查询，table 列被标记为 <derived3>，表示查询结果来自一个衍生表，其中 3 代表该查询衍生自第三个 select 查询，即 id 为 3 的 select。

第二行：id 为 3，表示该查询的执行次序为 2（4→3），是整个查询中第三个 select 的一部分。因查询包含在 from 中，所以为 derived。

第三行：select 列表中的子查询，select_type 为 subquery，为整个查询中的第二个 select。

第四行：select_type 为 union，说明第四个 select 是 union 里的第二个 select，最先执行。

第五行：代表从 union 的临时表中读取行的阶段，table 列的 <union1,4> 表示用第一个和第四个 select 的结果进行 union 操作。

3. type

表示 MySQL 在表中找到所需行的方式，又称“访问类型”，常见类型如下：

+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
	ALL		index		range		ref		eq_ref		const, system		NULL	
+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+

由左至右，由最差到最好

e.g.

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

a. ALL : Full Table Scan , MySQL 将遍历全表以找到匹配的行

```
mysql> explain select * from t1 where column_without_index = '';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	ALL	NULL	NULL	NULL	NULL	516	Using where

1 row in set (0.00 sec)

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

b. index : Full Index Scan , index 与 ALL 区别为 index 类型只遍历索引

```
mysql> explain select id from t1;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	index	NULL	PRIMARY	4	NULL	516	Using index

1 row in set (0.00 sec)

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

c. range：索引范围扫描，对索引的扫描开始于某一点，返回匹配值域的行，常见于 between、<、> 等的查询

e.g.

```
mysql> explain SELECT * FROM t1 WHERE id between 30 and 60;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	range	PRIMARY	PRIMARY	4	NULL	31	Using where

1 row in set (0.00 sec)

```
mysql> explain select * from t1 where id in (1, 2, 6);
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	range	PRIMARY	PRIMARY	4	NULL	3	Using where

1 row in set (0.00 sec)

TIPS :

range 访问类型的不同形式的索引访问性能差异

```
mysql> explain select id from t1 where col1 in ('ac','ab','aa') and col2 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	range	idx_col1_col2	idx_col1_col2	388	NULL	3	

```
1 row in set (0.00 sec)
```

```
mysql> explain select id from t1 where col1 > 'aa' and col2 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	range	idx_col1_col2	idx_col1_col2	194	NULL	299	

```
1 row in set (0.00 sec)
```

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	------------	--------	---------------	------

d. ref：非唯一性索引扫描，返回匹配某个单独值的所有行。常见于使用非唯一索引即唯一索引的非唯一前缀进行的查找

e.g.

```
mysql> create index idx_col1_col2 on t1(col1,col2);
Query OK, 1000 rows affected (0.15 sec)
Records: 1000 Duplicates: 0 Warnings: 0
```

```
mysql> select count(distinct col1) from t1;
```

count(distinct col1)
7

```
1 row in set (0.00 sec)
```

```
mysql> explain select * from t1 where col1 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	ref	idx_col1_col2	idx_col1_col2	194	const	284	

```
1 row in set (0.00 sec)
```

e.g.

```
mysql> explain select * from t1, t2 WHERE t1.col1 = t2.col1;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t2	ALL	NULL	NULL	NULL	NULL	639	
1	SIMPLE	t1	ref	idx_col1_col2	idx_col1_col2	195	shared.t2.col1	45	

```
2 rows in set (0.00 sec)
```

```
mysql> explain select * from t1, t2 WHERE t1.col1 = t2.col1 AND t1.col2 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t2	ALL	NULL	NULL	NULL	NULL	639	
1	SIMPLE	t1	ref	idx_col1_col2	idx_col1_col2	390	shared.t2.col1,const	45	

```
2 rows in set (0.00 sec)
```

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

e. eq_ref：唯一性索引扫描，对于每个索引键，表中只有一条记录与之匹配。常见于主键或唯一索引扫描

e.g.

