External Merge Sort

Purpose: The size of the file is too big to be held in the memory during sorting. This algorithm minimizes the number of disk accesses and improves the sorting performance.

Example:

No. of rows (records) to be sorted = 110, 814
Size of each record = 1500 bytes

Size of each disk block (database page) = 8 KB (8,192 bytes with data size 8060 bytes)
Each record is to be stored in only one disk block

No. of records / diskBlock = $\lfloor 8060/1500 \rfloor$ = 5 records/block

Total no. of disk blocks for the entire file = $\lceil 110{,}814 / 5 \rceil = \lceil 22{,}162.8 \rceil$ = 22,163 blocks

The amount of memory available for the sorting = 10 blocks (buffer size)

Sorting algorithm consists of two phases: **Sort phase** followed by **Merge phase**

Sort phase:

- Divide the entire file into groups of 10 blocks (memory buffer capacity)
- No. of groups = $\lceil 22{,}163 / 10 \rceil = \lceil 2216.3 \rceil$ = 2217 groups
- The sort phase code will run 2217 times
- In each run, read one group of disk blocks into the memory buffer (10 x 5 = 50 records), sort the records in the memory buffer, save the sorted records in a temporary sub file.

At the end of this phase, 2217 temporary sorted sub files will be created.

Code:

k – the no disk blocks the memory buffer can hold
m – the total no. of runs (groups)
i – the run index value (1 to 2217)

Merge phase: 2217 sorted sub files will be merged into a single sorted file in several passes.

Pass 1:

Input: 2217 sorted sub files each with 10 disk blocks (50 sorted records)
Perform several runs (uses 10 block space of the memory buffer) for the merge

Run 1: Read the first 9 sorted sub files (one disk block from each file)
Write 1 big merged sub file (final size = 90 blocks = 450 records)
Run 2: Read the next 9 sorted sub files
Write 1 big merged sub file (90 blocks)
…..
Run 246: process 9 sorted sub files
Run 247: process the last remaining 3 sorted sub files (output 23 blocks)

Total number of runs in Pass 1: $\lceil 2217 / 9 \rceil = \lceil 246.33 \rceil$ = 247 runs

Pass 2:

Input: 247 sorted big sub files each with 90 disk block (450 sorted records)
Perform several runs (uses 10 block space of the memory buffer) for the merge
Run 1: Read the first 9 sorted big sub files (one disk block from each file)
Write 1 big merged sub file (final size = 810 blocks = 4050 rec)
Run 2: Read the next 9 sorted big sub files
Write 1 big merged sub file (810 blocks)
…..
Run 27: process 9 sorted big sub files
Run 28: process the last remaining 4 sorted big sub files (output 293 blks)
Total number of runs in Pass 2: $\lceil 247 / 9 \rceil = \lceil 27.44 \rceil$ = 28 runs

Pass 3:

Input: 28 sorted big sub files each with 810 disk block (4050 sorted records)
Perform several runs (uses 10 block space of the memory buffer) for the merge
Run 1: Read the first 9 sorted big sub files (one disk block from each file)
Write 1 big merged sub file (final size = 7290 blocks = 36450 rec)
Run 2: Read the next 9 sorted big sub files
Write 1 big merged sub file (7290 blocks)
Run 3: process 9 sorted big sub files
Run 4: process the last remaining 1 sorted big sub file (output 293 blocks)
Total number of runs in Pass 3: $\lceil 28 / 9 \rceil = \lceil 3.11 \rceil$ = 4 runs

Pass 4:

Input: 4 sorted big sub files each with 7290 disk block (36450 sorted records)
Perform several runs (uses 10 block space of the memory buffer) for the merge
Run 1: Read the 4 sorted big sub files (one disk block from each file)
Write 1 big merged sub file (final size = 22163 blocks = 110,814 rec)
Total number of runs in Pass 4: $\lceil 4 / 9 \rceil = \lceil 0.44 \rceil$ = 1 run

Initial runs in the sort phase N_R = 2217

Degree of merging D_M = 9

Number of passes = $\lceil \log_{DM} N_R \rceil = \lceil \log_9 2217 \rceil = \lceil (\ln 2217) / (\ln 9) \rceil = \lceil 3.506 \rceil = 4$

Code:

i – the pass index value (1 to 4)
p – the number of passes
n – run index for the current pass
q – the total no. of runs for the current pass