Assignment 3: Query optimization

Goal:

In this assignment you will learn to examine query execution plans as well as become familiar with the methods to influence the execution times of queries.

Instruction:

This is an individual assignment. The deliverables to be submitted are specified at the end of this document.

The data files and scripts are located in the **Documents/Data** folder of the course's Oma workspace.

Tasks:

All the tasks can be done with **MariaDB** and **HeidiSQL**. Use the same database that was created in the previous assignment.

First, verify that there is an index for the **`Family_name`** field of the **`Employee`** table. If not, you can create one at this point.

1. Familiarize yourself with the **`SHOW INDEX`** statement with the help of MariaDB documentation (https://mariadb.com/kb/en/mariadb/documentation/). Examine the indexes for the **`Employee`** table by using the statement. Take a screenshot as the answer and explain in detail, how you can verify the existence of an index.

2. Write a two-table query which fetches the name of the selected employee and the name of her/his department (you can specify a suitable **`WHERE`** condition). When the query works, examine the query plan (**`EXPLAIN SELECT`**…). Display your SQL statement and explain the information that the command shows.

3. In the end of a query, you can specify one of the options: **`FORCE INDEX`**, **`USE INDEX`**, or **`IGNORE INDEX`**. Explain the differences between the options. Try each of the options in a query on the **`Employee`** table with a **`WHERE`** condition on **`Family_name`**. For each query, also show the query plan and, based on the plans, explain the effect of the above options on the query implementation.

4. Now, we will examine an example of syntactic optimization of a query. Create an index for the **`Salary`** field of the **`Employee`** table.

   Then, consider a query based on bonus pay. The size of the bonus pay is half of the monthly salary. Execute the following query, which fetches all the employees whose bonus pay is at least 1500 euros:

   ```
   SELECT First_name, Last_name FROM Employee WHERE Salary/2 > 1500;
   ```

Write down the execution time. Check if an index was used (**EXPLAIN SELECT**…). Make a technical change in the query that enables, or even forces the use of the index. What did you do and what is the new execution time? Does the index help?

5. MariaDB allows you to specify the maximum key length in an index. The space reservation for the **Family_name** field is 40 characters, but the index can be built so that, for example, only six first characters of the family name are used in index records. In this case, family names *Rautiainen* and *Rautiala* would end up in the same index record.

   This kind of index is created with statement:

   ```
   CREATE INDEX idx_Family_name(Family_name (6));
   ```

   With the help of **mysqlslap** utility, find out the optimal length of the indexed value, i.e. should your index be based on the whole family name, or, for example, 2, 4, 6 or 8 first characters?

   Technical note: the easiest way to modify an index is to 1) first delete it with a **DROP INDEX** statement, then 2) create a modified index with a **CREATE INDEX** statement.

   Use **mysqlslap** utility whose use is summarized below. Use fixed values for **concurrency** and **iterations** parameters for the whole duration of the experiment.

   ```
   mysqlslap --user=root --password --concurrency=5 --iterations=10
   --create-schema=firma --query=queries.sql
   ```

   The **concurrency** parameter specifies the number of concurrent database connections, and the **iterations** parameter the number of repetitions within each connection. In addition, the command specifies that database **firma** is used, whereas the queries are found in the **queries.sql** file.

6. Now, try logging of slow queries. This happens with the help of system variables. The system variables are shown by the command:

   ```
   show variables;
   ```

   All the system variable names containing the text **slow** will be shown by:

   ```
   show variables like '%slow%';
   ```

   System variables can be set by the **set** command. Set on logging slow (more than 2 seconds) queries:

   ```
   set global slow_queries_log = ON;

   set global slow_launch_time = 2;

   set global slow_query_log_file='<file path> ';
   ```

   Specify the path to get the results in your own file folder instead of MariaDB's system folder.

Then, write a slow query. Which query can you use? What was written in the file for slow queries? How could this kind of logs be used?

Deliverables:

Submit a pdf document with answers to all the above questions. Clarifying screenshots are welcome.