CSE 215: DATABASE Assignment

Submitted by:

Iftekhar Hakim Kaowsar

ID – 1705045

Section – A

Date of submission – 23/12/2020

Designing the storage manager for a Database Management System –

My ID is 1705045, so the block size is 6 KB.

There will be 2 types of header.

One file header. It will contain total number blocks, tuples, pointer to first block in file.

Block header for each block. It will contain pointer to next block.

Algorithms:

We will be using sequential data structure, it is notable that it has some merits and demerits. Here searching is quite time worthy and memory usage is better.

Inserting blocks in between blocks:

1. Allocate a new block of size 6 KB. Define it to be Y.
2. If total number of block = 0:
   Pointer to first block in file header = Y
3. New number of blocks in header= Number of blocks in header + 1
4. We find the block after which the new block will be inserted. Define it to be X.
5. Set Next block pointer of Y = Next block pointer of X
6. Set Next block pointer of X = Y

Inserting blocks in the end of file:

1. Allocate a new block of size 6 KB. Define it to be Y.
2. If total number of block = 0:
   Pointer to first block in file header = Y
3. New number of blocks in header= Number of blocks in header + 1
4. We need to find current last block of the file. Define it to be X.
5. Set Next block pointer of X = Y.

Inserting fixed length tuples into the block in *overlapping mode*:

1. Find number of tuples current in the block. Define it to be cnt.
2. Current_size = cnt * single tuple size (in KB)
3. Current size + single tuple size (in KB) <= 6:
   - Find last existing tuple. Define it to be A.
   - Set next tuple pointer of A = new tuple to be inserted.
4. Otherwise:
   - Add a new block next to the current block.
   - Insert the new tuple distributed over current block and then newly inserted block.
5. Increase number of tuples in header

Inserting fixed length tuples into the block in *non-overlapping mode:*

1. Find number of tuples current in the block. Define it to be cnt.
2. Current_size = cnt * single tuple size (KB)
3. Current size + single tuple size (in KB) <= 6:
   - Find last existing tuple. Define it to be A.
   - Set next tuple pointer of A = new tuple to be inserted.
4. Otherwise:
   - Add a new block next to the current block.
   - Insert the tuple in newly inserted block.
5. Increase number of tuples in header

Find i-th tuple *(in overlapping and non-overlapping mode):*

1. If number of tuples in header < i:
   Return Null
2. Set n = 1
3. Current_tuple = First tuple in file
4. While n<i:
   Current_tuple = Current_tuple -> next_tuple_pointer
   n+=1
   (here we may have to change block)
5. Return Current_tuple

Delete i-th tuple *(in overlapping and non-overlapping mode):*

1. Find i-th tuple (using algorithm written above). Define it to be X.
2. If X = NULL:
   Return
3. If X -> next_tuple_pointer = NULL:
   Remove tuple X, de-allocate its space
   Y = (i-1) th tuple
   Y -> next_tuple_pointer = NULL
4. Otherwise:
   Swap i-th tuple with last tuple in file
   Delete last tuple (or total_tuple_number th tuple) using step 4
5. Decrease number of tuples in header

Delete a set of tuple *(in overlapping and non-overlapping mode):*

1. For all element in set:
   Find the tuple in file (using algorithm written above)
   Delete it (using algorithm written above)

Delete a range of tuples *(in overlapping and non-overlapping mode):*

1. Let [L,R] be the range
2. For i=R to L:
   Delete i-th tuple (using algorithm written above)

Update i-th tuple *(in overlapping and non-overlapping mode):*

1. Find the i-th tuple (using algorithm written above)
2. Change its value as required

Update a set of tuples *(in overlapping and non-overlapping mode):*

1. For all element in set:
   Find the tuple in file
   Update its value as required

Update a range of tuples *(in overlapping and non-overlapping mode):*

1. Let [L,R] be the range
2. For i=L to R:
   Find the i-th tuple in file (algorithm written above)
   Update its value as required