执行计划查看

mysql执行计划

在企业的应用场景中，为了知道优化SQL语句的执行，需要查看SQL语句的具体执行过程，以加快SQL语句的执行效率。

可以使用explain+SQL语句来模拟优化器执行SQL查询语句，从而知道mysql是如何处理sql语句的。

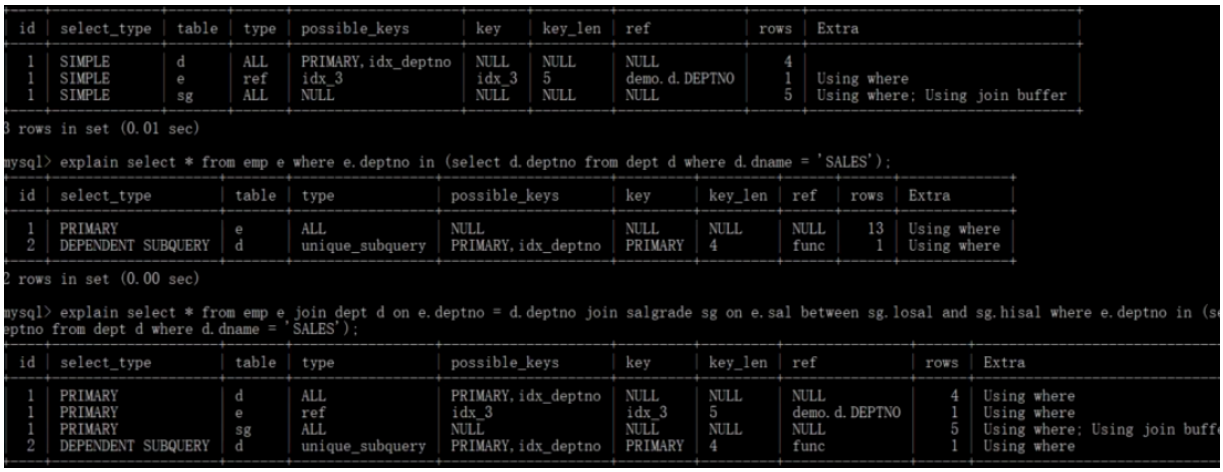
官网地址： https://dev.mysql.com/doc/refman/5.5/en/explain-output.html

1、执行计划中包含的信息

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查询的序列号，包含一组数字，表示查询中执行select子句或者操作表的顺序



```
+----+-------------+-------+------+-----------------+-------+---------+-----------------+------+----------------------------------------------+
| id | select_type | table | type | possible_keys   | key   | key_len | ref             | rows | Extra                                        |
+----+-------------+-------+------+-----------------+-------+---------+-----------------+------+----------------------------------------------+
|  1 | SIMPLE      | d     | ALL  | PRIMARY,idx_deptno | NULL  | NULL    | NULL            |    4 |                                              |
|  1 | SIMPLE      | e     | ref  | idx_3           | idx_3 | 5       | demo.d.DEPTNO   |    1 | Using where                                  |
|  1 | SIMPLE      | sg    | ALL  | NULL            | NULL  | NULL    | NULL            |    5 | Using where; Using join buffer               |
+----+-------------+-------+------+-----------------+-------+---------+-----------------+------+----------------------------------------------+
3 rows in set (0.01 sec)

mysql> explain select * from emp e where e.deptno in (select d.deptno from dept d where d.dname = 'SALES');
+----+--------------------+-------+--------------+-----------------+---------+---------+------+------+-------------+
| id | select_type        | table | type         | possible_keys   | key     | key_len | ref  | rows | Extra       |
+----+--------------------+-------+--------------+-----------------+---------+---------+------+------+-------------+
|  1 | PRIMARY            | e     | ALL          | NULL            | NULL    | NULL    | NULL |   13 | Using where |
|  2 | DEPENDENT SUBQUERY | d     | unique_subquery | PRIMARY,idx_deptno | PRIMARY | 4       | func |    1 | Using where |
+----+--------------------+-------+--------------+-----------------+---------+---------+------+------+-------------+
2 rows in set (0.00 sec)

mysql> explain select * from emp e join dept d on e.deptno = d.deptno join salgrade sg on e.sal between sg.losal and sg.hisal where e.deptno in (s
eptno from dept d where d.dname = 'SALES');
+----+--------------------+-------+--------------+-----------------+---------+---------+---------------+------+----------------------------------+
| id | select_type        | table | type         | possible_keys   | key     | key_len | ref           | rows | Extra                            |
+----+--------------------+-------+--------------+-----------------+---------+---------+---------------+------+----------------------------------+
|  1 | PRIMARY            | d     | ALL          | PRIMARY,idx_deptno | NULL    | NULL    | NULL          |    4 | Using where                      |
|  1 | PRIMARY            | e     | ref          | idx_3           | idx_3   | 5       | demo.d.DEPTNO |    1 | Using where                      |
|  1 | PRIMARY            | sg    | ALL          | NULL            | NULL    | NULL    | NULL          |    5 | Using where; Using join buff     |
|  2 | DEPENDENT SUBQUERY | d     | unique_subquery | PRIMARY,idx_deptno | PRIMARY | 4       | func          |    1 | Using where                      |
+----+--------------------+-------+--------------+-----------------+---------+---------+---------------+------+----------------------------------+
```

