本文基于mysql5.7

目录

·背景

·filesort的两种模式

·filesort的排序算法

·外部排序

·mysql使用filesort时的优化手段

**背景**

在日常开发中，会遇到order by的字段无法使用索引的情况，可能是数据量小不用刻意使用索引优化order by，也可能是业务逻辑复杂导致无法使用索引去优化order by。此时mysql会使用filesort对查询出的结果进行排序。

我们通过explain查看MySQL执行计划时，经常会看到在Extra列中显示Using filesort。

除order by外，Using filesort还经常出现在group by、distinct、join等情况下。

**filesort的两种模式**

1.回表排序模式

a.根据过滤条件获取到查询的数据，获取排序字段值及rowID(行指针)作为键值对。

b.排序。

c.排序完成后进行回表操作，根据rowID读取用户需要返回的数据(字段)。

Tips：由于排序后的rowID不是顺序的，会导致随机IO，Mysql引入了read\_rnd\_buffer来进行优化。Mysql在回表时，会读取一批rowID到read\_rnd\_buffer中并进行排序，再通过rowID读取数据，此时为顺序IO。可以通过read\_rnd\_buffer\_size来配置read\_rnd\_buffer的大小。

2.不回表排序模式

a.根据过滤条件获取到查询的数据，获取用户需要返回的数据及排序字段作为键值对。

b.排序

c.排序后直接返回，而不用回表查询。

不回表排序模式是典型的空间换时间的方法，但是如果用户查询的数据非常大的话，会花费很多时间在磁盘外部排序上，导致更多IO操作，效率可能还不如回表排序模式。

因此mysql提供了max\_length\_for\_sort\_data参数，当“排序的键值对大小” > max\_length\_for\_sort\_data时，MySQL认为磁盘外部排序的IO效率不如回表的效率，会选择回表排序模式，否则选择不回表排序模式。

**filesort的排序算法**

sort\_buffer\_size：mysql设置的每个数据库会话可以用来排序的内存大小。

1. 需要排序的记录小于sort\_buffer\_size时，使用快速排序直接在内存中对记录进行排序。
2. 需要排序的记录大于sort\_buffer\_size时，使用外部排序对记录进行排序。
3. 对于有limit m n的情况，使用堆排序来进行优化（因为堆排序是选择排序，比较适合有limit语句的查询，得到最大/小的m+n个记录即可，不用对所有记录进行排序）。

使用堆排序对limit m n进行优化需要同时满足以下两个条件：

1. 使用内存排序时，m+n需小于总记录个数的1/3。堆排序的比较和交换次数比快排多，时间复杂度的常数因子比快排大，效率较低。
2. m+n个记录小于sort\_buffer\_size。

**外部排序**

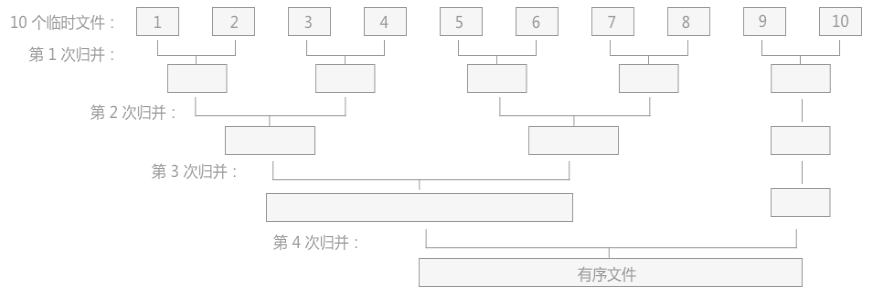
外部排序由两个相对独立的阶段完成：

1. 按可用内存大小，将外存上的大文件分割成若干个长度为l的子文件，依次读入内存并利用合适的内部排序方法对它们进行排序，然后将排序完成的有序子文件重新写入外存。
2. 对这些有序子文件进行归并，得到整个有序文件。

例：有一个含有10000个记录的文件，内存的可使用容量仅能对1000个记录进行内部排序

1.将文件分割成10个子文件，每个子文件为1000个记录。依次对10个子文件进行排 序，得到10个有序的子文件。

2.对10个子文件进行两两归并操作，直至得到一个完整的有序文件。



上面例子中对文件进行两两归并，称为"2-路归并排序"，为了减少归并的趟数，提升效率，在实际应用中可以选择k个文件进行归并，称为"k-路归并排序"。

在一般情况下，对m个初始归并段进行"k-路归并排序"时，归并的趟数s：

s=⌊ logkm ⌋

由此可见，为了减少归并趟数，可以通过增加k或者减少m来实现。

增加k可以通过败者树/胜者树实现多路归并排序。

减少m可以通过置换-选择排序算法来实现。

在使用置换-选择排序算法进行分片后，可以通过构造赫夫曼树生成最佳归并树。

（本文对外部排序就不深入了，有兴趣可以自行了解）

**mysql使用filesort时的优化手段**

1. 尽量避免使用select \*，只取需要用到的字段。
2. 合理使用limit，仅取出需要的数据。
3. 根据设备内存合理设置sort\_buffer\_size的大小，可以避免一部分的磁盘外部排序。
4. 设计表结构时合理定义字段长度。