DB2

Forum: https://forum-db.informatik.uni-tuebingen.de/c/ss20-db2

Assignment 7 (16.06.2020)

Submission: Tuesday, 23.06.2020, 10:00 AM

Relevant videos: up to `DB2 - Chapter 09 - Video #43`.

https://tinyurl.com/DB2-2020

1. [8 Points] **B$^+$Tree - Insert**

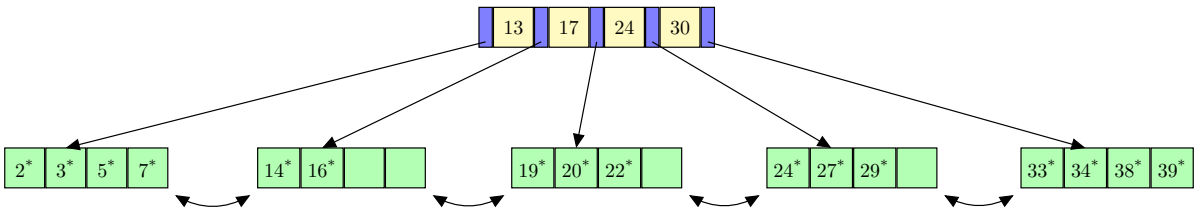

Figure 1: A B$^+$Tree

In this assignment, let us assume that all key values inserted into the B+Tree are unique. For this exercise, further assume the index' `Insert`-operation to **not implement *redistribution***.

Answer the following questions, each based on the unmodified B$^+$Tree of Figure 1:

(a) Identify *four* leaf node entries $(a, \ldots, d)$ which, when inserted one after another (`Insert` $a, \ldots,$ `Insert` d), fill the leaf level pages completely.

(b) How many `Insert` operations are at least needed to grow the size of the tree by two levels?

(c) Take all entries of the leaf level sequence set (2,3,5,...) and `Insert` them, one after another, into a new B$^+$Tree of order $o = 1$. Hand in sketches of the B$^+$Tree instances after the first three `Insert` steps, together with the finally resulting B$^+$Tree.

Note: You may hand in your solutions as plain text, PDF or PNG image file.

2. [12 Points] **B$^+$Tree - Maintenance**

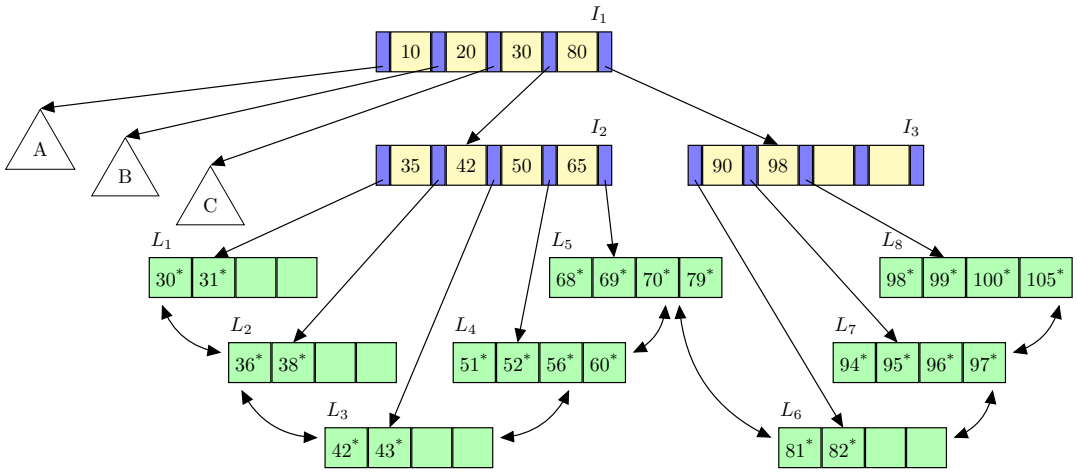

Figure 2: Another B$^+$Tree

Again, let us assume that all key values inserted into the B+Tree are unique. However, for this exercise assume both index operations, `Insert` and `Delete`, to implement ***redistribution* on leaf level**. Remember that redistribution is only performed between direct siblings, i.e. directly linked leaf nodes having the same parent node.

Answer the following questions, each based on the unmodified B$^+$Tree of Figure 2:

(a) Write down all nodes (I_j or L_k) which have to be read by the RDBMS to answer the following queries:

i. "Find all records with a key value greater than 38"

ii. "Delete the record with the key value 81"

(b) Add a record with the key value 104 to the tree. Sketch the resulting tree.

(c) Name all key values which, on insert, will directly cause the tree to increase in depth.

(d) Delete the entry with key value 36 from the tree. Sketch the resulting tree.

(e) The subtrees A, B and C have not been specified completely. Describe any characteristics which can still be inferred about these subtrees.

Note: You may hand in all tree sketches as plain text, PDF or PNG image file.

3. [10 Points] **B$^+$Tree - PostgreSQL**

We provided you with a SQL file `btree.sql` which creates and populates a table

```
indexed (a INT PRIMARY KEY, b TEXT, c NUMERIC(3,2)).
```

Load the file with PostgreSQL and answer the following questions. Please hand in all SQL queries and intermediate results you used to find the final answer.

Note: The tasks require you to use functions `bt_metap(relname TEXT)`, `bt_page_stats(relname TEXT, blkno INT)` and `bt_page_items(relname TEXT, blkno INT)` previously mentioned in the lectures. For more information about these functions, read the documentation at

https://www.postgresql.org/docs/current/pageinspect.html#id-1.11.7.31.6.

(a) How many pages have been created for the index that PostgreSQL automatically created based on the primary key?

(b) Write a query to find the *root node* of the B$^+$Tree. Which page number does it have and what is its and fan-out?

(c) What is the *average* fan-out of all non-leaf nodes?

(d) Use the functions `bt_page_stats(...)` and `bt_page_items(...)` to manually traverse the B$^+$Tree from the root to the index leaf page that holds the key entry for $a = 150\,000$. What are the **minimal** and **maximal key values** found on that leaf node? On which pages of relation `indexed` do the entries of that leaf node point?

Note: Function `bt_page_items(...)` returns a column `data` which represents the key value of an index entry. The `data` value is encoded as a hexadecimal string. For example, the value `'77 97 01 00 00 00 00 00'` is read as the hexadecimal number `0x19777` which can be converted to the decimal number 104311.

For your convenience, we provided you with a function `data_to_numeric(TEXT)` in `btree.sql`, to convert these `data` values to decimal numbers. It can be used as follows:

```
SELECT data_to_numeric('77␣97␣01␣00␣00␣00␣00␣00');

data_to_numeric
-----------------
          104311
(1 row)
```