# 17 如何正确地显示随机消息

随机显示三个单词

mysql> select word from words order by rand() limit 3;

建表语句和初始化数据

mysql> CREATE TABLE `words` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`word` varchar(64) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB;

delimiter ;;

create procedure idata()

begin

declare i int;

set i=0;

while i<10000 do

insert into words(word) values(concat(char(97+(i div 1000)), char(97+(i % 1000 div 100)), char(97+(i % 100 div 10)), char(97+(i % 10))));

set i=i+1;

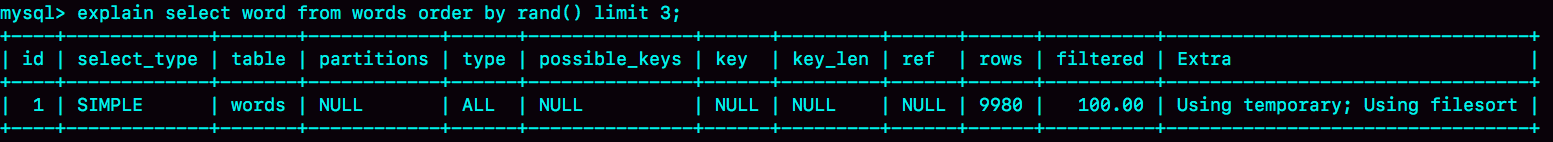
end while;

end;;

delimiter ;

call idata();

