technische universität dortmund

Architecture & Implementation of DBMS
Summer semester 2020
Prof. Dr. Jens Teubner · DBIS Group

Exercise 3

Released: May 4, 2020 · Discussion: May 11, 2020

1 Storing Records on Pages

1. How can records be stored on pages? Describe the ways to manage pages in a file and discuss their advantages and disadvantages.
2. Describe the structure and concept of record identifiers (rids). How are they used to locate a record?
3. How does a DBMS deal with fields of variable length in a record?

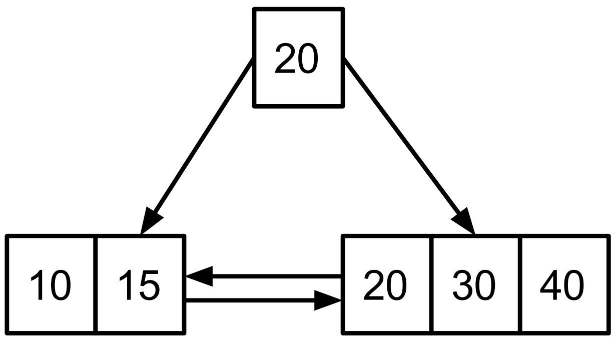
2 Indexing and Clustering

1. What is an index on a file and what is it used for?
2. What kind of indexes do you know? Describe them briefly.
3. What is clustering? Which kind of queries can be sped up by using a clustered index?

3 B$^+$-tree

The major indexing structure in modern DBMS are B^+-trees. Therefore we will examine them more closely in the following assignment.

1. Explain the functioning of B-trees and B$^+$-trees. What is the cost for Searches, Insertions and Deletions?
2. How do B$^+$-trees and B-trees differ?
3. Insert the subsequent keys into the below-mentioned B$^+$-tree of order two. In case of a split two elements shall be added to the left leaf and three into the right leaf, like in the given example. Discuss the effect of inserting presorted data into a B$^+$-tree. Keys: 35, 7, 26, 18, 22, 5, 42, 13, 46, 27, 8, 32, 38, 24, 45, 25.
4. Delete the following keys from the previous tree: 25, 45, 24, 38, 32, 8, 27, 46, 13, 42, 5, 22, 26, 7, 35.



4 Partitioned B-trees

The interplay of lexicographical order and B-trees can be smartly used beyond indexing of tables. For example Goetz Graefe proposed the idea of "Partitioned B-trees"[1][2]and their usage for *bulk updates*.

1. Read the papers and describe the idea of "Partitioned B-trees".
2. What are the benefits of "Partitioned B-trees" and what are the concerns?
3. How do "Partitioned B-trees" achieve faster *bulk insertions* compared to traditional approaches?
4. How do index-reading operations have to be changed to realize "Partitioned B-trees" in a DBMS?

5 B$^+$-Tree Implementation

Discussion with one week later with Exercise 4.
B$^+$-Trees are one of the major indexing structures in modern DBMS. In this execise you have to implement four methods for a given B$^+$-Tree:

- A `locate_leaf`-method for traversing the tree to a leaf node that contains a given key
- A `get`-method for returning the value of a given key
- A `get`-method for returning the value of a given key range
- An `insert_into_leaf`-method that inserts a key-value-pair into a given node

[1]Graefe, G. (2003, January). *Sorting And Indexing With Partitioned B-Trees.* In CIDR (Vol. 3, pp. 5-8).
[2]Goetz Graefe. *"Partitioned B-trees - a user's guide."* In Datenbanksysteme für Business, Technologie und Web (BTW), pages 668–671, Leipzig, Germany, February 2003.

Implementation instructions:

1. Implement the methods in `src/include/index/b_plus_tree.hpp`
2. Run `SELECT * FROM tag WHERE movie_id = 2`. This will take some time
3. Create an index over `tag.movie_id` with `CREATE INDEX tag_movie ON tag(movice_id)`
4. Enable index scan in the `beedb.ini` by setting `enable-index-scan` to 1 or by using the flag `--enable-index-scan`
5. Rerun `SELECT * FROM tag WHERE movice_id = 2`. This should be much faster now