<https://mp.weixin.qq.com/s/zbYV2YBngt-7_XqhBm6ZEw>

# SQL 优化：学会看 Explain执行计划

## Explain有什么用

当Explain 与 SQL语句一起使用时，MySQL 会显示来自优化器关于SQL执行的信息。也就是说，MySQL解释了它将如何处理该语句，包括如何连接表以及什么顺序连接表等。

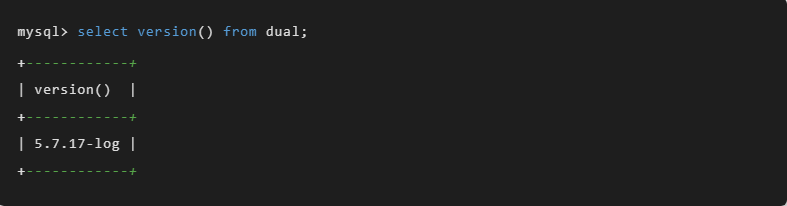
* 表的加载顺序
* sql 的查询类型
* 可能用到哪些索引，哪些索引又被实际使用
* 表与表之间的引用关系
* 一个表中有多少行被优化器查询 .....

## Explain有哪些信息

Explain 执行计划包含字段信息如下：分别是 id、select\_type、table、partitions、type、possible\_keys、key、key\_len、ref、rows、filtered、Extra 12个字段。



下边我们会结合具体的SQL示例，详细的解读每个字段以及每个字段中不同参数的含义，以下所有示例数据库版本为 MySQL.5.7.17。



我们创建三张表 one、two、three，表之间的关系 one.two\_id = two.two\_id AND two.three\_id = three.three\_id。