**内存临时表 ：**语句简单执行流程复杂



Extra 字段显示Using temporary ，表示需要使用临时表，Useing filesort表示要执行排序

回顾上一篇的全字段排序和rowid排序

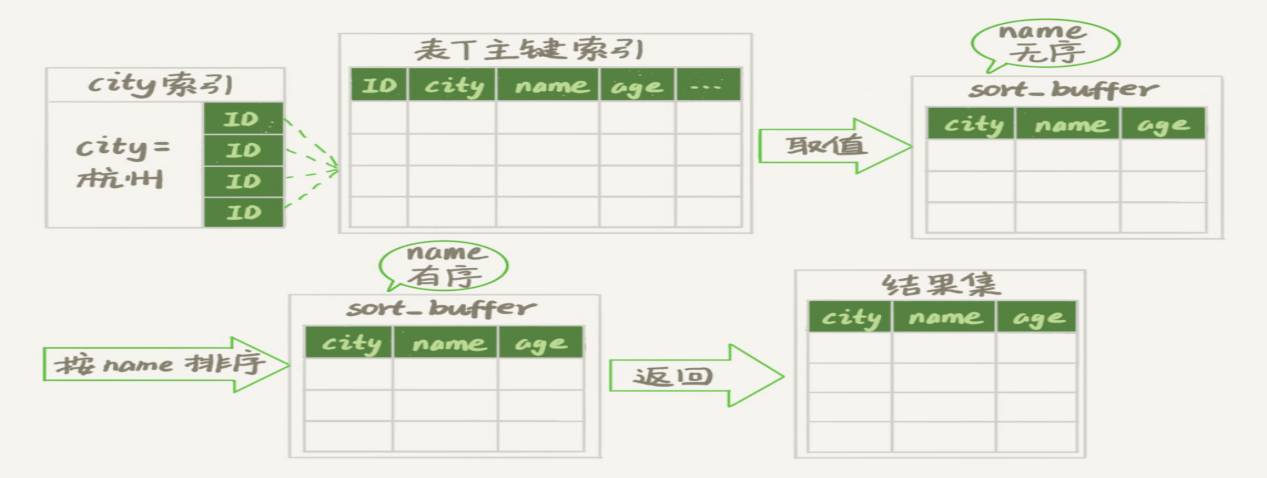


图2 全字段排序

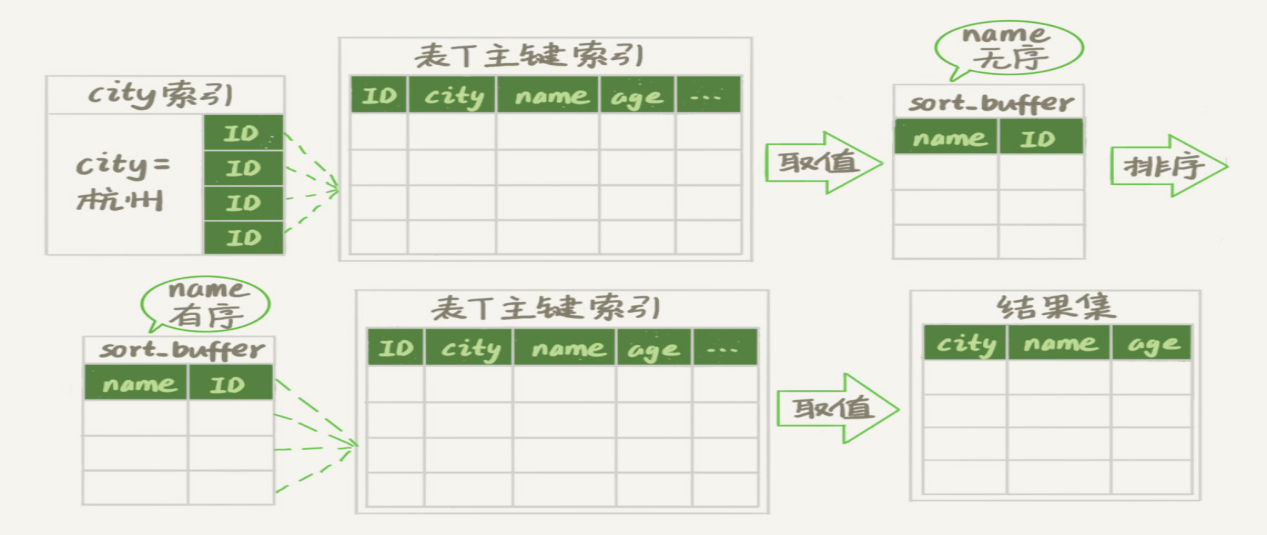


图3 rowid 排序

问题：对于临时表的排序来说，会选择哪一种算法呢？

回顾上一篇的结论：**对于InnoDB表来说，执行全字段排序会减少磁盘访问，因此会被优先选择**

强调的是“InnoDB表”，**对于内存表，回表过程只是简单的根据行的位置，直接访问内存得到数据，根本不会导致多访问磁盘。**优化器没有了这一层顾虑，会优先考虑的，就是用于排序的行越小越好，所以，MySql这时会选择rowid排序。

看看 执行流程

1. 创建一个临时表，这个临时表使用的memory引擎，表里面有两个字段，第一个字段是double类型，即为字段R,第二个字段是varchar(64)类型，即为字段W,并且，这个表没有索引
2. 从wrords表，按主键顺序取出所有的word的值，对于每一个word值，调用rand()函数

生成一个大于0小于1的随机小数，并把这个随机小数和word分别存入临时表R和W字段，到此，扫描行数10000.

1. 现在临时表有10000行数据了，接下来在这个没有索引的临时表上按照字段R排序
2. 初始化sort\_buffer,sort\_buffer中有两个字段，一个是double类型，一个整型
3. 从内存临时表一行一行的取出R值和位置信息，分别存入sort\_buffer中的两个字段里，这个过程要对内存临时表做全表扫描，此时扫描数增加10000，变成20000.
4. 在sort\_buffer中根据R的值进行排序，注意，这个过程没有涉及到表操作，所以不会增加扫描行数
5. 排序完成后，取前三个结果的位置信息，依次到内存临时表中取出word的值，返回客户端，这个过程，访问表的三行数据，总扫描行数变成20003.

