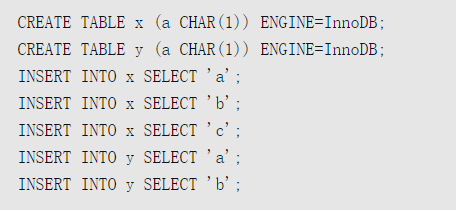
# 概述

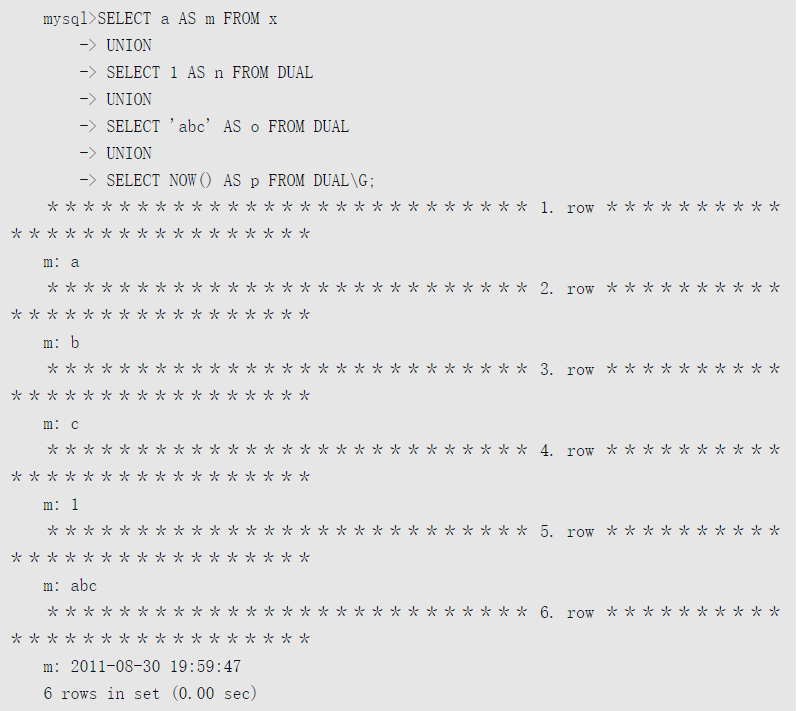
通常来说，将联接操作看成是表之间的水平操作，因为该操作生成的虚拟表包含两个表中的列。两个表之间的集合操作，一般将这些操作视为垂直操作。MySQL数据库支持两种集合操作：UNION ALL和UNION DISTINCT。

与联接操作一样，集合操作也是对两个输入进行操作，并生成一个虚拟表。在联接操作中，一般把输入表称为左输入和右输入，或者第一个输入和第二个输入。集合操作的两个输入必须拥有相同的列数，若数据类型不同，MySQL数据库会自动将进行隐式转化。同时，结果列的名称由第一个输入决定。

在进一步阐述集合操作之前，先根据下列语句创建测试表x、y：



接着来看对一个不同类型的数据进行集合操作：



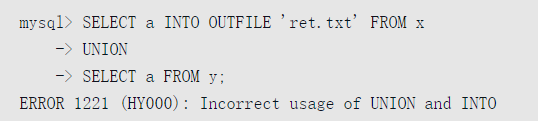
这里对各种不同的类型进行了联接，先从x表中取出类型为CHAR（1）的字符，第二个进行UNION操作的是整型数1，第三个是字符串abc，第四个是日期类型。虽然类型各不相同，但是MySQL数据库会自动对其进行判断，选出一种类型进行隐式转换。

另一方面，在这个例子中对每个选取操作都进行了别名定义，从最后的结果可以看出， MySQL数据库选择了m这个别名，也就是集合操作中第一个SELECT输入的别名。

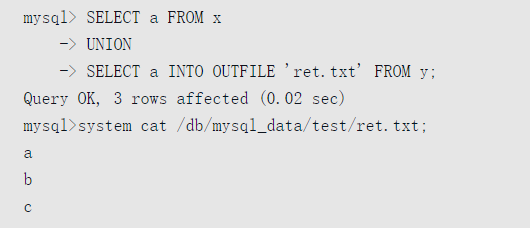
除了以下两点，集合操作中的SELECT语句和一般的SELECT查询并无不同：

1. 只有最后一个SELECT可以应用INTO OUTFILE，但是整个集合的操作将被输出到文件中。
2. 不能在SELECT语句中使用HIGH\_PRIORITY关键字。