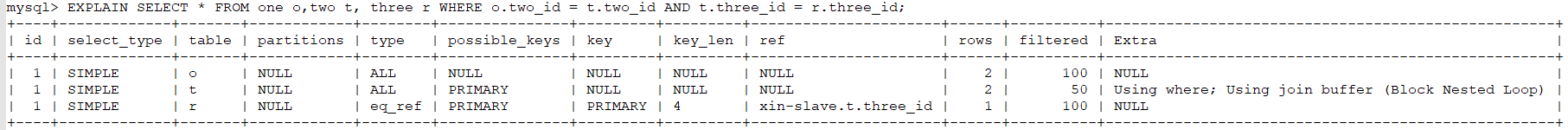
## Explain执行计划详解

### 一、id

id： ：表示查询中执行select子句或者操作表的顺序，**id的值越大，代表优先级越高，越先执行**。 id大致会出现 3种情况：

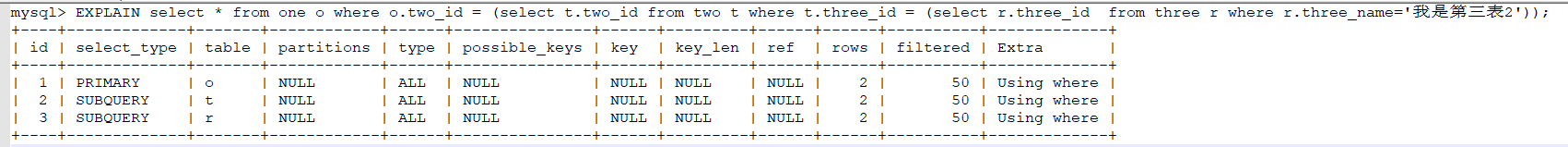
##### 1、id相同

看到三条记录的id都相同，可以理解成这三个表为一组，具有同样的优先级，执行顺序由上而下，具体顺序由优化器决定。



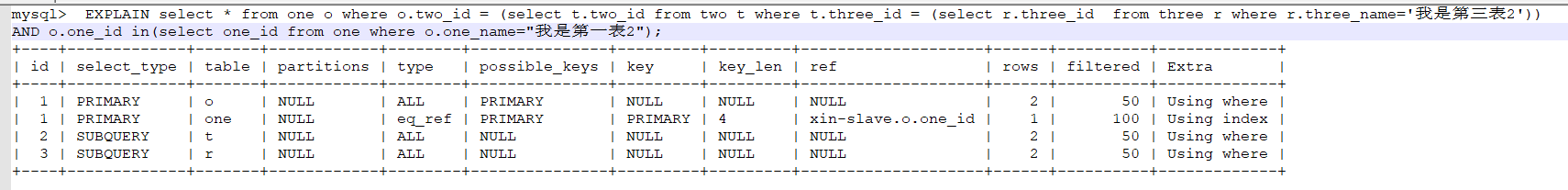
##### 2、id不同

如果我们的 SQL 中存在子查询，那么 id的序号会递增，id值越大优先级越高，越先被执行 。当三个表依次嵌套，发现最里层的子查询 id最大，最先执行。



##### 3、以上两种同时存在

将上边的 SQL 稍微修改一下，增加一个子查询，发现 id的以上两种同时存在。相同id划分为一组，这样就有三个组，同组的从上往下顺序执行，不同组 id值越大，优先级越高，越先执行。



### 二、select\_type

select\_type：表示 select 查询的类型，主要是用于区分各种复杂的查询，例如：普通查询、联合查询、子查询等。

##### 1、SIMPLE

SIMPLE：表示最简单的 select 查询语句，也就是在查询中不包含子查询或者 union交并差集等操作。

##### 2、PRIMARY

PRIMARY：当查询语句中包含任何复杂的子部分，最外层查询则被标记为PRIMARY。

##### 3、SUBQUERY

SUBQUERY：当 select 或 where 列表中包含了子查询，该子查询被标记为：SUBQUERY 。

##### 4、DERIVED

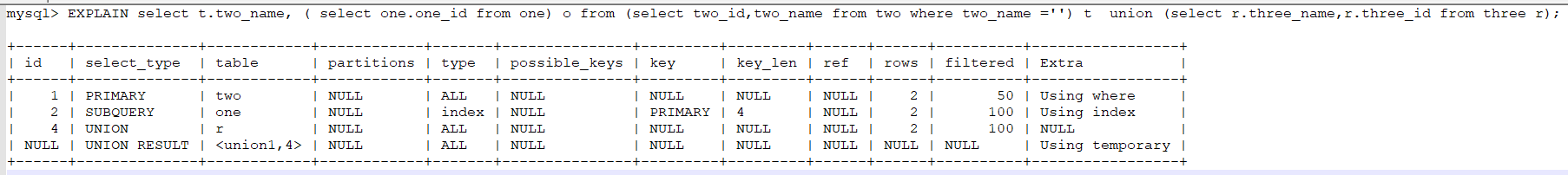
DERIVED：表示包含在from子句中的子查询的select，在我们的 from 列表中包含的子查询会被标记为derived 。

##### 5、UNION

UNION：如果union后边又出现的select 语句，则会被标记为union；若 union 包含在 from 子句的子查询中，外层 select 将被标记为 derived。

