Middle East Technical University | Department of Computer Engineering

CENG 351

Data Management and File Structures

Fall 2015-2016

Programming Assignment 2

Due date: January 11, 2016, Monday, 23:55

1 Constructing a B+ Tree for an existing file

In this programming assignment, you are going to construct a primary B+ Tree index for an existing already sorted file (input.txt) and handle search operation on this tree. This assignment requires the implementation of bulk loading a B+tree given in Section 10.8.2, in Ramakrishnan's book: Database Management Systems, 3^{rd} edition, starting at page 360. You will implement PA2 using C or C++ programming language.

2 Command Line Specifications

Your program will be invoked with necessary parameters for constructing the B+ tree. The parameters will be provided as pairs, as you may be familiar from Unix:

```
-<identifier> <value>
```

Arguments are as follows:

primary-key: The name of the field that is used as a key for the construction of the B+ Tree index.
btree-order: Order of the B+ Tree index to be created.
btree-bkfr: Bucket factor of B+ Tree.

An example invocation will be:

```
./pa2 --btree-order <order_of_the_tree> --btree-bkfr <bucket_factor>
--primary-key <the_name_of_the_primary_key_field>
```

Of course, this ordering is not strict, the below example will be valid as well:

```
./pa2  --primary-key <the_name_of_the_primary_key_field> --btree-order <order_of_the_tree>
--btree-bkfr <bucket_factor>
```

Please note that given input.txt will be already sorted in ascending order w.r.t. *the_name_of_the_primary_key_field* given as program argument.

3 General Specifications

Your program should do the following:

1. Read the bucket factor as input. Note that the input file can have an arbitrary number of records. Therefore, the number of buckets created will vary according to the file-size and the nature of the data.
2. Read the order of the B+ Tree as input. In this phase of the program, assume that the size of the available memory is big enough to hold all index nodes.

3. Read the name of the field to be used as a primary key for the construction of the B+ Tree index as input.
4. Read the input file (input.txt) record by record, create your buckets by organizing the records into buckets with 70% fullness.
5. While creating your buckets, construct your B+ tree and write the data buckets to a binary output file (output.dat) with a header.

After constructing the B+ tree index, your program will not exit, but wait for user input. The possible inputs the user can give are as follows:

a) Print the B+ tree

print_btree

Your program will print the whole B+ tree in an breadth first order. The index nodes must beprinted between tags <index>...</index>, the data nodes must be printed in tags <data>...</data> and the records must be printed in tags <record>...</record> For example, if you have the tree in Salzberg's book, pp.145, you must output:

Sample Print B+tree in breadth first order

```
<index>
Donna
</index>
        <index>
        Brian
        Bruce
        </index>
        <index>
        Paul
        </index>
                <data>
                <record>Aaron|blue|47</record>
                </data>
                <data>
                <record>Brian|red|23</record>
                </data>
                <data>
                <record>Bruce|red|59</record>
                <record>Claire|green|60</record>
                </data>
                <data>
                <record>Donna|blue|21</record>
                <record>Marcia|blue|79</record>
                </data>
                <data>
                <record>Paul|green|12</record>
                <record>Tim|red|2</record>
                </data>
```

If the tree is empty, you will output:

empty

b) Searching the key equality in B+ tree

search_btree_equality <key> <field_list>

Your program will search through the B+ tree for the given key and print the associated record from the file to the standard output. This command has a mandatory argument, <key>, and an optional argument, <field_list>. <key> is the argument that is used to select the wanted records. You will output the records which has this value in their primary key field (That is, the field which you will use to construct the B+ tree). <field_list> argument is used to specify which fields of the selected records are to be printed. For example:

search_btree_equality Marcia color name blue|Marcia

Note that the field names are separated by a " " character.
If <field_list> argument is not given, your program will output the whole record, for instance:

search_btree_equality Marcia Marcia|blue|79

If the key is not found in the file, your program will output:

```
no matching records found
```

c) Searching the given key value range in B+ tree

```
search_btree_range <lower_bound_key> <upper_bound_key> <field_list>
```

Your program will output all the records in the file whose key values are between <lower_bound_key> and <upper_bound_key>. An example call will be:

```
search_btree_range Greg Martin name color
Marcia|blue
```

The name and color fields of all the records with key >= Greg and <= Martin will be printed.

At most one of these arguments may be NULL. If the lower bound is NULL,

```
search_btree_range NULL Martin name color age
Aaron|blue|47
Brian|red|23
Bruce|red|59
Claire|green|60
Donna|blue|21
Marcia|blue|79
```

All the records from the beginning of the file up to the records with key = Martin (including these records also) will be printed.

If the upper bound is NULL,

```
search_btree_range Greg NULL
Marcia|blue|79
Paul|green|12
Tim|red|2
```

All the records beginning from the ones with key = Greg (including these records also) up to the end of the file will be printed.

If no matching record is found in the file, your program will output:

```
no matching records found
```

d) Quitting the program

```
quit
```

The primary key field that you will build the tree according to will either be a string or an integer.

4 Input File Organization

Assume that the input file is a sorted file with header information.

The structure of the sorted file will be as follows:
A Header line will start with '#' character.
There will be 1 + numberofFields header lines.
Header lines will be followed by record lines.

#recordSize numberofFields
#field1Type field1Name
#field2Type field2Name
...
...
#fieldNType fieldNName
record1
record2
...
...
recordK

where each field of record is separated by '|' character.

5 Data File Organization

This binary output file that you will create will be organized in the following order:

a) Header: Contains metadata information.

b) Data Buckets: Contains the actual data

Header: The header will contain the following fields:

a) Number of Index Nodes: 4 bytes (long)

b) Number of data buckets: 4 bytes (long)

c) Number of records: 4 bytes (long)

d) Bucket factor: 4 bytes (long)

e) B+ Tree order: 4 bytes (long)

f) Data bucket size in bytes: 4 bytes (long)

g) Index Node size in bytes: 4 bytes (long)

h) Record format: Same as the input file.

Data Buckets:

a) Data buckets will contain a set of records.

b) Remember that each data bucket has the same size regardless of the number of records it contains.

c) Therefore you have to fill the empty records of a bucket with a string of '$' characters which has length of recordSize.

d) The records in the data buckets must be sorted in the order they are present in the input file.

6 Submission Specifications

Submission will be done via COW.

a) You must exactly obey the input/output specifications otherwise your homework will not be graded.

b) Create a directory named as <ID><pa2> such as: 1234567pa2.

c) Copy all of your source files (C, CPP, H) into the directory.

d) Write a Makefile in order to compile your program which will output the pa2.exe after issuing the command 'make'. A simple Makefile is as follows:

```
all: gcc –o pa2.exe pa2.c
```

e) Copy the makefile to the submission directory.

f) Create the archive file by issuing the command:

g) tar –cvf 1234567pa2.tar 1234567pa2

h) Notice that the archive file must be named as <Student_ID><pa2.tar>

i) Submit the pa2 via COW.