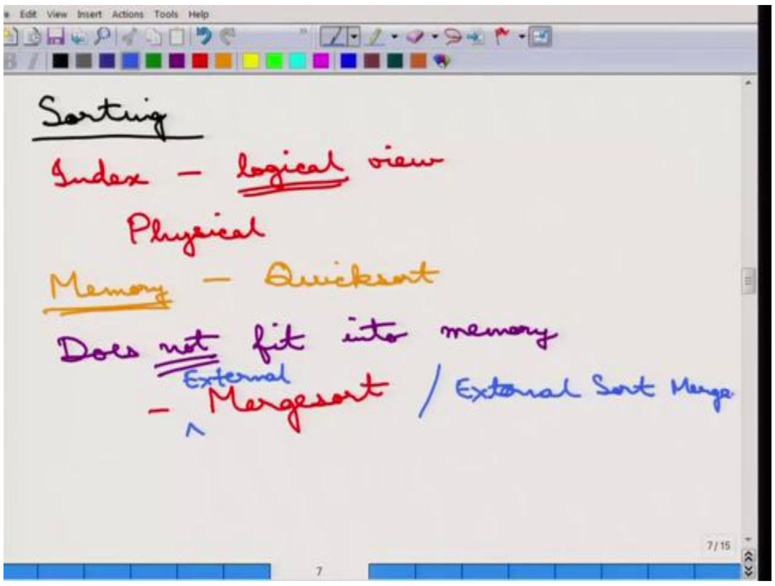
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 21
Query Processing: Sorting

(Refer Slide Time: 00:11)



Sorting

Index – logical view

Physical

Memory – Quicksort

Does not fit into memory

– External Mergesort / External Sort Merge

Next important thing that we will cover is sorting, sorting is one of the most fundamental tasks in Computer Science and databases are no exception. Then, there are different ways of sorting, the sorting is done generally for the display purposes is that, remember that order by condition in SQL. The relational algebra produces sets, so the sorting has got no valid here, but for display purposes it is being useful.

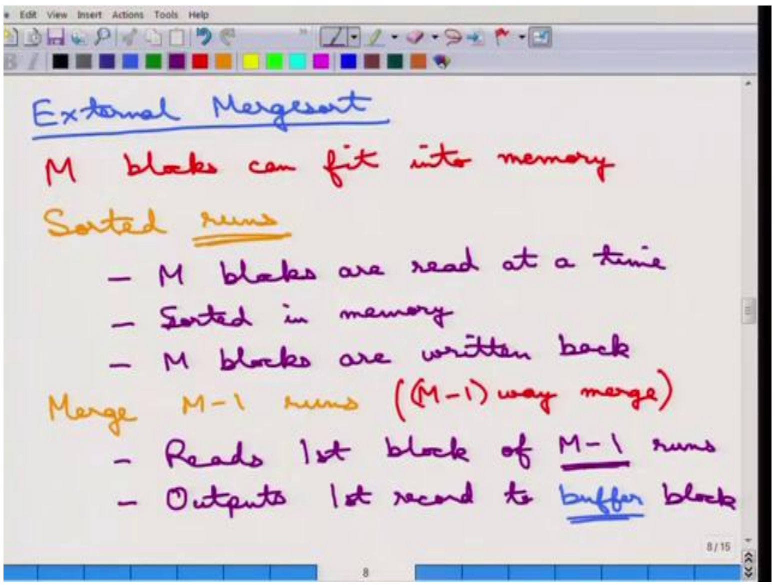
However, it is not the display purpose that is useful, it can be it is also useful, because when the tuples are produced in a sorted order and these tuples are in an intermediate result that can feed to other query expressions later on. If the tuples are produced in a sorted order that may help the query algorithm later. So, sorting is the very, very important operation, that databases also do very regular and what are the different kinds of sorting that we will go over.

First of all understand that, the index, so index generally stores the ripples in a sorted manner, but index only provides the logical view. Because, it has got only pointers to the actual tuples, so it is only a logical view, whereas, sorting here means that we want the physically sorted tuples. So, there is a difference between just outputting the index etcetera.

So, there are two different ways that sorting can be done, when the tuples fits in memory mean all the tuples fits in memory, then quick sort can be used and I am assuming, all of you know what the quick sort algorithm is. Quick sort can be used when it fits in memory. However, when it does not fit into memory, then quick sort is not a good algorithm and then what is being done is the merge sort algorithm.

