

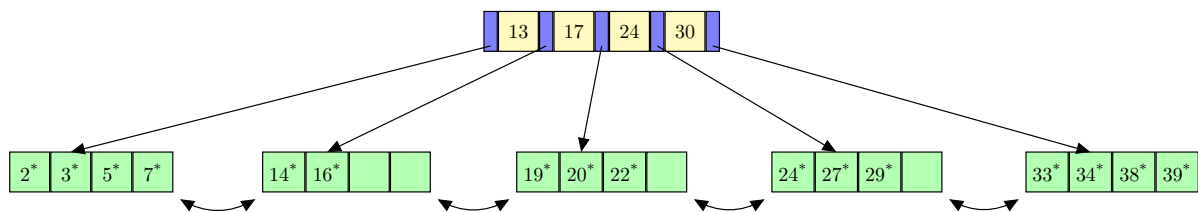
Databasesystems 2

Forum: <https://forum-db.informatik.uni-tuebingen.de/c/ss18-db2>

Assignment 7 (12.06.2018)

Submission: Tuesday, 19.06.2018, 10:00 AM

Please note that students currently have the opportunity to **evaluate lectures**. Please help us to improve **your** courses by providing precious feedback. Check your Mailbox now and participate **today**.

1. [10 Points] B⁺Tree - InsertFigure 1: A B⁺Tree

Consider the B⁺Tree of order $o = 2$ in Figure 1 to be a **UNIQUE-index**¹ that does not allow for duplicate key entries. The index's **Insert**- and **Delete**-operations do **not** implement *redistribution*.

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

- Identify *four* leaf node entries (a, \dots, d) which, when inserted one after another (**Insert** $a, \dots, \text{Insert } d$), fill the leaf level pages completely.
- How many **Insert** operations are at least needed to grow the size of the tree by two levels?
- 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.

2. [10 Points] B⁺Tree - Maintenance

Consider the B⁺Tree of order $o = 2$ in Figure 2 to be a **UNIQUE-index** that does not allow for duplicate key entries. Both operations, **Insert** and **Delete**, **do** implement *redistribution* on leaf as well as inner node levels.

Your task is the following:

- Write down all nodes (I_j or L_k) which have to be read by the RDBMS to answer the following queries:
 - "Find all records with a key value greater than 38"
 - "Delete the record with the key value 81"
- Add a record with the key value 109 to the tree. Sketch the resulting tree.
- Choose a key value to insert which increases the depth of the **original** tree.
- The subtrees A , B and C have not been specified completely. Describe any characteristics which can still be inferred about the subtrees.

¹<https://www.postgresql.org/docs/current/static/indexes-unique.html>

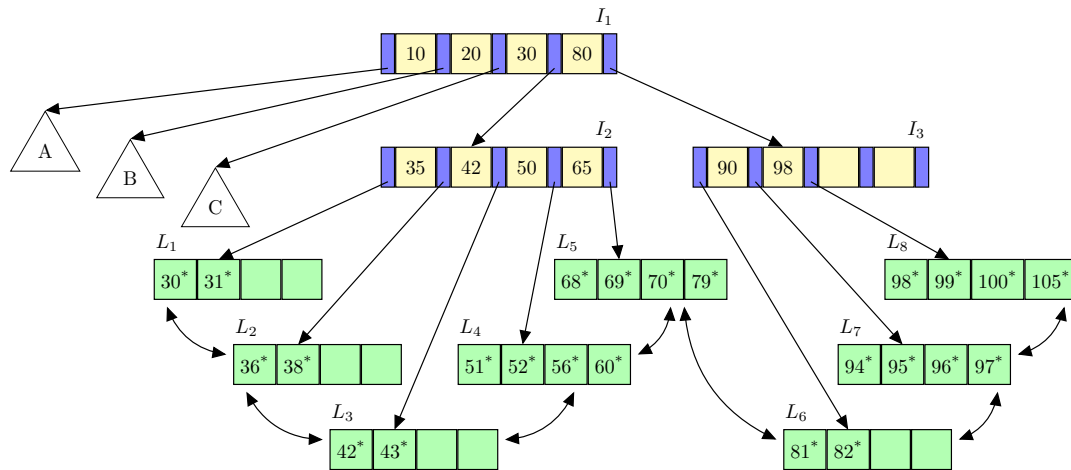


Figure 2: Another B⁺Tree

3. [10 Points] B⁺Tree - PostgreSQL

We provided you with a PostgreSQL query 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.

The tasks require you to use the functions `bt_metap(relname TEXT)`, `bt_page_stats(relname TEXT, blkno INT)` and `bt_page_items(relname TEXT, blkno INT)` previously mentioned in the lecture. For more information about the functions, read the documentation at <https://www.postgresql.org/docs/10/static/pageinspect.html#id-1.11.7.32.5> thoroughly.

- How many pages were created for the index that PostgreSQL automatically creates based on the primary key?
- Write a query to find the root node of the B⁺Tree. What is the page number and fan-out of the root node?
- Write a SQL query to calculate the *average* fan-out of all non-leaf nodes and write down the result.
- Use the functions `bt_page_stats(...)` and `bt_page_items(...)` to manually traverse the B⁺Tree from the root to find the index leaf page that holds the entry for key $a = 150\,000$. Give the minimal and maximal key values found in that leaf node.

The function `bt_page_items(...)` provides you with the `data` column which represents the key value of an index entry. The `data` is represented as a hexadecimal string. For example, the value `'77 97 01 00 00 00 00 00'` is read as the hexadecimal number `0x19777` and thus can be converted to a decimal number 104311.

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

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

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