##### 6、UNION RESULT

UNION RESULT：代表从union的临时表中读取数据，而table列的<union1,4>表示用第一个和第四个select的结果进行union操作。

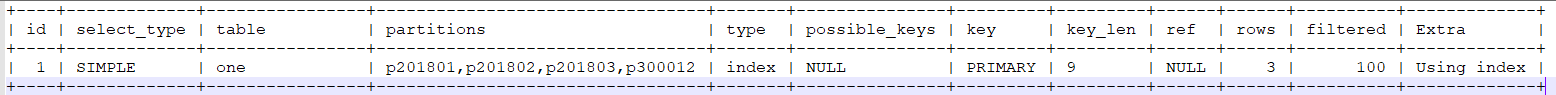


### 三、table

查询的表名，并不一定是真实存在的表，有别名显示别名，也可能为临时表，例如上边的DERIVED、 <union1,4>等。

### 四、partitions

查询时匹配到的分区信息，对于非分区表值为NULL，当查询的是分区表时，partitions显示分区表命中的分区情况。



### 五、type

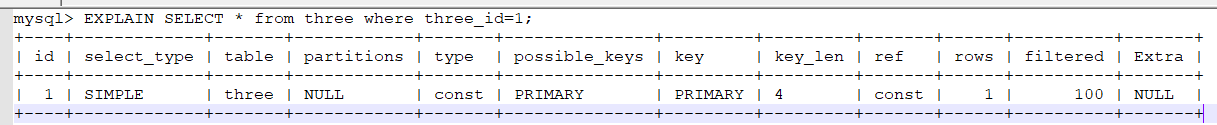
type：查询使用了何种类型，它在 SQL优化中是一个非常重要的指标，以下性能从好到坏依次是：system  > const > eq\_ref > ref  > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL

##### 1、system

system： 当表仅有一行记录时(系统表)，数据量很少，往往不需要进行磁盘IO，速度非常快。

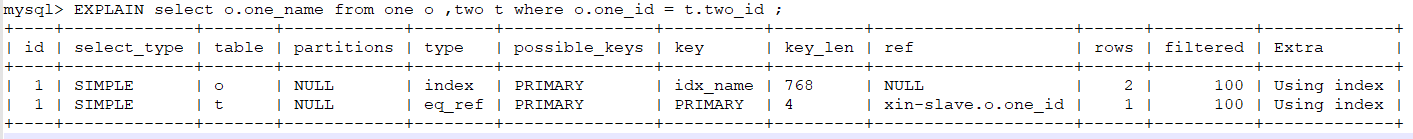
##### 2、const

const：表示查询时命中 primary key 主键或者 unique 唯一索引，或者被连接的部分是一个常量(const)值。这类扫描效率极高，返回数据量少，速度非常快。