But these merge sort is not the basic merge sort that we have studied, this is that merge sort with the twist, this is called an external merge sort algorithm. And we will see what the external merge sort algorithm next, external merge sort is also sometimes called external sort merge, this is the same name that is given to you.

(Refer Slide Time: 02:26)



External Mergesort

M blocks can fit into memory

Sorted runs
- M blocks are read at a time
- Sorted in memory
- M blocks are written back

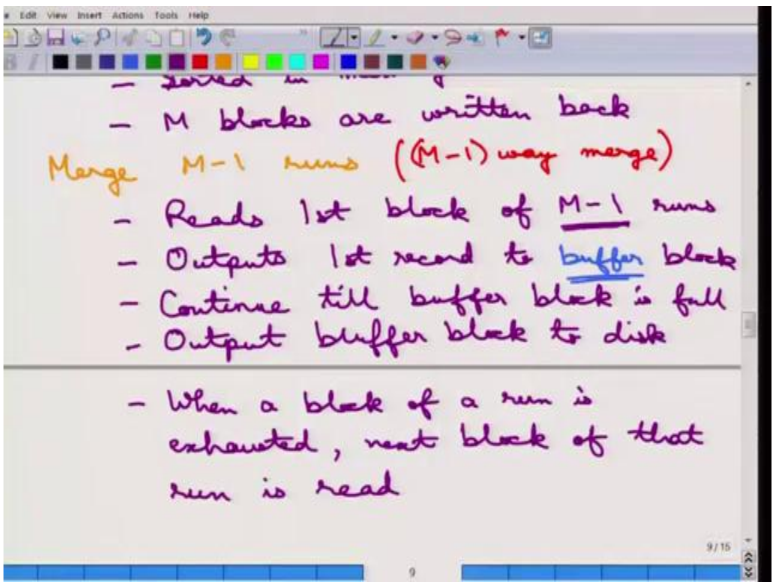
Merge M-1 runs ((M-1) way merge)
- Reads 1st block of M-1 runs
- Outputs 1st record to buffer block

So, we will next study, what the external merge sort algorithm is, so we will study what the external merge sort algorithm is. Now, suppose only M blocks can fit into memory and the size of the relation is of course, more than M block; that is why this inter relation cannot fit into the memory. So, what is being done is sorted runs are produced sorted runs, now what is a run is the following way it is being done.

So, first of all M blocks are read at a time, then they are sorted in memory using any algorithm, let us say the quick sort or whatever sorted in memory, then these M blocks are return back. So, these are the three steps that is done for each run of M blocks, fine. So, what is being done is M blocks are read, the first M blocks are read, then they are again sorted in memory, then they are return back. Then, once that is run, the next set of M blocks are read, then they are again sorted in memory and then, they are return back and so on, so forth.

So, essentially these one is called a run, this is that so for each run M blocks is read and sorted, fine. So, this is the first step, after that the merge step comes, which is merge M minus 1 runs at a time, this is called M minus 1 way merge. So, what this algorithm does is that, it reads first block of M minus 1 runs, then it outputs first record that is the based out of the M blocks to the buffer blocks, so there is a buffer block. So, that is why you see this is M minus 1 this is a buffer block; that is being test, so that is the buffer block, which is being done.

(Refer Slide Time: 05:08)


- M blocks are written back

Merge M−1 runs ((M−1) way merge)

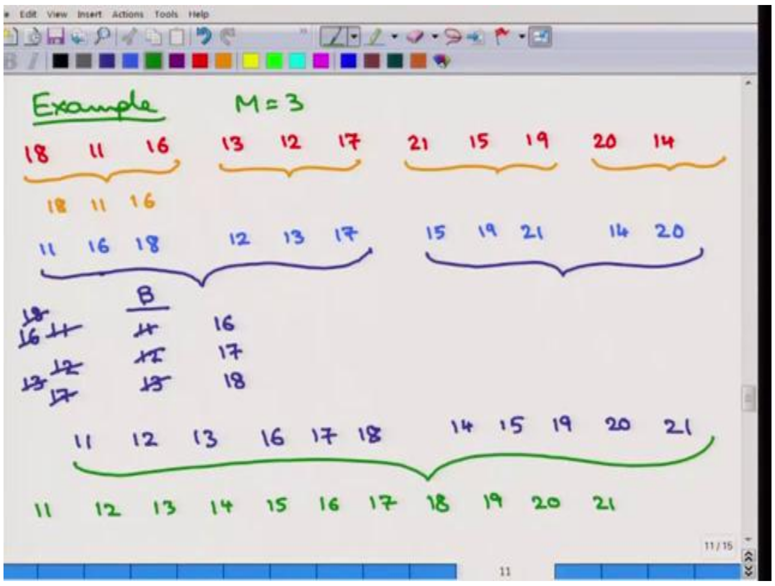
- Reads 1st block of M−1 runs
- Outputs 1st record to buffer block
- Continue till buffer block is full
- Output buffer block to disk

