[MYSQL explain详解](http://blog.csdn.net/zhuxineli/article/details/14455029)

标签： [WHERE子句用于限制哪一个行匹配下一个](http://www.csdn.net/tag/WHERE%e5%ad%90%e5%8f%a5%e7%94%a8%e4%ba%8e%e9%99%90%e5%88%b6%e5%93%aa%e4%b8%80%e4%b8%aa%e8%a1%8c%e5%8c%b9%e9%85%8d%e4%b8%8b%e4%b8%80%e4%b8%aa)[如果Extra值不为Using wher](http://www.csdn.net/tag/%e5%a6%82%e6%9e%9cExtra%e5%80%bc%e4%b8%8d%e4%b8%baUsing%20wher)[查询可能会有一些错误 如果想](http://www.csdn.net/tag/%e6%9f%a5%e8%af%a2%e5%8f%af%e8%83%bd%e4%bc%9a%e6%9c%89%e4%b8%80%e4%ba%9b%e9%94%99%e8%af%af%20%20%e5%a6%82%e6%9e%9c%e6%83%b3)

2013-11-24 17:55 103228人阅读 [评论](http://blog.csdn.net/zhuxineli/article/details/14455029#comments)(10) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/zhuxineli/article/details/14455029#report)

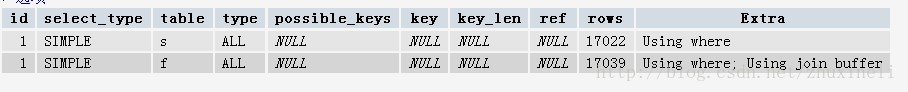
http://static.blog.csdn.net/images/category_icon.jpg 分类：

mysql（13） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

explain显示了[**MySQL**](http://lib.csdn.net/base/mysql)如何使用索引来处理select语句以及连接表。可以帮助选择更好的索引和写出更优化的查询语句。

先解析一条sql语句，看出现什么内容

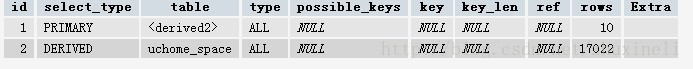
EXPLAINSELECTs.uid,s.username,s.name,f.email,f.mobile,f.phone,f.postalcode,f.address  
FROM uchome\_space ASs,uchome\_spacefieldASf  
WHERE 1   
AND s.groupid=0  
AND s.uid=f.uid  


**1. id**

SELECT识别符。这是SELECT查询序列号。这个不重要,查询序号即为sql语句执行的顺序，看下面这条sql

EXPLAINSELECT\*FROM(SELECT\* FROMuchome\_space LIMIT10)ASs

它的执行结果为



可以看到这时的id变化了

**2.select\_type**

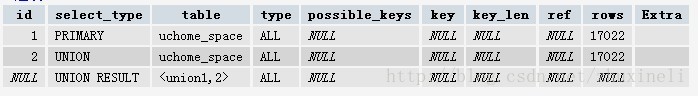
select类型，它有以下几种值

2.1 simple 它表示简单的select,没有union和子查询

2.2 primary 最外面的select,在有子查询的语句中，最外面的select查询就是primary,上图中就是这样

2.3 union union语句的第二个或者说是后面那一个.现执行一条语句，explain   
select \* from uchome\_space limit 10 union select \* from uchome\_space limit 10,10

会有如下结果



第二条语句使用了union

2.4 dependent union    UNION中的第二个或后面的SELECT语句，取决于外面的查询

2.5 union result        UNION的结果,如上面所示

还有几个参数，这里就不说了，不重要

**3 table**

输出的行所用的表，这个参数显而易见，容易理解

**4 type**

连接类型。有多个参数，先从最佳类型到最差类型介绍 **重要且困难**

4.1 system

表仅有一行，这是const类型的特列，平时不会出现，这个也可以忽略不计

4.2 const

表最多有一个匹配行，const用于比较primary key 或者unique索引。因为只匹配一行数据，所以很快

记住一定是用到primary key 或者unique，并且只检索出两条数据的 情况下才会是const,看下面这条语句

explain SELECT \* FROM `asj\_admin\_log` limit 1,结果是

http://img.blog.csdn.net/20131108000019375?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemh1eGluZWxp/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