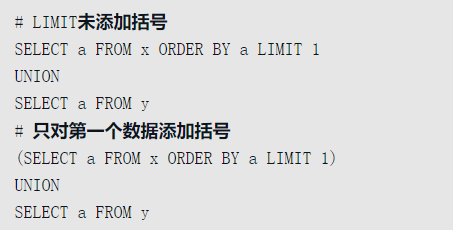
注意，在集合操作中，INTO OUTFILE只能存在于最后一个SELECT语句中，否则MySQL数据库会提示语法错误，例如：



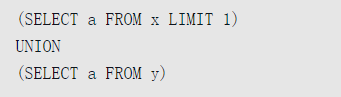
此外，虽然INTO OUTFILE只存在于最后一个SELECT语句中，但导出的结果是整个集合操作的结果，例如：



还有一点需要注意的是，若SELECT语句中包含LIMIT和ORDER BY子句，最好的做法是为参与集合操作的各SELECT语句添加括号，否则执行集合查询会得到错误提示，例如：



这个集合操作的正确SQL语句应该为：



# UNION DISTINCT/UNION ALL

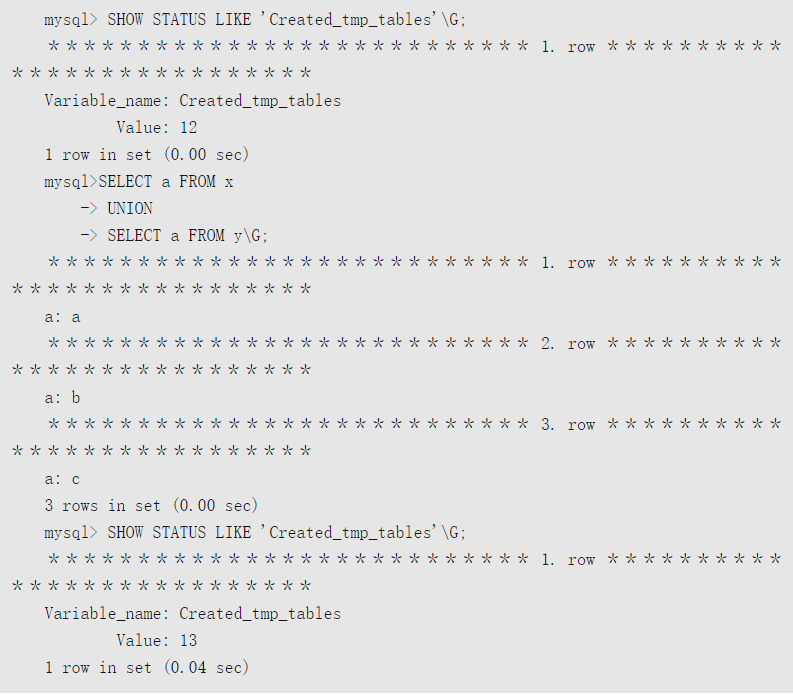
## 概述

UNION DISTINCT组合两个输入，并应用DISTINCT过滤重复项。一般省略DISTINCT关键字，直接用UNION，例如：

MySQL数据库目前对UNION DISTINCT的实现方式如下：

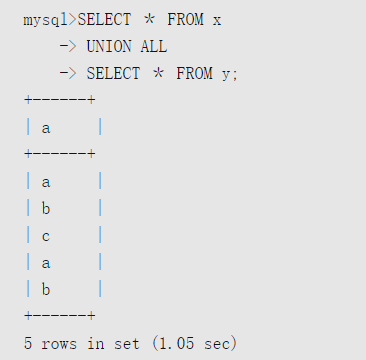
1. 创建一张临时表，即虚拟表。
2. 对这张临时表的列添加唯一索引（Unique Index）。
3. 将输入的数据插入临时表。
4. 返回虚拟表。

因为添加了唯一索引，所以可以过滤掉集合中重复的项。可以通过观察服务器状态变量Created\_tmp\_tables来确认UNION DISTINCT的实现方式，例如：