- When a block of a run is exhausted, next block of that run is read

And once the buffer block is done, continue till buffer block is full, then output buffer block to disk. After this, a new buffer block is done and so on, so forth this exhausted is done. And when a block of a run is exhausted, next block of that run is read and this continues on. So, these may not even finish in the second merge, because M minus 1 runs may not be enough, there may be more than M minus 1 blocks that are produced in the first stage, so this keeps on going and finally, the output is produced.

Now, doing the one important thing to remember is that, this can go on in more than one pass, so this keeps on passing. But, the more important part is that after a b pass, some part of the input is processed and the output is sorted up to that point. So, let us take an example to understand the external sort merge algorithm in more detail.

(Refer Slide Time: 06:39)



So, here is an example and suppose here the M equal to 3, so only three blocks can fit at a time and this is the data that we need to sort, so 18, 11, 16 etcetera. So, at the first pass three things will be input to memory, so 18, 11 and 16 will be input to memory and of course, this will be sorted and what will be output is 11, 16, 18. Now, similarly more runs will be created and the outputs will be return back to the disk and these are the outputs that will be created after the first run.

Now, note that case may happen that in some cases, the last block may be half full, but it does not matter, all it needs to create is to put in a sorted run, so now, note that each of this is a sorted run. Now, since the next state will be trying to do the following is that, since M equal to 3. So, the first two blocks first sorted runs will be brought into memory, now not everything will be brought into memory, only the first blocks will be brought into memory.

So, what will be there in a memory is 11 and 12, then there is a buffer block created, the buffer block which essentially output the first out of these to the least out of this 11 and 12. So, 11 will be output will be read into the buffer block and note that each buffer block can hold up to three of these values. So, 11 will be created, now as soon as 11 is output, the next one from these block which had a 11, next one from these run which had 11, then the next block will be read, which means 16 will be brought into memory.

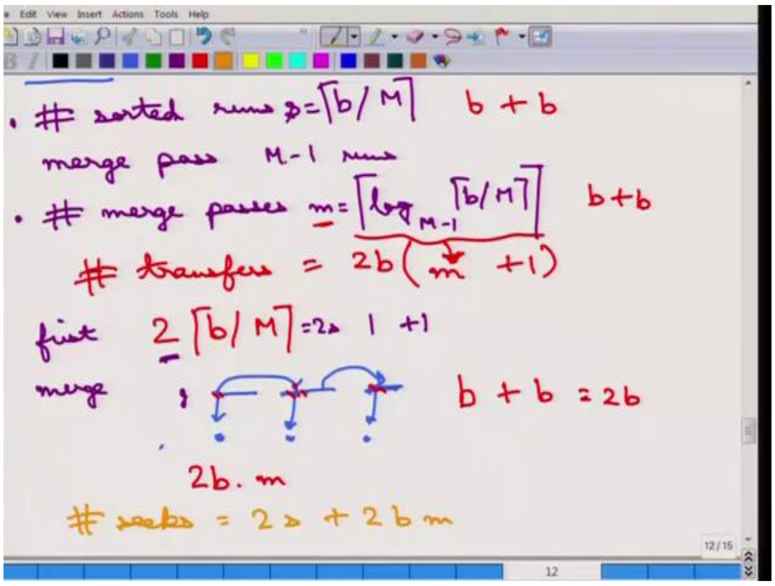
Now, the next comparison will be between 16 and 12 and 12 will be output, once 12 is the output 13 will be brought into memory. Then, out of these 13 will be created and once sorting is done 17 will be bought into memory, more important you what will happen is that at this finding time these buffer block is full, because you can contact three of these things and this is done. So, these will be return back to the disk and the buffer will be emptied out.