```
mysql> explain select * from t1, t2 where t1.id = t2.id;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t2	ALL	PRIMARY	NULL	NULL	NULL	639	
1	SIMPLE	t1	eq_ref	PRIMARY	PRIMARY	4	shared.t2.ID	1	

```
2 rows in set (0.00 sec)
```


ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

- f. const、system：当 MySQL 对查询某部分进行优化，并转换为一个常量时，使用这些类型访问。如将主键置于 where 列表中，MySQL 就能将该查询转换为一个常量
- e.g.

```
mysql> explain select * from (select * from t1 where id = 1) d1;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	system	NULL	NULL	NULL	NULL	1	
2	DERIVED	t1	const	PRIMARY	PRIMARY	4		1	

```
2 rows in set (0.00 sec)
```

TIPS：system 是 const 类型的特例，当查询的表只有一行的情况下，使用 system

ALL	index	range	ref	eq_ref	const, system	NULL
-----	-------	-------	-----	--------	---------------	------

g. NULL : MySQL 在优化过程中分解语句, 执行时甚至不用访问表或索引

e.g.

```
mysql> explain extended select * from t1 where id = (select min(id) from t2);
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	t1	const	PRIMARY	PRIMARY	4	const	1	100.00	
2	SUBQUERY	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

2 rows in set, 1 warning (0.00 sec)

```
mysql> show warnings;
```

Level	Code	Message
Note	1003	select '1' AS `ID`,`hu` AS `col1`,`dba` AS `col2` from `shared`.`t1` where 1

1 row in set (0.00 sec)

4. possible_keys

指出 MySQL 能使用哪个索引在表中找到行，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询使用

5. key

显示 MySQL 在查询中实际使用的索引，若没有使用索引，显示为 NULL

TIPS：查询中若使用了覆盖索引，则该索引仅出现在 key 列表中

```
mysql> explain select col1, col2 from t1;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	index	NULL	idx_col1_col2	390	NULL	682	Using index

1 row in set (0.00 sec)

6. key_len

表示索引中使用的字节数，可通过该列计算查询中使用的索引的长度

```
mysql> desc t1;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
col1	char(4)	YES	MUL	NULL	
col2	char(4)	YES		NULL	

```
3 rows in set (0.00 sec)
```

```
mysql> explain select * from t1 where col1 = 'ab';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	ref	idx_col1_col2	idx_col1_col2	13	const	143	

```
1 row in set (0.00 sec)
```

```
mysql> explain select * from t1 where col1 = 'ab' and col2 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t1	ref	idx_col1_col2	idx_col1_col2	26	const,const	1	

```
1 row in set (0.01 sec)
```

TIPS : key_len 显示的值为索引字段的最大可能长度，并非实际使用长度，即 key_len 是根据表定义计算而得，不是通过表内检索出的

7. ref

表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

```
mysql> explain select * from t1, t2 where t1.col1 = t2.col1 and t1.col2 = 'ac';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | ... | table | type | possible_keys | key          | key_len | ref          | rows |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | ... | t2    | ALL  | NULL         | NULL        | NULL    | NULL        | 640 |
| 1 | ... | t1    | ref  | idx_col1_col2 | idx_col1_col2 | 26      | shared.t2.col1,const | 82 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

本例中，由 key_len 可知 t1 表的 idx_col1_col2 被充分使用， col1 匹配 t2 表的 col1， col2 匹配了一个常量，即 'ac'

8. rows

表示 MySQL 根据表统计信息及索引选用情况，估算的找到所需的记录所需要读取的行数

```
mysql> explain select * from t1, t2 where t1.id = t2.id and t2.col1 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t2	ALL	PRIMARY	NULL	NULL	NULL	640	Using where
1	SIMPLE	t1	eq_ref	PRIMARY	PRIMARY	4	shared.t2.ID	1	

```
2 rows in set (0.00 sec)
```

```
mysql> create index idx_col1_col2 on t2(col1,col2);
```

```
Query OK, 1001 rows affected (0.17 sec)
```

```
Records: 1001 Duplicates: 0 Warnings: 0
```

```
mysql> explain select * from t1, t2 where t1.id = t2.id and t2.col1 = 'ac';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows
1	SIMPLE	t2	ref	PRIMARY,idx_col1_col2	idx_col1_col2	195	const	142
1	SIMPLE	t1	eq_ref	PRIMARY	PRIMARY	4	shared.t2.ID	1

```
2 rows in set (0.00 sec)
```

9. Extra

包含不适合在其他列中显示但十分重要的额外信息

a. Using index

该值表示相应的 select 操作中使用了覆盖索引 (Covering Index)

```
mysql> explain select col2 from t1 where col1 = 'ab';
+----+-----+-----+-----+-----+-----+-----+-----+
| id | ... | possible_keys | key           | key_len | ref  | rows | Extra                               |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1  | ... | idx_col1_col2 | idx_col1_col2 | 13      | const | 143 | Using where; Using index          |
+----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```