由于向临时表添加了唯一索引，插入的速度显然会因此而受到影响。如果确认进行UNION操作的两个集合中没有重复的选项，最有效的办法应该是使用UNION ALL。

UNION ALL组合两个输入中所有项的结果集，并包含重复的选项，例如：



正如前面所说，如果确认两个输入中没有重复项，应该选择UNION ALL。如果两个输入中有重复项，也可以在数据库端使用UNION ALL，在应用程序端进行DISTINCT的去重操作。

## 优化

# MINUS/EXCEPT

参考：

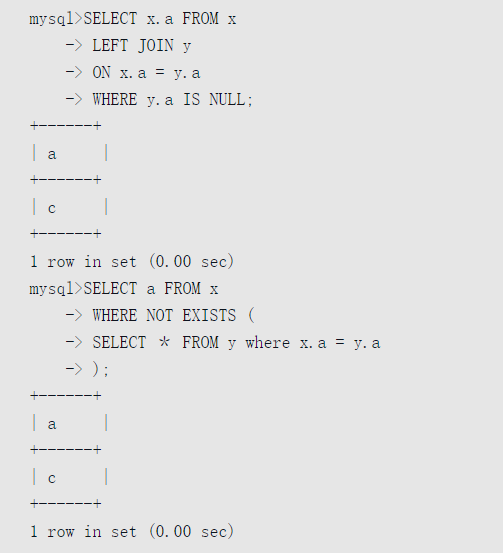
<https://blog.csdn.net/caomiao2006/article/details/52445114>

## 概述

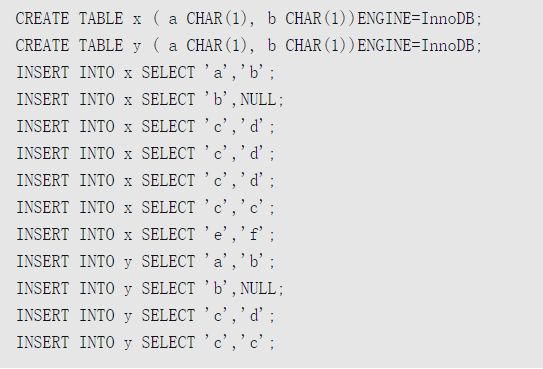
MySQL数据库并不原生支持EXCEPT的语法，不过我们仍然可以通过一些手段来得到EXCEPT的结果。EXCEPT集合操作允许用户找出位于第一个输入中但不位于第二个输入中的行数据。同UNION一样，EXCEPT可分为EXCEPT DISTINCT和EXCEPT ALL。

## 等价转换

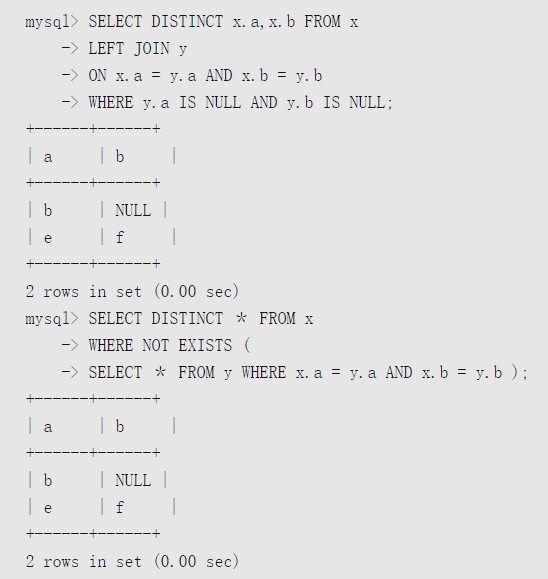
EXCEPT DISTINCT返回位于第一个输入中但不位于第二个输入中的不重复行。常见的方法是使用LEFT JOIN或NOT EXISTS。但是直接应用这些方法可能会有一些错误产生。下面通过一个例子来说明。先对前一个小节中表x和y应用EXCEPT集合操作：



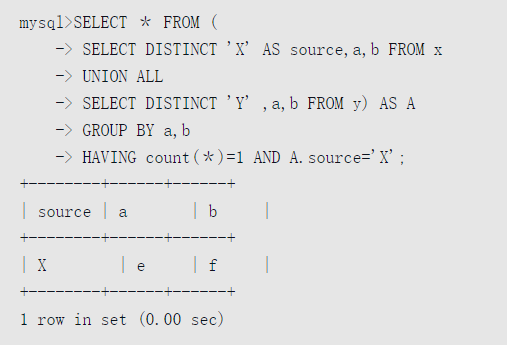
上述两种方式都能返回集合的EXCEPT操作，看似没有问题，但是如果输入项中包含NULL值，情况就不这么简单了。我们对表x和y进行如下重构：