虽然只搜索一条数据，但是因为没有用到指定的索引，所以不会使用const.继续看下面这个

explain SELECT \* FROM `asj\_admin\_log` where log\_id = 111

http://img.blog.csdn.net/20131108000237546?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemh1eGluZWxp/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

log\_id是主键，所以使用了const。所以说可以理解为const是最优化的

4.3 eq\_ref

对于eq\_ref的解释，mysql手册是这样说的:"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=比较带索引的列。看下面的语句

explain select \* from uchome\_spacefield,uchome\_space where uchome\_spacefield.uid = uchome\_space.uid

得到的结果是下图所示。很明显，mysql使用eq\_ref联接来处理uchome\_space表。

http://img.blog.csdn.net/20131108103157453?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemh1eGluZWxp/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

目前的疑问：

       4.3.1 为什么是只有uchome\_space一个表用到了eq\_ref,并且sql语句如果变成

       explain select \* from uchome\_space,uchome\_spacefield where uchome\_space.uid = uchome\_spacefield.uid

       结果还是一样，需要说明的是uid在这两个表中都是primary

4.4 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。

看下面这条语句 explain select \* from uchome\_space where uchome\_space.friendnum = 0，得到结果如下，这条语句能搜出1w条数据

http://img.blog.csdn.net/20131108110428796?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvemh1eGluZWxp/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

4.5 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。

上面这五种情况都是很理想的索引使用情况

4.6 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。

4.7 unique\_subquery

4.8 index\_subquery

4.9 range 给定范围内的检索，使用一个索引来检查行。看下面两条语句

explain select \* from uchome\_space where uid in (1,2)

explain select \* from uchome\_space where groupid in (1,2)

uid有索引，groupid没有索引，结果是第一条语句的联接类型是range,第二个是ALL.以为是一定范围所以说像 between也可以这种联接,很明显

explain select \* from uchome\_space where friendnum = 17

这样的语句是不会使用range的，它会使用更好的联接类型就是上面介绍的ref

4.10 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。（也就是说虽然all和Index都是读全表，但index是从索引中读取的，而all是从硬盘中读的）

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。

4.11  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。

**5 possible\_keys** 提示使用哪个索引会在该表中找到行，不太重要

**6 keys** MYSQL使用的索引，简单且重要

**7 key\_len** MYSQL使用的索引长度

**8 ref**ref列显示使用哪个列或常数与key一起从表中选择行。

**9 rows** 显示MYSQL执行查询的行数，简单且重要，数值越大越不好，说明没有用好索引

**10 Extra**  该列包含MySQL解决查询的详细信息。

10.1 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。一直没见过这个值

10.2 Not exists

10.3 range checked for each record

没有找到合适的索引

10.4 using filesort

MYSQL手册是这么解释的“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。”目前不太明白

10.5 using index 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信息。这个比较容易理解，就是说明是否使用了索引

explain select \* from ucspace\_uchome where uid = 1的extra为using index（uid建有索引）

explain select count(\*) from uchome\_space where groupid=1 的extra为using where(groupid未建立索引)

10.6 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。

出现using temporary就说明语句需要优化了，举个例子来说

EXPLAIN SELECT ads.id FROM **ads**, city WHERE   city.city\_id = 8005   AND ads.status = 'online'   AND city.ads\_id=ads.id**ORDER BY ads.id desc**

**id  select\_type  table   type    possible\_keys   key      key\_len  ref                     rows  filtered  Extra                            
------  -----------  ------  ------  --------------  -------  -------  --------------------  ------  --------  -------------------------------  
     1  SIMPLE       city   ref     ads\_id,city\_id  city\_id  4        const                   2838    100.00 Using temporary; Using filesort  
     1  SIMPLE       ads     eq\_ref  PRIMARY         PRIMARY  4        city.ads\_id       1    100.00  Using where**

**这条语句会使用using temporary,而下面这条语句则不会**

EXPLAIN SELECT ads.id FROM **ads**, city WHERE   city.city\_id = 8005   AND ads.status = 'online'   AND city.ads\_id=ads.id**ORDER BYcity.ads\_id desc**

**id  select\_type  table   type    possible\_keys   key      key\_len  ref                     rows  filtered  Extra                        
------  -----------  ------  ------  --------------  -------  -------  --------------------  ------  --------  ---------------------------  
     1  SIMPLE       city   ref     ads\_id,city\_id  city\_id  4        const                   2838    100.00 Using where; Using filesort  
     1  SIMPLE       ads    eq\_ref  PRIMARY         PRIMARY  4        city.ads\_id       1    100.00  Using where**