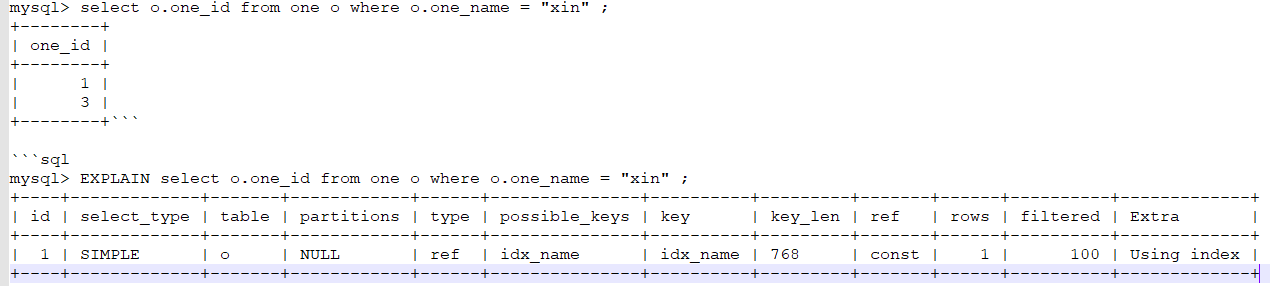
##### 3、eq\_ref

eq\_ref：查询时命中主键primary key 或者 unique key索引， type 就是 eq\_ref。



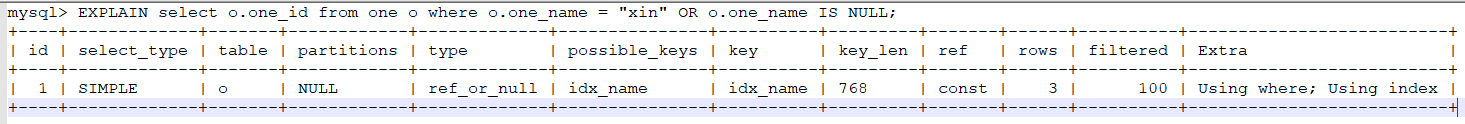
##### 4、ref

ref：区别于eq\_ref ，ref表示使用非唯一性索引，会找到很多个符合条件的行。



##### 5、ref\_or\_null

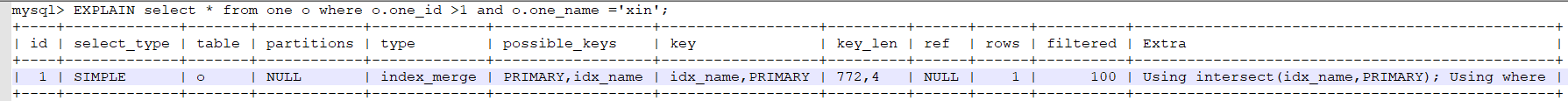
ref\_or\_null：这种连接类型类似于 ref，区别在于 MySQL会额外搜索包含NULL值的行。



##### 6、index\_merge

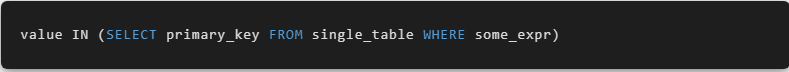
index\_merge：使用了索引合并优化方法，查询使用了两个以上的索引。

下边示例中同时使用到主键one\_id 和 字段one\_name的idx\_name 索引 。



##### 7、unique\_subquery

unique\_subquery：替换下面的 IN子查询，子查询返回不重复的集合。



##### 8、index\_subquery

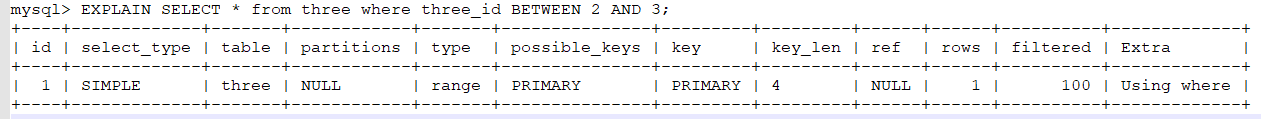
index\_subquery：区别于unique\_subquery，用于非唯一索引，可以返回重复值。



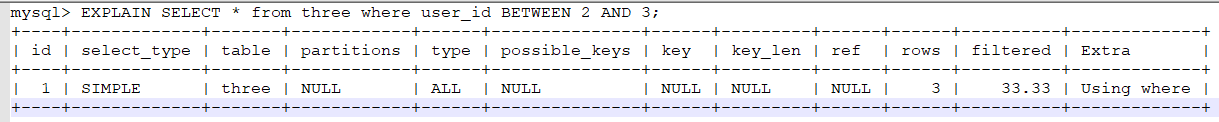
##### 9、range

range：使用索引选择行，仅检索给定范围内的行。简单点说就是针对一个有索引的字段，给定范围检索数据。在where语句中使用 bettween...and、<、>、<=、in 等条件查询 type 都是 range。

举个栗子：three表中three\_id为唯一主键，user\_id普通字段未建索引。



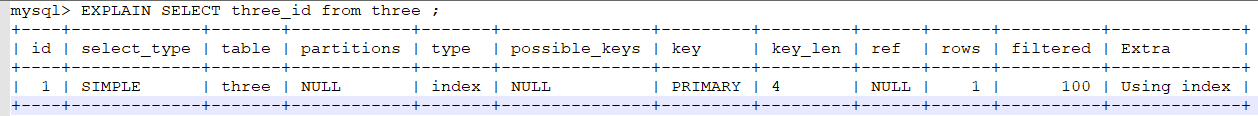
从结果中看到只有对设置了索引的字段，做范围检索 type 才是 range。



##### 10、index

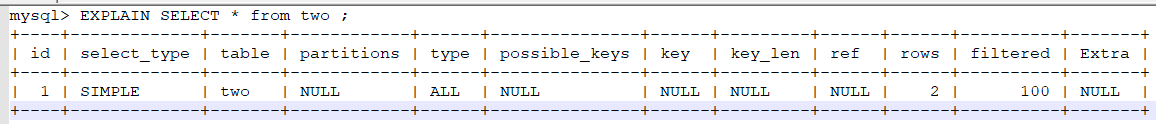
index：**Index 与ALL 其实都是读全表，区别在于index是遍历索引树读取，而ALL是从硬盘中读取。**