这次表x和y中有两个列，并且可能包含NULL值。从上述的输入数据来看，对表x和y进行EXCEPT操作后应该得到（'e'，'f '），显然其存在于x表中，但不存在于y表中。按照之前介绍的LEFT JOIN和NOTEXISTS来进行EXCEPT集合操作，结果如下；



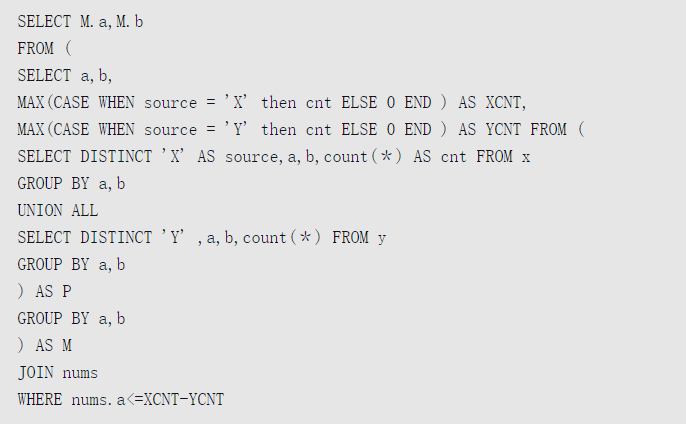
可以看到，无论采用哪种方式，最后返回的结果集都是（'b'，NULL）和（'e'，'f'）。产生这个问题的“元凶”就是b列中的NULL值。在对NULL进行“等于”操作时，返回的是NULL，即未知，这在之前已经讨论过。因此要产生正确的结果，需要使用下面的SQL语句：



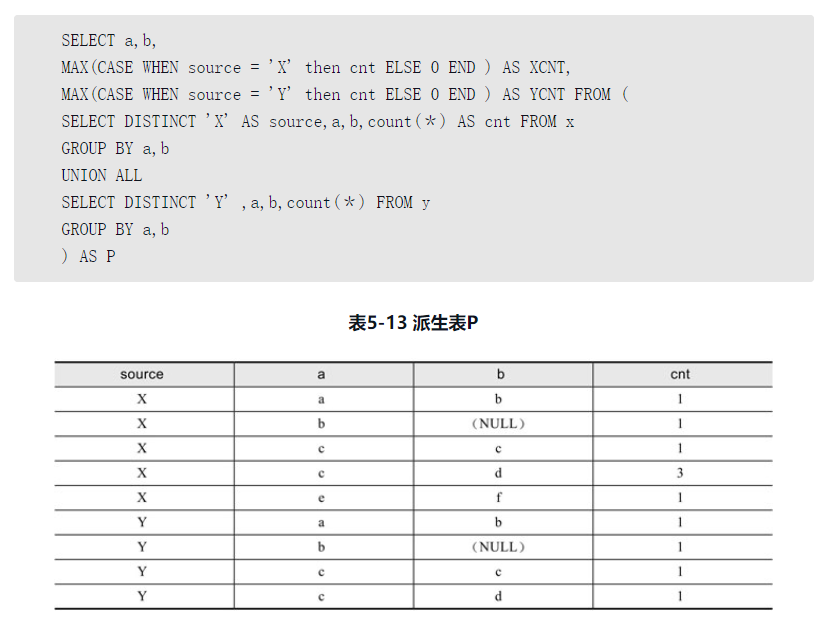
这里需要先通过UNION ALL得到所有的集合。接着添加额外的列source来区分输入数据来自于哪个表。因为在UNION操作中只选择第一个查询的列作为名称，因此只需在第一个输入中指定别名即可。接着对派生表做GROUP BY分组操作，并且加上条件来源为第一个输入的表。