id号分为三种情况：

1、如果id相同，那么执行顺序从上到下

explain select * from emp e join dept d on e.deptno = d.deptno join salgrade sg on e.sal between sg.losal and sg.hisal;

2、如果id不同，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

explain select * from emp e where e.deptno in (select d.deptno from dept d where d.dname = 'SALES');

3、id相同和不同的，同时存在：相同的可以认为是一组，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

explain select * from emp e join dept d on e.deptno = d.deptno join salgrade sg on e.sal between sg.losal and sg.hisal where e.deptno in (select d.deptno from dept d where d.dname = 'SALES');

select_type

主要用来分辨查询的类型，是普通查询还是联合查询还是子查询

select_type Value	Meaning
SIMPLE	Simple SELECT (not using UNION or subqueries)
PRIMARY	Outermost SELECT
UNION	Second or later SELECT statement in a UNION
DEPENDENT UNION	Second or later SELECT statement in a UNION, dependent on outer query
UNION RESULT	Result of a UNION.
SUBQUERY	First SELECT in subquery
DEPENDENT SUBQUERY	First SELECT in subquery, dependent on outer query
DERIVED	Derived table
UNCACHEABLE SUBQUERY	A subquery for which the result cannot be cached and must be re-evaluated for each row of the outer query
UNCACHEABLE UNION	The second or later select in a UNION that belongs to an uncacheable subquery (see UNCACHEABLE SUBQUERY)

```
--sample:简单的查询，不包含子查询和union
explain select * from emp;

--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
explain select staname,ename supname from (select ename staname,mgr from emp) t join emp on t.mgr=emp.empno ;

--union:若第二个select出现在union之后，则被标记为union
explain select * from emp where deptno = 10 union select * from emp where sal >2000;

--dependent union:跟union类似，此处的depentent表示union或union all联合而成的结果会受外部表影响
explain select * from emp e where e.empno  in ( select empno from emp where deptno = 10 union select empno from emp where sal >2000)

--union result:从union表获取结果的select
explain select * from emp where deptno = 10 union select * from emp where sal >2000;

--subquery:在select或者where列表中包含子查询
explain select * from emp where sal > (select avg(sal) from emp) ;

--dependent subquery:subquery的子查询要受到外部表查询的影响
explain select * from emp e where e.deptno in (select distinct deptno from dept);

--DERIVED: from子句中出现的子查询，也叫做派生类，
explain select staname,ename supname from (select ename staname,mgr from emp) t join emp on t.mgr=emp.empno ;

--UNCACHEABLE SUBQUERY：表示使用子查询的结果不能被缓存
 explain select * from emp where empno = (select empno from emp where deptno=@@sort_buffer_size);

--uncacheable union:表示union的查询结果不能被缓存：sql语句未验证
```

table

对应行正在访问哪一个表，表名或者别名，可能是临时表或者union合并结果集　1、如果是具体的表名，则表明从实际的物理表中获取数据，当然也可以是表的别名

2、表名是derivedN的形式，表示使用了id为N的查询产生的衍生表

3、当有union result的时候，表名是union n1,n2等的形式，n1,n2表示参与union的id

type——优化的时候，是一个很重要的衡量标准

type显示的是访问类型，访问类型表示我是以何种方式去访问我们的数据，最容易想的是全表扫描，直接暴力的遍历一张表去寻找需要的数据，效率非常低下，访问的类型有很多，效率从最好到最坏依次是：

system > const > eq_ref > ref > fulltext > ref_or_null > index_merge > unique_subquery > index_subquery > range > index > ALL