下边示例：three\_id 为主键，不带 where 条件全表查询 ，type结果为index 。



##### 11、ALL

ALL：将遍历全表以找到匹配的行，性能最差。



### 六、possible\_keys

possible\_keys：表示在MySQL中通过哪些索引，能让我们在表中找到想要的记录，**一旦查询涉及到的某个字段上存在索引，则索引将被列出**，**但这个索引并不定一会是最终查询数据时所被用到的索引**。具体请参考上边的例子。

### 七、key

key：区别于possible\_keys，key是查询中实际使用到的索引，若没有使用索引，显示为NULL。具体请参考上边的例子。

当 type 为 index\_merge 时，可能会显示多个索引。

### 八、key\_len

key\_len：表示查询用到的索引长度（字节数），原则上长度越短越好 。

* 单列索引，那么需要将整个索引长度算进去；
* 多列索引，不是所有列都能用到，需要计算查询中实际用到的列。

注意：key\_len只计算where条件中用到的索引长度，而排序和分组即便是用到了索引，也不会计算到key\_len中。

### 九、ref

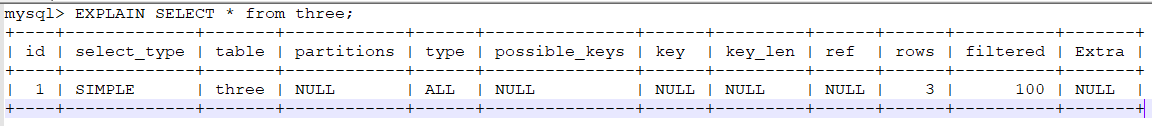
ref：常见的有：const，func，null，字段名。

* 当使用常量等值查询，显示const，
* 当关联查询时，会显示相应关联表的关联字段
* 如果查询条件使用了表达式、函数，或者条件列发生内部隐式转换，可能显示为func
* 其他情况null

### 十、rows

rows：以表的统计信息和索引使用情况，估算要找到我们所需的记录，需要读取的行数。

这是评估SQL 性能的一个比较重要的数据，mysql需要扫描的行数，很直观的显示 SQL 性能的好坏，一般情况下 rows 值越小越好。



### 十一、filtered

filtered 这个是一个百分比的值，表里符合条件的记录数的百分比。简单点说，这个字段表示存储引擎返回的数据在经过过滤后，剩下满足条件的记录数量的比例。

在MySQL.5.7版本以前想要显示filtered需要使用explain extended命令。MySQL.5.7后，默认explain直接显示partitions和filtered的信息。

### 十二、Extra

Extra ：不适合在其他列中显示的信息，Explain 中的很多额外的信息会在 Extra 字段显示。

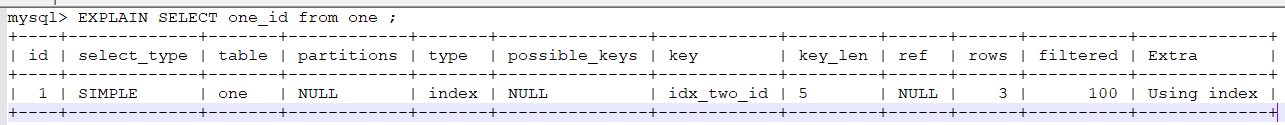
##### 1、Using index

Using index：我们在相应的 select 操作中使用了覆盖索引，通俗一点讲就是查询的列被索引覆盖，使用到覆盖索引查询速度会非常快，SQl优化中理想的状态。

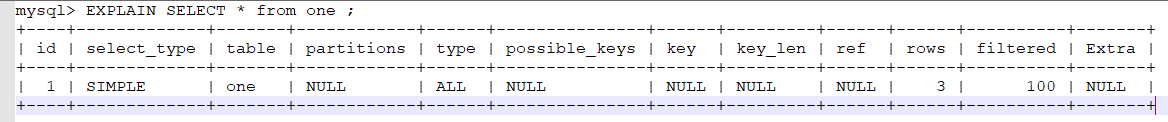
什么又是覆盖索引?