查看慢查询日志 slow log

# Query\_time: 0.900376 Lock\_time: 0.000347 Rows\_sent: 3 Rows\_examined: 20003

SET timestamp=1541402277;

select word from words order by rand() limit 3;

其中Rows\_examined:20003就表示这个语句执行过程中扫描了20003行，

执行了流程图

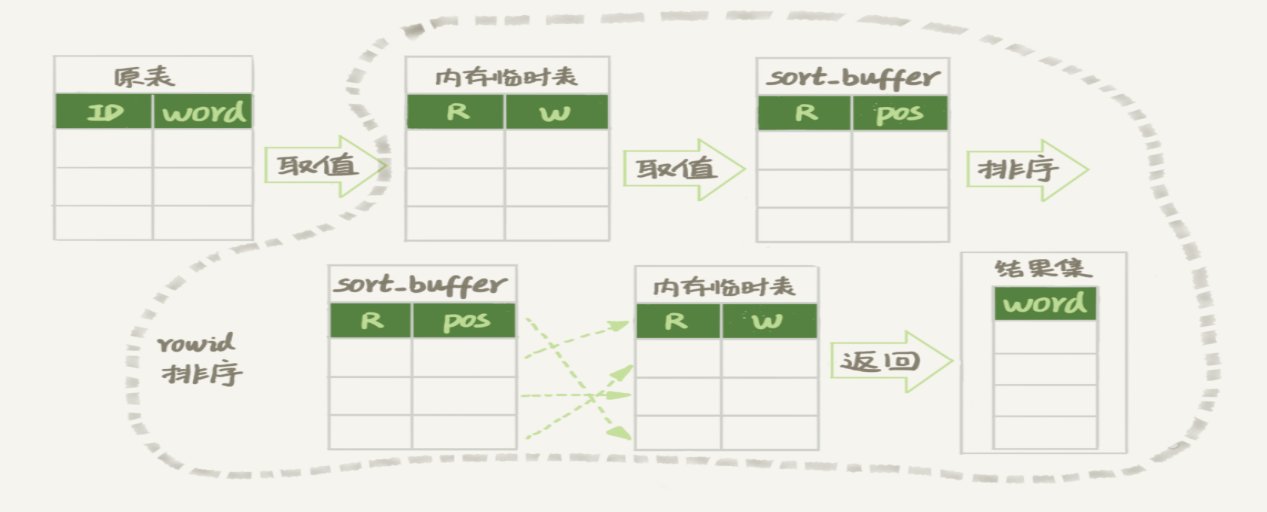


图4 随机排序完整流程图1

图中pos就是位置信息，回到一个基本概念，**MySQL的表用什么来定位一行数据的？**

如果一个InnoDB表的主键删除，会自己生成一个长度为6字节的rowid来作为主键，这也就是排序模式里面，rowid排序的来历，表示的是：每个引擎用来唯一标识数据行的信息

·对于有主键的InnoDB表莱说，这个rowid就是主键ID

·对于没有主键的InnoDB表来说，这个rowid就是系统生成的

·MEMORY引擎不是索引组织表。在这个例子里面，可以认为它是以一个数组，这个rowid其实是数组的下标

**Order by rand() 使用了内存临时表，内存临时表排序的时候使用了rowid排序方法**

**磁盘临时表**

不是所有的临时表都是内存临时表，tmp\_table\_size这个配置限制了内存临时表的大小，默认值是16M。如果临时表大小超过了tmp\_table\_size，那么内存临时表会转换成磁盘临时表

磁盘临时表的使用的默认引擎是InnoDB,由参数internal\_tmp\_disk\_storage\_engine控制

当使用磁盘临时表，对应的就是一个没有显示索引的InnoDB表的排序过程

为了复现这个过程，把tmp\_table\_size设置为1024,把sort\_buffer\_size设置为32768,把max\_length\_for\_sort\_data设置为16

set tmp\_table\_size=1024;

set sort\_buffer\_size=32768;

set max\_length\_for\_sort\_data=16;

/\* 打开 optimizer\_trace，只对本线程有效 \*/

SET optimizer\_trace='enabled=on';

/\* 执行语句 \*/

select word from words order by rand() limit 3;

