Ryan Farley
DBMS
HW6
11 Dec 24

Extended for vision issues from medical appointment
My computer could not handle tests.
Maximum rows would reach and server kept disconnecting/crashing.
Everything should be syntactically fine besides my computer based issues and problems should have a “correct” answer.

1) Similar to what we did in class, use a stored procedure (generate_accountsLinks to an external site.) to create the accounts table with the following columns:

- **account_num (Primary Key): 5-digit account number**
- **branch_name: The branch name where the account is located.**
- **balance: The balance of the account.**
- **account_type: The type of the account (e.g., Checking, Savings).**

```
delimiter $$
create procedure generate_accounts()
begin
   create table accounts (
      account_num int(5),
      branch_name varchar(255),
      balance decimal(10, 2),
      account_type varchar(50)
   );
end $$
delimiter ;

call generate_accounts();
```

2) For timing analysis, you will need to populate the table with 50,000, 100,000, and 150,000 records.

```
delimiter $$
create procedure populate_accounts(record_count int)
begin
   declare i int;
   set i = 1;
   while i <= record_count do
      insert into accounts (account_num, branch_name, balance, account_type)
```

```
        values (i, concat('branch_', i), 1000.00, 'Checking');
        set i = i + 1;
    end while;
end $$
delimiter ;

-- call populate_accounts(5000);
-- call populate_accounts(100000);
-- call populate_accounts(150000);
```

Computer cannot handle the experiments

3) Create indexes on the branch_name and account_type columns to optimize query performance. You should also experiment with creating indexes on other columns based on your chosen queries (e.g., balance).

```
create index index_branch_name on accounts (branch_name);
create index index_account_type on accounts (account_type);
create index index_account_type_balance on accounts (account_type, balance);
```

4) You will compare point queries and range queries:

Example: Retrieve accounts from the "Downtown" branch with a balance of exactly 50,000.

SELECT count(*) FROM accounts WHERE branch_name = 'Downtown' AND balance = 50000;

Example: Retrieve accounts from the "Downtown" branch with a balance between 10,000 and 50,000.

SELECT count(*) FROM accounts WHERE branch_name = 'Downtown' AND balance BETWEEN 10000 AND 5000

```
select count(*) from accounts where balance = 1000.00;
select count(*) from accounts where balance between 500.00 and 10000.00;
```

5) Experiment with the following dataset sizes: 50K, 100K, 150K
6) For each dataset size, execute both point queries and range queries 10 times and record the execution time for each run.

Perform timing analyses for two point queries of your choice, both with and without the appropriate indexes (e.g., on branch_name, balance, or account_type).
Perform timing analyses for two range queries of your choice, both with and without the appropriate indexes (e.g., on balance, account_type).
Analyze the results of your timing tests and compare how the execution times change when indexes are applied. Discuss any performance differences in your report.
Measure the execution time for each query using TIMESTAMPDIFF(MICROSECOND, start_time, end_time) for high precision.
7) Create a stored procedure to measure average execution times
Create a stored procedure that:
Takes an SQL query as input (as a string).
Executes the query 10 times.
Calculates the total execution time and then computes the average time.
Returns the average execution time in microseconds.
Use PREPARE, EXECUTE, and DEALLOCATE to dynamically run the SQL query multiple times inside the procedure.
8) Summarize the results of the timing experiments
After performing the timing experiments with and without indexes, you are required to summarize the results in a table format.
The table should clearly show the execution times of the queries, both with and without indexes, across different dataset sizes.

```
-- Tasks 5/6/7/8

delimiter $$
create procedure measure_avg_execution_time(query_text varchar(1000))
begin
    declare start_time datetime;
    declare end_time datetime;
    declare total_time bigint;
    declare avg_time bigint;
    declare i int;

    set i = 1;
    set total_time = 0;

-- Execute the query 10 times
    while i <= 10 do
        set start_time = now();
        set @dynamic_query = query_text;
prepare stmt from @dynamic_query;
        execute stmt;
        deallocate prepare stmt;
        set end_time = now();
```

```
  -- Calc exec time and add to total
       set total_time = total_time + TIMESTAMPDIFF(MICROSECOND, start_time, end_time);
   set i = i + 1;
    end while;

     -- Return avg exec time and total exec time
    set avg_time = total_time / 10;
    select avg_time, total_time;
end $$
delimiter ;


-- No index
alter table accounts drop index index_branch_name;
alter table accounts drop index index_account_type;
alter table accounts drop index index_account_type_balance;
call measure_avg_execution_time('select count(*) from accounts where account_type =
"Checking" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name =
"branch_1" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where account_type =
"Checking" and balance between 500.00 and 10000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name =
"branch_1" and balance between 500.00 and 10000.00;');

-- Index
call measure_avg_execution_time('select count(*) from accounts where account_type =
"Checking" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name =
"branch_1" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where account_type =
"Checking" and balance between 500.00 and 10000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name =
"branch_1" and balance between 500.00 and 10000.00;');
```