这是为什么呢？他俩之间只是一个order by不同，MySQL 表关联的[**算法**](http://lib.csdn.net/base/datastructure)是 Nest Loop Join，是通过驱动表的结果集作为循环基础数据，然后一条一条地通过该结果集中的数据作为过滤条件到下一个表中查询数据，然后合并结果。EXPLAIN 结果中，第一行出现的表就是驱动表（Important!）以上两个查询语句，驱动表都是 city，如上面的执行计划所示！

对驱动表可以直接排序，对非驱动表（的字段排序）需要对循环查询的合并结果（临时表）进行排序（Important!）

因此，order by ads.id desc 时，就要先 using temporary 了！

驱动表的定义

[wwh999](http://blog.csdn.net/wwh999/article/details/643493) 在 2006年总结说，当进行多表连接查询时， [驱动表] 的定义为：  
1）指定了联接条件时，满足查询条件的记录行数少的表为[驱动表]；  
2）未指定联接条件时，行数少的表为[驱动表]（Important!）。

**永远用小结果集驱动大结果集**

**今天学到了一个很重要的一点：当不确定是用哪种类型的join时，让mysql优化器自动去判断，我们只需写select \* from t1,t2 where t1.field = t2.field**

10.7 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。（这个说明不是很理解，因为很多很多语句都会有where条件，而type为all或index只能说明检索的数据多，并不能说明错误，useing where不是很重要，但是很常见）

如果想要使查询尽可能快，应找出Using filesort 和Using temporary的Extra值。

10.8 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

10.9 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。

实例讲解

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。

[explain 执行计划详解](http://www.cnblogs.com/liujingyuan789/p/6061188.html)

**id：id是一组数字，表示查询中执行select子句或操作表的顺序，如果id相同，则执行顺序从上至下，如果是子查询，id的序号会递增，id越大则优先级越高，越先会被执行。**

id列为null的就表是这是一个结果集，不需要使用它来进行查询。

**select\_type：**

**simple：表示不需要union操作或者不包含子查询的简单select查询。有连接查询时，外层的查询为simple，且只有一个。**

**primary：一个需要union操作或者含有子查询的select，位于最外层的单位查询的select\_type即为primary。且只有一个。**

**subquery：除了from字句中包含的子查询外，其他地方出现的子查询都可能是subquery**

**dependent subquery：与dependent union类似，表示这个subquery的查询要受到外部表查询的影响。**

**derived：from字句中出现的子查询，也叫做派生表，其他数据库中可能叫做内联视图或嵌套select。**

**union：union连接的两个select查询，第一个查询是dervied派生表，除了第一个表外，第二个以后的表select\_type都是union。**

**dependent union：与union一样，出现在union 或union all语句中，但是这个查询要受到外部查询的影响**

**union result：包含union的结果集，在union和union all语句中,因为它不需要参与查询，所以id字段为null。**

**table：**

显示的查询表名，如果查询使用了别名，那么这里显示的是别名。

如果不涉及对数据表的操作，那么这显示为null。

如果显示为尖括号括起来的<derived N>就表示这个是临时表，后边的N就是执行计划中的id，表示结果来自于这个查询产生。

如果是尖括号括起来的<union M,N>，与<derived N>类似，也是一个临时表，表示这个结果来自于union查询的id为M,N的结果集。

**type：**

**依次从好到差：system，const，eq\_ref，ref，fulltext，ref\_or\_null，unique\_subquery，index\_subquery，range，index\_merge，index，ALL。**

**除了all之外，其他的type都可以使用到索引，除了index\_merge之外，其他的type只可以用到一个索引。**

**system：表中只有一行数据或者是空表，且只能用于myisam和memory表。如果是Innodb引擎表，type列在这个情况通常都是all或者index**

**const：使用唯一索引或者主键，返回记录一定是1行记录的等值where条件时，通常type是const。其他数据库也叫做唯一索引扫描。**

**eq\_ref：出现在要连接过个表的查询计划中，驱动表只返回一行数据，且这行数据是第二个表的主键或者唯一索引，且必须为not null，唯一索引和主键是多列时，只有所有的列都用作比较时才会出现eq\_ref。**

**ref：不像eq\_ref那样要求连接顺序，也没有主键和唯一索引的要求，只要使用相等条件检索时就可能出现，常见与辅助索引的等值查找。或者多列主键、唯一索引中，使用第一个列之外的列作为等值查找也会出现，总之，返回数据不唯一的等值查找就可能出现。**