So, now, the buffer block can hold from other values, so this is return back to the disk. Then, out of, then the same process is continued, so out of 16 and 17, 16 will be, then put into the buffer block after 16, then 18 will be brought, then 17 will be output, once 17 is erased, there is nothing more left in the blocks. So, nothing else is brought and then, 18 is done, so 18 will be also erase, so then this will be produced as 16, 17 and 18.

Now, in the same manner this two will be sorted and, what will be produced here is the we can simply write it down 14, 15, 19, 20, 21. In the next step this there are only two blocks left know, so these two will be sorted and the total sorted order will be produced 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21; that is being done. So, this is you see the this follow the merge sort procedure, but couple of things taken into account is that only three things at a time is bigger than, because; that is what the memory as the capacity and in the buffer block is used etcetera, this is the external merge sort algorithm and this is how the things are actually being done.

Now, note in these example two passes when not enough, it requires three passes, because even in the second pass they have us not enough space the first pass is always equal to create the sorted runs and then, after that they may be many passes required.

(Refer Slide Time: 10:31)



- # sorted runs $s = \lceil b/M \rceil$ $b + b$

merge pass $M-1$ runs

- # merge passes $\underline{m} = \left\lceil \log_{M-1} \lceil b/M \rceil \right\rceil$ $b+b$

transfers $= 2b(m + 1)$

first $\underline{2} \lceil b/M \rceil = 2s$ 1 +1

merge $b + b = 2b$

$2b \cdot m$

seeks $= 2s + 2bm$

Now, one important thing to do is to analyze these algorithms. So, how do we analyze the cost of these external merge sorts, so cost of these external merge sort algorithm is the done in the following manner. Suppose the relation contains b blocks and memory is can contain only M blocks of course, b is greater than M. So, the total number of sorted runs; that is produced, the first thing is that number of sorted runs; that is produced is a b by M.

Now, in each merge pass M minus 1 runs are sorted. So, the total number of merge passes, that are required is log of M minus 1 of these number b by M, so this is the thing and total number of things; that is required is the following total number of merge passes required is the log of M minus 1 of these the total number of sorted runs. Now, during each of these merge passes and the first pass, so these are the two important passes all the blocks are read and all the blocks are written.

So, the total number of block transfers is that, so if all the number of blocks are read; that is b and all the blocks are written that is b again, so total number of transfers is simply 2 b times this number, this entire number plus 1 for the first pass. So, if we say this is and if we say this is small m, then this is m plus 1, so; that is the total number of transfers; that is required to understand this. So, now, what is the total number of seeks, so how do we compute the total number of seeks is that.

So, the initial sorted run reads all these M blocks and there is only one seek, so for the total number of seeks is twice of b by M. Because, every time total number of sorted runs is that every time for the first run of the portion of the data is read and then, written. So, for each reading the sorted run requires one seek and then, writing it back requires another seek. So, its twice the number of sorted runs use essentially 2 s; that is, what is the number of things for the first pass this is only for the first pass.

Now, during the merge pass this is the first sorted run think, so each of the merge passes, what is being done? Is that blocks from different runs may be a read. So, first the one block is read, so suppose there are multiple blocks this is one block, this is one block, where one block. So, first this block is being read, then next block may be read, then next block may be read and so on, so forth. So, the number of this a worst case, is that every time something is output to the buffer block another block is being read.

So, the number of seeks is essentially just copying from one block to another one run to the other, so there may be many, many seeks. So, the number of seeks is essentially all the number of, so for each of these there may be again 2 b. So, for each block may be need to be read and return for each of these blocks there is b seeks for reading and b seeks for outputting, because for each of these blocks, again it hopes to some other things and so on, so forth, so each of these b there is a jump.

So, there is 2 b seeks for merge phase, so the total number of merge things there is that 2 b times this M there M, that we use there. So, the total number of seeks therefore, is essentially the total number of seeks in the entire algorithm is simply 2 s plus 2 b times M, so this can be the worst case number of seeks. So, the total number of transfers is 2 b times M plus 1 and the total number of seeks is 2 s plus 2 b M and this is of course, the worst case analysis of the merge sort plan it can be bad, but it generally it is lesser than that, because we have assumed the worst case that every blocks jumps to is the seek.