Exercise 4

Released: April 23, 2019 · Discussion:May 6, 2019

1 Block Nested Loops Join vs. Index Nested Loops

Given two relations R and S, where $|R| = 1000$ pages and $|S| = 100\,000$ pages with 20 tuples each. Both relations are stored on a hard drive with seek time of 10ms and a transfer rate of 10000 pages per second.

1. Calculate the time needed to process $R \bowtie S$ using the **Block Nested Loops Join** algorithm. Assume a block size of 100 pages.
2. Assume there is an unclustered index on the joining attribute in relation S. Further assume that the index is cached by the Buffer Manager and that the join $R \bowtie S$ has a selectivity of 1 %. How long does the **Index Nested Loops Join** algorithm take to calculate $R \bowtie S$?

2 Hashing vs. Sorting

Sorting and Hashing share many similarities as stated by Graefe[1].

1. First read the paper mentioned in the footnote. How are sorting and hashing related to each other?
2. How do they differ in performance? Are there workloads where sorting or hashing is better than the other?
3. How can the following operations be efficiently realized using sorting and hashing respectively? Outline a algorithm in pseudocode.
   (a) Duplicate elimination ("`SELECT DISTINCT`")
   (b) `UNION` with eliminating duplicates
   (c) Intersection ("`INTERSECT`")

[1] Graefe, Goetz, Ann Linville, and Leonard D. Shapiro. "Sort vs. Hash revisited". IEEE Transactions on Knowledge and Data Engineering 6.6 (1994): 934-944.

3 Pipelining

Pipelining is a way to reduce the response time and memory footprint of a query. The lecture discussed the Volcano iterator model as a way to implement pipelining in a DBMS.

1. Outline the idea of pipelining and its realization in the Volcano iterator model.
2. How do you have to change your algorithms from the previous assignment to adhere to the Volcano iterator model?

4 B$^+$-Tree Implementation

B$^+$-Trees are one of the major indexing structures in modern DBMS. In this execise you have to implement three methods for a given B$^+$-Tree-Template:

- 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
- An `insert_into_leaf`-method that inserts a key-value-pair into a given node

Download the file `02_b_plus_tree.zip` from the course website[2] and extract it. If you extract it into the folder of the buffer manager project you may have to merge the `CMakeLists.txt` files. You have to complete the file `src/index/b_blus_tree.hpp`

Build instructions:

1. Extract the archive and navigate into the extracted folder.
2. Run `cmake` to create a makefile for your system: `cmake .`
3. Run `make` to create an executable binary file: `make`
4. Execute the created binary file (e.g. `./02_b_plus_tree` on linux)

[2] `http://dbis.cs.tu-dortmund.de/cms/en/teaching/ss19/arch-dbms/exercises/02_b_plus_tree.zip`