相比EXCEPT DISTINCT，EXCEPT ALL要复杂得多。EXCEPT ALL不仅需要考虑行数据是否存在，还要关心每一行出现的次数。假设要返回输入A“EXCEPT ALL ”输入B的结果，如果一行数据在输入A中出现了x次，在输入B中出现了y次，则该行在输出中将出现MAX（0，x-y）次 ，即如果x大于y，该行结果中将出现x-y次，否则结果中将不包含这一行数据。

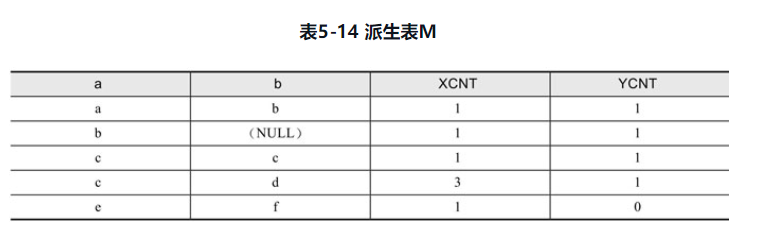
对于之前的表x和y，表x中有3条记录都为（'c'，'d'），表y中有1条记录为（'c'，'d'）。因此，除了记录（'e'，'f '）外，执行x表“EXCEPT ALL”表y后，还应该有两条（'c'，'d'）的记录。这个过程的SQL语句如下：



这里使用了数字辅助表nums来统计min（0，x-y）行的重复数据。这里的SQL语句看起来非常复杂，如果不能直接看明白，可以分别执行各个语句。这里将这个SQL语句分三个步骤来理解。先是生成派生表P，执行如下的SQL语句，结果如表5-13所示。



接着对派生表P进行分组操作，通过MAX（CASE …）表达式统计出从X和Y集合中选出的分组a和b的个数，分别用列XCNT和YCNT表示，并再生成派生表M，得到结果如表5-14所示。



最后根据XCNT-YCNT的值得到最终数据，并通过JOIN表nums来得到该行记录的多个副本。通过EXPLAIN也能很清楚地得到SQL语句的执行计划，如图5-23所示。



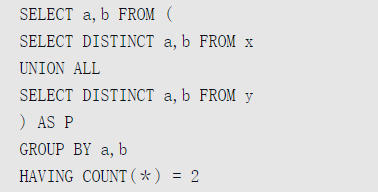
# INTERSECT

## 概述

INTERSECT返回在两个输入中都出现的行。和EXCEPT一样，不能简单地使用LEFT JOIN或者NOTEXISTS来解决INTERSECT问题，因为同样可能存在NULL值的问题。

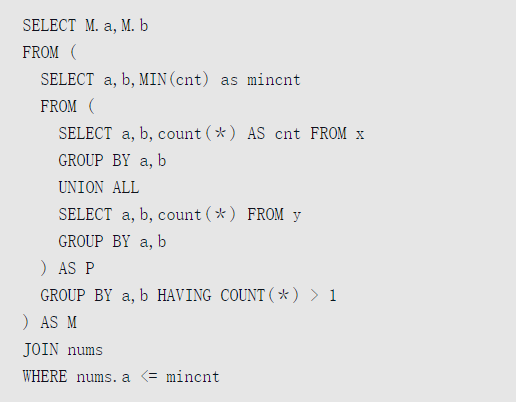
## 等价转换

对于INTERSECT DISTINCT，可以通过如下SQL语句来实现：



派生表查询对两个输入的不重复行数据执行UNION ALL操作。之后外部查询再按a和b列进行分组，并且只返回出现过两次的组。也就是说，该查询只返回在两个集中都出现过的不重复行，这就是INTERSECT DISTINCT的定义。

与EXCEPT ALL一样，INTERSECT ALL也与行的出现次数有关。如果行R在一个输入表中出现了x次，在另一个输入表中出现了y次，则它应该在结果中出现MIN（x，y）次。这个问题的解决方案同样可以参考EXCEPT ALL，借助数字辅助表nums来实现，示例如下：



派生表P对每个输入集合的不重复行记录执行UNION ALL，并计算它们在集合中出现的次数。派生表M的查询按照a和b进行分组，并通过HAVING子句筛选出在两个输入集合中都存在的行记录(COUNT(＊)>1)，之后返回它们的最小计数，即MIN（x，y）。最后通过数字辅助表nums，并根据条件nums.a <= mincnt来生成行记录的多个副本。