一条 SQL只需要通过索引就可以返回，我们所需要查询的数据（一个或几个字段），而不必通过二级索引，查到主键之后再通过主键查询整行数据（select \* ）。

one\_id表为主键



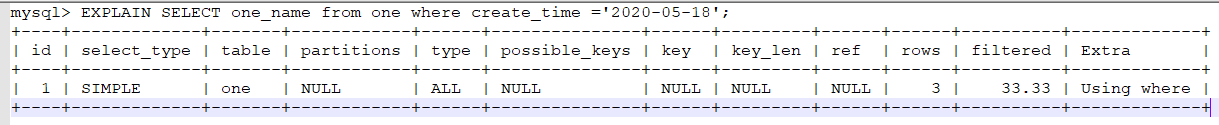
**注意**：想要使用到覆盖索引，我们在 select 时只取出需要的字段，不可select \*，而且该字段建了索引。



##### 2、Using where

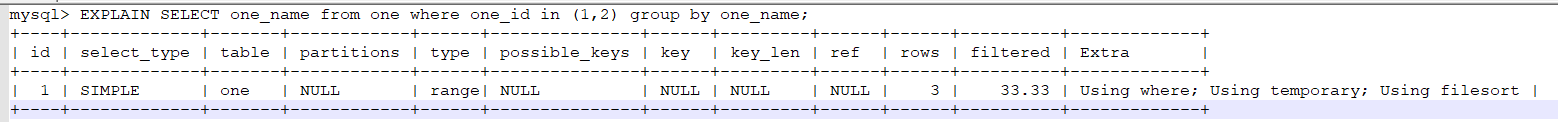
Using where：查询时未找到可用的索引，进而通过where条件过滤获取所需数据，但要注意的是并不是所有带where语句的查询都会显示Using where。