**fulltext：全文索引检索，要注意，全文索引的优先级很高，若全文索引和普通索引同时存在时，mysql不管代价，优先选择使用全文索引。**

**ref\_or\_null：与ref方法类似，只是增加了null值的比较。实际用的不多。**

**unique\_subquery：用于where中的in形式子查询，子查询返回不重复值唯一值。**

**index\_subquery：用于in形式子查询使用到了辅助索引或者in常数列表，子查询可能返回重复值，可以使用索引将子查询去重。**

**range：索引范围扫描，常见于使用>,<,is null,between ,in ,like等运算符的查询中。**

**index\_merge：表示查询使用了两个以上的索引，最后取交集或者并集，常见and ，or的条件使用了不同的索引，官方排序这个在ref\_or\_null之后，但是实际上由于要读取所个索引，性能可能都不如range。**

**index：索引全表扫描，把索引从头到尾扫一遍，常见于使用索引列就可以处理不需要读取数据文件的查询、可以使用索引排序或者分组的查询。**

**all：这个就是全表扫描数据文件，然后再在server层进行过滤返回符合要求的记录。**

**possible\_keys：查询可能使用到的索引都会在这里列出来。**

**key：查询真正使用到的索引，select\_type为index\_merge时，这里可能出现两个以上的索引，其他的select\_type这里只会出现一个。**

**key\_len：用于处理查询的索引长度，如果是单列索引，那就整个索引长度算进去，如果是多列索引，那么查询不一定都能使用到所有的列，具体使用到了多少个列的索引，这里就会计算进去，没有使用到的列，这里不会计算进去。留意下这个列的值，算一下你的多列索引总长度就知道有没有使用到所有的列了。要注意，mysql的ICP特性使用到的索引不会计入其中。另外，key\_len只计算where条件用到的索引长度，而排序和分组就算用到了索引，也不会计算到key\_len中。**

**ref：如果是使用的常数等值查询，这里会显示const，如果是连接查询，被驱动表的执行计划这里会显示驱动表的关联字段，如果是条件使用了表达式或者函数，或者条件列发生了内部隐式转换，这里可能显示为func。**

**rows：这里是执行计划中估算的扫描行数，不是精确值。**

**extra：这个列可以显示的信息非常多，有几十种，常用的有：**

**distinct：在select部分使用了distinc关键字**

**no tables used：不带from字句的查询或者From dual查询。**

使用not in()形式子查询或not exists运算符的连接查询，这种叫做反连接。即，一般连接查询是先查询内表，再查询外表，反连接就是先查询外表，再查询内表。

**using filesort：排序时无法使用到索引时，就会出现这个。常见于order by和group by语句中。**

**using index：查询时不需要回表查询，直接通过索引就可以获取查询的数据。**

**using\_union：表示使用or连接各个使用索引的条件时，该信息表示从处理结果获取并集**

**using intersect：表示使用and的各个索引的条件时，该信息表示是从处理结果获取交集**

**using sort\_union和using sort\_intersection：与前面两个对应的类似，只是他们是出现在用and和or查询信息量大时，先查询主键，然后进行排序合并后，才能读取记录并返回。**

**using where：表示存储引擎返回的记录并不是所有的都满足查询条件，需要在server层进行过滤。查询条件中分为限制条件和检查条件，5.6之前，存储引擎只能根据限制条件扫描数据并返回，然后server层根据检查条件进行过滤再返回真正符合查询的数据。5.6.x之后支持ICP特性，可以把检查条件也下推到存储引擎层，不符合检查条件和限制条件的数据，直接不读取，这样就大大减少了存储引擎扫描的记录数量。extra列显示using index condition**

**using temporary：表示使用了临时表存储中间结果。临时表可以是内存临时表和磁盘临时表，执行计划中看不出来，需要查看status变量，used\_tmp\_table，used\_tmp\_disk\_table才能看出来。**

**firstmatch(tb\_name)：5.6.x开始引入的优化子查询的新特性之一，常见于where字句含有in()类型的子查询。如果内表的数据量比较大，就可能出现这个**

**loosescan(m..n)：5.6.x之后引入的优化子查询的新特性之一，在in()类型的子查询中，子查询返回的可能有重复记录时，就可能出现这个**

**filtered：使用explain extended时会出现这个列，5.7之后的版本默认就有这个字段，不需要使用explain extended了。这个字段表示存储引擎返回的数据在server层过滤后，剩下多少满足查询的记录数量的比例，注意是百分比，不是具体记录数。**