/\* 查看 OPTIMIZER\_TRACE 输出 \*/

SELECT \* FROM `information\_schema`.`OPTIMIZER\_TRACE`\G

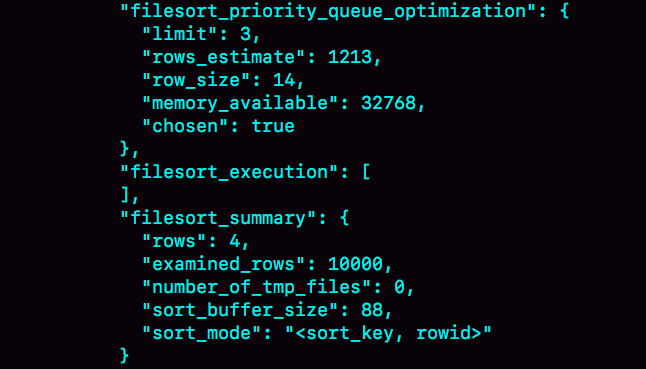


图 5 OPTIMIZE\_TRACE部分结果

因为将max\_length\_for\_sort\_data设置为16，小于word字段的长度定义，所以看到sort\_mode里面显示是rowid排序，符合预期，参与排序是随机值R值和rowid字段组成。

R字段存放的随机值是8个字节，rowid是6个字节（至于为什么是6个字节，思考太）

数据总行数是10000，140000字节，超过了sort\_buffer\_size定义的32768字节了，但是，number\_of\_tmp\_files值是0，难道不用临时文件了吗？

MySql 5.6 引入一个新的排序算法，即优先队列排序算法，看看为什么没有使用临时文件的算法，也就是归并排序算法，而是采用了优先队列排序算法。

现在用的SQL,只需取R值最小的3个rowid,但是，使用归并排序算法，虽然最终也能得到前3个值，但是这个算法结束后，已经将10000行数据都拍好序了。也就是说，后面的997行也是有序的，而优先队列排序算法，可以精确的只得到三个最小值，执行流程如下：

1. 对于这10000个准备排序的（rowid）先去前三行，构成一个堆。
2. 取下一个行（R’,rowid’），更当前对立面的最大的R比较，如果R’小于R，把这个（R,rowid ）从堆中去掉，换成（R’,rowid’）,
3. 重复第2步，直到第10000个（R’,rowid’）完成比较

