# 概述

在MySQL中的ORDER BY有两种排序实现方式：

1、利用有序索引获取有序数据；

2、文件排序。

在使用explain分析查询的时候，利用有序索引获取有序数据显示Using index。如果MySQL在排序的时候没有使用到索引那么就会输出using filesort，即使用文件排序。

文件排序是通过相应的排序算法，将取得的数据在内存中进行排序：MySQL需要将数据在内存中进行排序，所使用的内存区域也就是我们通过sort\_buffer\_size系统变量所设置的sort buffer（排序区）。这个sort buffer是每个Thread独享的，所以说可能在同一时刻在MySQL中可能存在多个sort buffer内存区域。

参考：

<https://blog.csdn.net/lijingkuan/article/details/70341176>

<https://www.cnblogs.com/aeolian/p/10212892.html>

# 原理

MySQL对排序有两种实现：

## 双路排序

### 原理

第一遍扫描出需要排序的字段，然后进行排序后，根据排序结果，第二遍再扫描一下需要select的列数据。这样会引起大量的随机IO，效率不高，但是节约内存。排序使用quick sort。但是如果内存不够则会按照block进行排序，将排序结果写入磁盘文件，然后再将结果合并。

**具体过程：**

1、读取所有满足条件的记录。

2、对于每一行，存储一对值到缓冲区（排序列，行记录指针），一个是排序的索引列的值，即order by用到的列值，和指向该行数据的行指针（缓冲区的大小为sort\_buffer\_size大小）。

3、当缓冲区满后，运行一个快速排序（qsort）来将缓冲区中数据排序，并将排序完的数据存储到一个临时文件，并保存一个存储块的指针，当然如果缓冲区不满，则不会重建临时文件了。

4、重复以上步骤，直到将所有行读完，并建立相应的有序的临时文件。

5、对块级进行排序，这个类似于归并排序算法，只通过两个临时文件的指针来不断交换数据，最终达到两个文件，都是有序的。

6、重复5直到所有的数据都排序完毕。

7、采取顺序读的方式，将每行数据读入内存，并取出数据传到客户端，这里读取数据时并不是一行一行读，读如缓存大小由read\_rnd\_buffer\_size来指定。

### 特点

采取的方法为：快速排序 + 归并排序。

但有一个问题，就是，一行数据会被读两次，第一次是where条件过滤时，第二个是排完序后还得用行指针去读一次，一个优化的方法是，直接读入数据，排序的时候也根据这个排序，排序完成后，就直接发送到客户端了。

## 单路排序

在MySQL4.1版本之前只有第一种排序算法双路排序，第二种算法是从MySQL4.1开始的改进算法，主要目的是为了减少第一次算法中需要两次访问表数据的IO操作，将两次变成了一次，但相应也会耗用更多的sortbuffer空间。当然，MySQL4.1开始的以后所有版本同时也支持第一种算法。

### 原理

即一遍扫描数据后将select需要的列数据以及排序的列数据都取出来，然后在sort buffer中排序，这样就不需要进行第二遍扫描了，当然内存不足时也会使用磁盘临时文件进行外排。

过程如下：

1、读取满足条件的记录

2、对于每一行，记录排序的key和数据行指针，并且把要查询的列也读出来

3、根据索引key排序

4、读取排序完成的文件，并直接根据数据位置读取数据返回客户端，而不是去访问表

### 特点

单路排序一次性将结果读取出来，然后在sort buffer中排序，避免了双路排序的两次读的随机IO。

这也有一个问题：当获取的列很多的时候，排序起来就很占空间，因此，max\_length\_for\_sort\_data变量就决定了是否能使用这个排序算法。

MySQL根据sort\_buffer\_size来判断是否使用磁盘临时文件，如果需要排序的数据能放入sort\_buffer\_size则无需使用磁盘临时文件，此时explain只会输出using filesort否则需要使用磁盘临时文件explain会输出using temporary;using filesort。

# 选择

MySQL主要通过比较我们所设定的系统参数max\_length\_for\_sort\_data的大小和Query语句所取出的字段类型大小总和来判定需要使用哪一种排序算法。

如果需要的列数据一行可以放入max\_length\_for\_sort\_data则使用一遍扫描否则使用两遍扫描（如果max\_length\_for\_sort\_data更大，则使用第二种优化后的算法，反之使用第一种算法）。所以如果希望ORDER BY操作的效率尽可能的高，一定要注意max\_length\_for\_sort\_data参数的设置。

如果数据库出现大量的排序等待，造成系统负载很高，而且响应时间变得很长，可以考虑是否为MySQL 使用了传统的第一种排序算法而导致，在加大了max\_length\_for\_sort\_data参数值之后，系统负载是否马上得到了大的缓解，响应是否快很多。

# 优化

对于文件排序的优化，应该让MySQL避免使用第一种双路排序，尽量选择使用第二种单路算法来进行排序。这样可以减少大量的随机IO操作，很大幅度地提高排序工作的效率。

1、加大max\_length\_for\_sort\_data参数的设置

在MySQL中，决定使用老的双路排序算法还是改进版单路排序算法是通过参数max\_length\_for\_ sort\_data来决定的。当所有返回字段的最大长度小于这个参数值时，MySQL就会选择改进后的单路排序算法，反之，则选择老式的双路排序算法。所以，如果有充足的内存让MySQL存放需要返回的非排序字段，就可以加大这个参数的值来让MySQL选择使用改进版的排序算法。

2、去掉不必要的返回字段或列长度尽量小一些

对于内存不是非常充裕的情况，不能强行增大配置项max\_length\_for\_sort\_data，否则可能会造成MySQL不得不将数据分成很多段，然后进行排序，这样可能会得不偿失。此时可以选择去掉不必要的返回字段或者将列长度尽可能设置小一些，让返回结果长度适应max\_length\_for\_sort\_data参数的限制。

3、增大sort\_buffer\_size参数设置

增大sort\_buffer\_size并不是为了让MySQL选择改进版的单路排序算法，而是为了让MySQL尽量**减少在排序过程中对需要排序的数据进行分段**，因为分段会造成MySQL不得不使用临时表来进行交换排序。

如果大量的查询较小的话，这个很好，就缓存中就搞定了。

1. 增加read\_rnd\_buffer\_size大小，可以一次性多读到内存中

该变量可以被任何存储引擎使用，当从一个已经排序的键值表中读取行时，会先从该缓冲区中获取而不再从磁盘上获取。默认为256K。

1. 改变tmpdir，使其指向多个物理盘（不是分区）的目录，这将机会循环使用做为临时文件区。

# 总结

当看到MySQL的explain输出using filesort时，说明排序时没有使用索引。如果输出using temporary;using filesort则说明使用文件排序和磁盘临时表，这种情况需要引起注意，效率会比较低。

总结来说，尽量避免出现文件排序，如果出现using filesort需要考虑优化。