Database Systems WS 2020/21
Prof. Dr.-Ing. Sebastian Michel
M.Sc. Angjela Davitkova / M.Sc. Nico Schäfer
Exercise 2: Handout 01.11.2021, Due 08.11.2021 16:00 CET
Lecture content: Videos up to #008

DBIS LAB TU KAISERSLAUTERN
https://dbis.cs.uni-kl.de

For the following questions, we will consider the `uni_db` schema imported in the previous sheet

Question 1: Index tuning (1 P.)

All relevant primary indices and foreign keys are already created (e.g.: *assistants.boss* is a foreign key referencing *professors.PID*).

As the system seems to be very unresponsive lately, the president of the university tasked you with improving the performance of the system. An analysis of the query load showed that the following values are queried regularly:

- Q1: A list of all participants (matrnr) of a specific exam, filtered by semester (e.g.: `WHERE semester < 4`).
- Q2: A histogram, showing how many assistants are working in which field, given a specific boss.
- Q3: The same histogram as before for the whole university.
- Q4: All exams of a given professor with a specific grade.
- Q5: An overview over all grades of a given student.
- Q6: The names of all lectures without prerequisites.

Which indices will you create to improve the performance of the given queries? For each index discuss why you created it. Hint: Maybe some indices can be used for multiple queries. Which of your created indices could benefit from being a hash index, instead of a B+ tree?

Question 2: Index Costs (1 P.)

Given the following values for the `lectures` table: Lets assume, there are $N = 350\,000$ lectures stored in this relation. Each is numbered sequentially between 1 and N. Each page on the disk can store exactly 10 lecture tuples. A B+ tree was created as primary index. It has a height of h and the leaves contain 20 entries, together with a pointer to a data page.

a) How many pages have to be accessed to answer the query `SELECT * FROM lecture WHERE LID BETWEEN 30 000 AND 40 000`? (Hint: Keep the classification of indices from the lecture in mind)

b) After analyzing the query load, the database administrator notices that many queries select lectures held by the same professor. He decides to cluster the table `lectures` by the `heldby` column. An additional index is created on the primary key `LID`. How would the lecture classify this index? How many page accesses would the query from question a) require now?

Question 3: Composite Index (1 P.)

The `lectures` table contains 1 000 lecture tuples, of which exactly 4 tuples fit into one page. Each lecture has a (uniformly distributed) SWS value between 1 and 20. There are 50 professors, each holding the same amount of lectures. The `lectures` table has two secondary composite-key indices on `(sws,heldby)` and `(heldby,sws)`. Both indices are B^+ trees with a height of h and each leaf can store 5 references to pages.

- Which index requires less page accesses for the query `SELECT * FROM lectures WHERE heldby = 5 and sws <=` 10? Please calculate the number of expected page accesses for both indices.

Question 4: Index Implementation (1 P.)

This question requires you to implement code in any programming language you want. The code has to compile and return the correct result. In OLAT, we provide a Java template with most of the boilerplate code already in place. This template checks your result for correctness and times the execution. Do not change anything else than the specified parts of the code.

Submit the code as a separate file (or archive, if you have more than one source file). If you use a different language than Java, provide instructions on how to compile and run your code. Also, your code should then include the same checks as the template Java main method.

Given a dataset of N tuples, consisting of a unique ID and a string value with $|S| = \frac{N}{10}$ distinct strings. Initially the dataset is ordered by the ID and each tuple is assigned a random string from S like the following:

(1, ABC), (2, BCD), (3, ABC), (4, XYZ), (5, XYX), (6, BCD), (7, XYZ)

The system should be able to answer the following queries:

- Return all tuples for which the string is equal to a given query string q.
- Return all tuples for which the string is lexicographically equal or greater to a given query string q.

Your task is to:

a) Implement a default access method, which iterates all tuples and returns the correct results. (Given in the Java template code)

b) Implement a dense index as introduced in the lecture.

If it is not possible for your index implementation to perform one of these operations, then use the default execution method and state why it is not possible.

Execute your code a few times with different query strings (For the java template, just execute it multiple times). Describe how the index creation time and query times for both operations differ in your implementations. You may change the values of N and S to check the effects on your implementation.