TIPS：覆盖索引（Covering Index）

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

包含所有满足查询需要的数据的索引称为 **覆盖索引**（Covering Index）

注意：

如果要使用覆盖索引，一定要注意 select 列表中只取出需要的列，不可 select *，因为如果将所有字段一起做索引会导致索引文件过大，查询性能下降

b. Using where

表示 MySQL 服务器在存储引擎受到记录后进行“后过滤”（ Post-filter ），

如果查询未能使用索引， Using where 的作用只是提醒我们 MySQL 将用 where 子句来过滤结果集

```
mysql> explain extended select t1.col2 from t1, t2 where t1.col1 = 'ab' and t1.id= t2.id ;
```

id	key	key_len	ref	rows	filtered	Extra
1	idx_col1_col2	13	const	143	100.00	Using where; Using index
1	PRIMARY	4	shared.t1.ID	1	100.00	Using index

```
2 rows in set, 1 warning (0.00 sec)
```

c. Using temporary

表示 MySQL 需要使用临时表来存储结果集，常见于排序和分组查询

```
mysql> explain select col1 from t1 where col1 in ('ac','ab','aa') group by col2\G
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: t1
         type: range
possible_keys: idx_col1_col2
         key: idx_col1_col2
        key_len: 13
         ref: NULL
         rows: 569
    Extra: Using where; Using index; Using temporary; Using filesort
1 row in set (0.00 sec)
```

```
mysql> explain select col1 from t1 where col1 in ('ac', 'ab') group by col1, col2\G
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: t1
         type: range
possible_keys: idx_col1_col2_col3
         key: idx_col1_col2_col3
        key_len: 26
         ref: NULL
         rows: 4
    Extra: Using where; Using index for group-by
1 row in set (0.00 sec)
```

d. Using filesort

MySQL 中无法利用索引完成的排序操作称为“文件排序”

```
mysql> explain select col1 from t1 where col1 = 'ac' order by col3\G
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: t1
         type: ref
possible_keys: idx_col1_col2_col3
         key: idx_col1_col2_col3
        key_len: 13
         ref: const
        rows: 142
      Extra: Using where; Using index; Using filesort
1 row in set (0.00 sec)
```

```
mysql> explain select col1 from t1 where col1 = 'ac' order by col2, col3\G
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: t1
         type: ref
possible_keys: idx_col1_col2_col3
         key: idx_col1_col2_col3
        key_len: 13
         ref: const
        rows: 142
      Extra: Using where; Using index
1 row in set (0.00 sec)
```

```
mysql> explain
-> select d1.col2, t2.col1
-> from (select col2, col3
->       from t1
->       where col1 in ('ab','ac')) d1, t2
-> where d1.col2 = t2.col1
-> group by d1.col2, t2.col1
-> order by t2.id;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	286	Using temporary; Using filesort
1	PRIMARY	t2	ref	idx_col1_col2	idx_col1_col2	195	d1.col2	45	Using where; Using index
2	DERIVED	t1	range	idx_col1_col2_col3	idx_col1_col2_col3	13	NULL	285	Using where; Using index

3 rows in set (0.00 sec)

```
mysql> desc t1;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
col1	char(4)	YES	MUL	NULL	
col2	char(4)	YES		NULL	
col3	char(4)	YES		NULL	

4 rows in set (0.00 sec)

```
mysql> desc t2;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
col1	varchar(64)	YES	MUL	NULL	
col2	varchar(64)	YES		NULL	

3 rows in set (0.00 sec)

```
mysql> select count(*) from t1 where col1 = 'ac';
```

count(*)
143

1 row in set (0.00 sec)

```
mysql> select count(*) from t1 where col1 = 'ab';
```

count(*)
143

1 row in set (0.01 sec)

MySQL 执行计划的局限

- EXPLAIN 不会告诉你关于触发器、存储过程的信息或用户自定义函数对查询的影响情况
- EXPLAIN 不考虑各种 Cache
- EXPLAIN 不能显示 MySQL 在执行查询时所作的优化工作
- 部分统计信息是估算的，并非精确值
- EXPLAIN 只能解释 SELECT 操作，其他操作要重写为 SELECT 后查看执行计划

.....

Q&A

谢谢