优先队列执行流程图

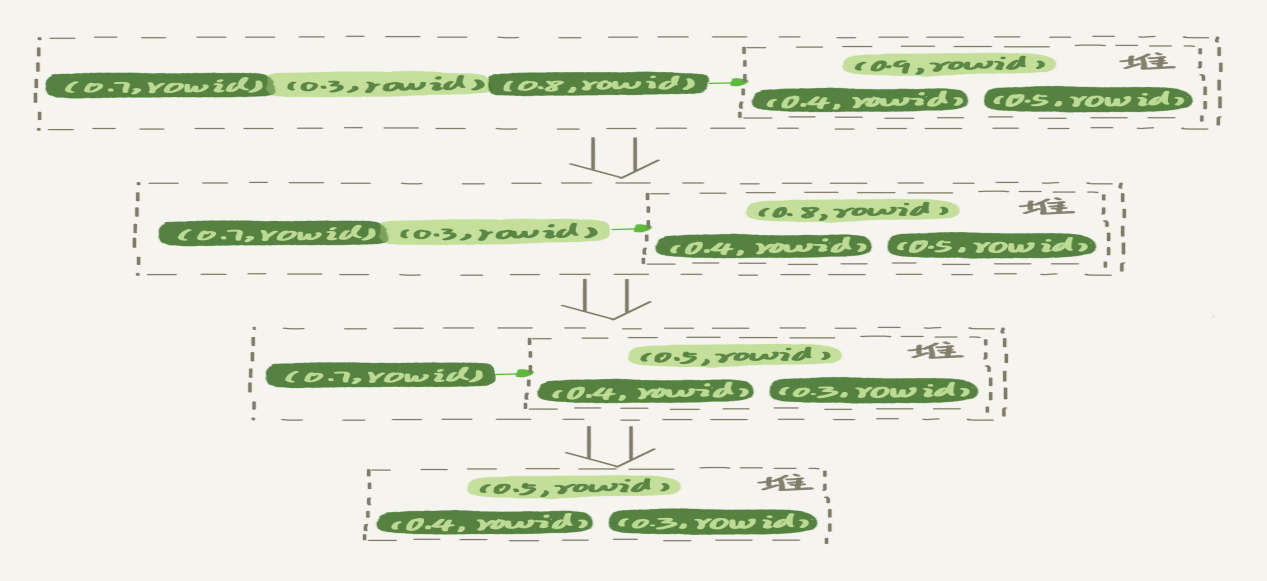


图 6 优先队列排序算法

图6是模拟6个（R，rowid）行，通过优先队列排序算法找到最小三个R值的行的过程。整个排序过程中，为了最快的拿到当前堆的最大值，总是保持最大值在堆顶，

图5 OPTIMIZE\_TRACE结果中，filesort\_priority\_queue\_optimization这个部分的chosen=true,就表示使用了优先队列排序算法，这个过程不需要使用临时文件，因此对应的number\_of\_temp\_files是0.

这个流程结束后，我们构造的堆里面，就是这10000行里面R最小的三行，然后，依次把它们的rowid取出来，去临时表里面拿到word字段，这个过程就跟上一篇文章的rowid排序的过程一样了。

上一篇文章的SQL

select city,name,age from t where city='杭州' order by name limit 1000 ;

这里用到limit，为什么没用到优先队列排序算法？原因是，这条SQL语句是limit1000,如果使用优先队列排序算法，需要维护对的大小就是1000行（R，rowid）,超过了设置的sort\_buffer\_size大小。所以只能用优先队列排序算法。

**随机排序算法**

问题先简化，如果随机选择1个word值，可以怎么做？

1. 取得这个表的id的最大值M和最小值N
2. 用随机函数生成一个最大值到最小值之间的数X=(M-N)\*rand()+ N;
3. 取不小于X的第一ID的行

随即算法1，

mysql> select max(id),min(id) into @M,@N from t ;

set @X= floor((@M-@N+1)\*rand() + @N);

select \* from t where id >= @X limit 1;

这个算法的效率很高，因为取max(id)和min(id)都不需要扫描索引，而第三步的select也可以用索引快速定位，可以认为值扫描了3行，但实际上，这个算法本身并不严格男足题目的随机要求，因为ID中间可能有空洞，因此选择不同行的的概率不一样，不是真正的随机。

不如id,分别1,2,4,5如果按照上面的算法，那么去id=4的概率是取得其他行数的两倍

如果是1,2，40000，40001，这个算法可以看作是bug

为了严格的随机结果，如下流程

1. 取得整个表的总行数，并记为C
2. 取得Y=floor(C\*rand()) floor函数的作用，就是去整数部分
3. 再用limit Y,1 取得一行

随机算法2，下面这段代码，就是上面的执行语句的序列

mysql> select count(\*) into @C from t;

set @Y = floor(@C \* rand());

set @sql = concat("select \* from t limit ", @Y, ",1");

prepare stmt from @sql;

execute stmt;

DEALLOCATE prepare stmt;

由于limit后面的参数不能直接跟变量，所以在上面的代码中使用了prepare+execute的方法，也可以把这些SQL写在应用程序中，会更简单些

随机算法2 解决了1里面明显概率不均匀问题

MySQL处理limitY,1的做法就是按顺序一个一个的读出来，丢掉前Y个，然后把下一个记录作为返回结果，因为这一步需要扫描Y+1行。再加上，第一步扫描的C行，总共需要扫描C+Y+1行，执行代价比随机算法1的代价高

当然，随机算法2跟直接order by rand()比起来，执行代价还是小很多