下边示例create\_time 并未用到索引，type 为 ALL，即MySQL通过全表扫描后再按where条件筛选数据。



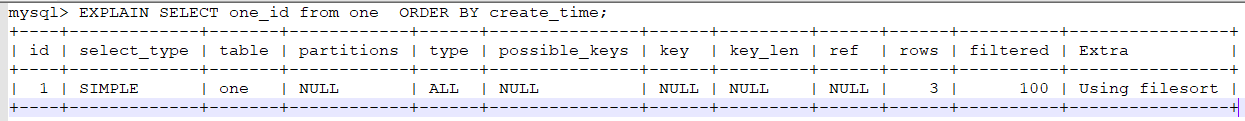
##### 3、Using temporary

Using temporary：表示查询后结果需要使用临时表来存储，一般在排序或者分组查询时用到。

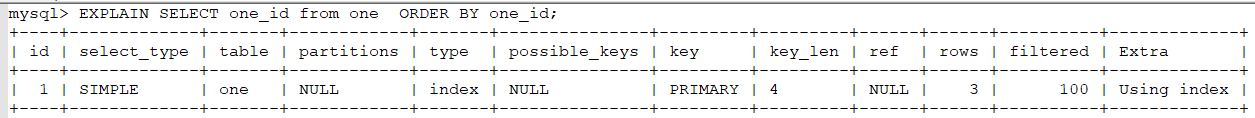


##### 4、Using filesort

Using filesort：表示无法利用索引完成的排序操作，也就是ORDER BY的字段没有索引，通常这样的SQL都是需要优化的。



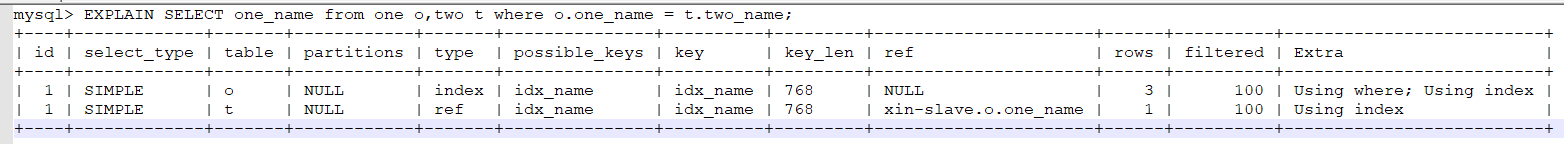
如果ORDER BY字段有索引就会用到覆盖索引，相比执行速度快很多。



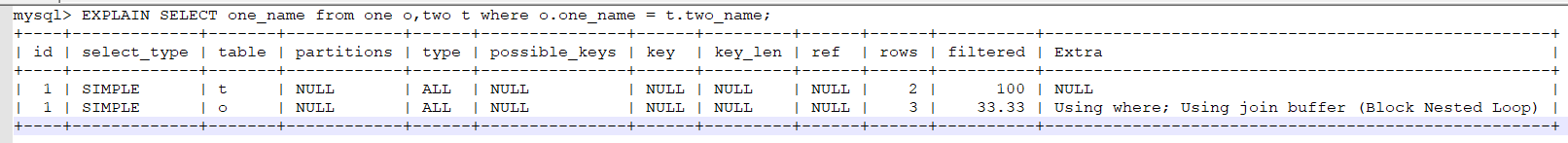
##### 5、Using join buffer

Using join buffer：在我们联表查询的时候，如果表的连接条件没有用到索引，需要有一个连接缓冲区来存储中间结果。

先看一下有索引的情况：连接条件 one\_name 、two\_name 都用到索引。

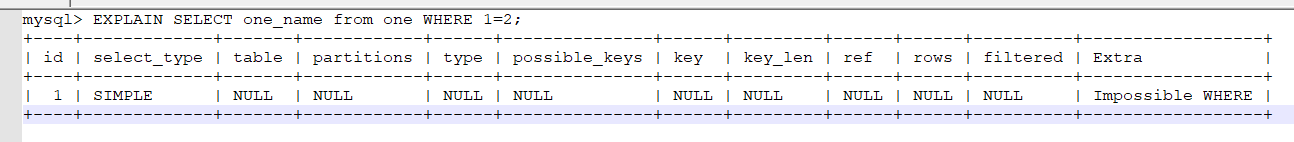


接下来删掉 连接条件 one\_name 、two\_name 的字段索引。发现Extra 列变成 Using join buffer，type均为全表扫描，这也是SQL优化中需要注意的地方。



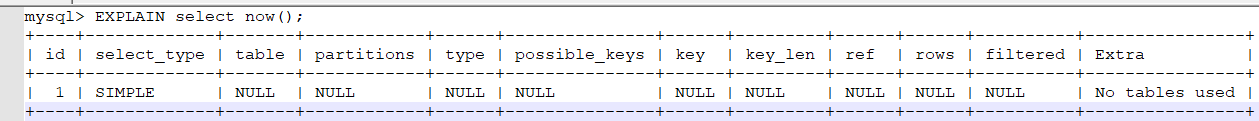
##### 6、Impossible where

Impossible where：表示在我们用不太正确的where语句，导致没有符合条件的行。



##### 7、No tables used

No tables used：我们的查询语句中没有FROM子句，或者有 FROM DUAL子句。



Extra列的信息非常非常多，这里就不再一一列举了，详见 MySQL官方文档 ：<https://dev.mysql.com/doc/refman/5.7/en/explain-output.html#jointype_index_merge>



## 总结

上边只是简单介绍了下 Explain 执行计划各个列的含义，了解它不仅仅是要应付面试，在实际开发中也经常会用到。比如对慢SQL进行分析，如果连执行计划结果都不会看，那还谈什么SQL优化呢？