————— 记住 蓝色的优先级基本就够了

一般情况下，得保证查询至少达到range级别，最好能达到ref

```
--all:全表扫描，一般情况下出现这样的sql语句而且数据量比较大的话那么就需要进行优化。
explain select * from emp;

--index：全索引扫描这个比all的效率要好，主要有两种情况，一种是当前的查询时覆盖索引，即我们需要的数据在索引中就可以索取，或者是使用了索引进行排序，这样就避免数据的
explain  select empno from emp;

--range：表示利用索引查询的时候限制了范围，在指定范围内进行查询，这样避免了index的全索引扫描，适用的操作符： =, <>, >, >=, <, <=, IS NULL, BETWEEN, LIKE
explain select * from emp where empno between 7000 and 7500;

--index_subquery：利用索引来关联子查询，不再扫描全表
explain select * from emp where emp.job in (select job from t_job);

--unique_subquery:该连接类型类似与index_subquery,使用的是唯一索引
 explain select * from emp e where e.deptno in (select distinct deptno from dept);

--index_merge：在查询过程中需要多个索引组合使用，没有模拟出来

--ref_or_null：对于某个字段即需要关联条件，也需要null值的情况下，查询优化器会选择这种访问方式
explain select * from emp e where  e.mgr is null or e.mgr=7369;

--ref：使用了非唯一性索引进行数据的查找
 create index idx_3 on emp(deptno);
 explain select * from emp e,dept d where e.deptno =d.deptno;

--eq_ref  ：使用唯一性索引进行数据查找
explain select * from emp,emp2 where emp.empno = emp2.empno;

--const：这个表至多有一个匹配行，
explain select * from emp where empno = 7369;

--system：表只有一行记录（等于系统表），这是const类型的特例，平时不会出现
```

possible_keys

显示可能应用在这张表中的索引，一个或多个，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询实际使用

```
explain select * from emp,dept where emp.deptno = dept.deptno and emp.deptno = 10;
```

key

实际使用的索引，如果为null，则没有使用索引，查询中若使用了覆盖索引，则该索引和查询的select字段重叠。

```
explain select * from emp,dept where emp.deptno = dept.deptno and emp.deptno = 10;
```

key_len

表示索引中使用的字节数，可以通过key_len计算查询中使用的索引长度，在不损失精度的情况下长度越短越好。越短，说明占用空间越小，这样IO的代价也就越低

```
explain select * from emp,dept where emp.deptno = dept.deptno and emp.deptno = 10;
```

ref

显示索引的哪一列被使用了，如果可能的话，是一个常数

```
explain select * from emp,dept where emp.deptno = dept.deptno and emp.deptno = 10;
```

rows

根据表的统计信息及索引使用情况，大致估算出找出所需记录需要读取的行数，此参数很重要，直接反应的sql找了多少数据，在完成目的的情况下越少越好

```
explain select * from emp;
```

extra

包含额外的信息。

```
--using filesort:说明mysql无法利用索引进行排序，只能利用排序算法进行排序，会消耗额外的位置
explain select * from emp order by sal;

--using temporary:建立临时表来保存中间结果，查询完成之后把临时表删除
explain select ename,count(*) from emp where deptno = 10 group by ename;

--using index:这个表示当前的查询时覆盖索引的，直接从索引中读取数据，而不用访问数据表。如果同时出现using where 表名索引被用来执行索引键值的查找，如果没有，表面
explain select deptno,count(*) from emp group by deptno limit 10;

--using where:使用where进行条件过滤
explain select * from t_user where id = 1;

--using join buffer:使用连接缓存，情况没有模拟出来

--impossible where：where语句的结果总是false
explain select * from emp where empno = 7469;
```