HW 6

Title: DB Assignment 6

Name: Brennen Cramp

Date: 12/10/2024

I created a procedure to be able to take in a query as a string and return the average time of ten query executions. After this procedure was created, I passed in the first point query with no indices being utilized on the table for 50,000 records and found the average execution time was 21,293.80 µs while passing in the range query had an average of 21,318.50 µs. Analyzing this date leads me to believe that when there are no indices for the smallest amount of records in the table, point and range queries are very similar to each other where the point query executed quicker by 24.7 µs. However, when indices were added to the table on the account_num (primary), branch_name, account_type, and balance (each index was added back to the table for each trials 'With Indexes' runs), the execution times were farther apart based on the query. The point query took 280.00 µs while the range query took 8,020.90 µs, with the point query executing 7,740.9 µs faster than the range query. Here, it can be observed that the point query is more efficient with indices than the range query on average.

Next, the accounts table was truncated and had 100,000 records added along with the indices being dropped from the table. The time_analysis procedure was passed the second point query with no indices being used and it averaged out to 44,461.30 µs for the execution time. The second range query was passed in next where the average execution time was 47,015.70 µs where just like the first queries being executed with no indices, the gap between the two is not very large. Here, the second point query was faster by 2,554.40 µs, greater than when both first queries were executed which was still expected. The indices were then added back and the

second point query executed at an average of 23,826.90 µs, an almost halving of the execution time for the point query when the table had no indices while the range query took 43,667.10 µs. It was noted that the inclusion of indices, while greatly decreasing the execution time for the point query, did not do the same for the range query with only a decrease of 3,348.60 µs. This demonstrates that using a point query with indices is advantageous to the range query with a delta of 19,840.2 µs, a considerable amount indicating that the use of the point query is still the most efficient query on the dataset.

The accounts table was once again truncated but this time had 150,000 records added along with the indices being dropped from the table. The third point query was passed into the time_analysis procedure to produce an average execution time of 67,728.80 µs. However, unlike expectations from previous runs, the third range query executed quicker than the point query with an average execution time of 66,176.60 µs, 1,552.2 µs faster than the point query. Due to this odd occurrence, more trials were conducted for both queries and each time the range query still had the faster average execution time, solidifying the results. After the indices had been added back to the table, the third point query was passed into the procedure to produce (consistently) the fastest time out of all the trials with an average execution time of 121.10 µs. Additionally, this was the greatest time delta with a difference of 67,607.70 µs. The final range query was then tested with an average time of 25,574.30 µs which was a considerable change from the query being executed with no indices on the table, a delta of 40,602.3 µs. Both the runtimes for the point and range queries were unexpected changes due to the previous execution of the second point and range queries having slower average execution times than the third queries. Comparing the two average execution times between both third queries still has the point query prevailing, executing faster than the range query by 25,453.2 µs.

When indices were applied, all average execution times for both the point and range queries ran faster, displaying that it was quicker to find the data needed as opposed to relying on no indices to search the table. When no indices were applied, the average execution times were very close to each other for both the point and range queries. Across the whole trial period, point queries still proved faster than range queries in almost every run except for when the third queries were executed when there were no indices applied to the table. In general, this showcases that when it comes to searching data in a table, aim at using point queries, especially with indices added to the table, to retrieve the desired data for greater efficiency as opposed to using range queries (given what the query is trying to do).

Query Type	Description	Dataset Size	Index Type	Execution Time (Average Microseconds)
Point Query 1	SELECT count(*) FROM accounts WHERE branch_name = 'Downtown' AND balance = 50000;	50,000 records	Without Indexes	21293.8
	Retrieve accounts from the "Downtown" branch with a balance of exactly $50,000.		With Index (e.g. account_num, branch_name, account_type, and balance)	280
Point Query 2	SELECT count(*) FROM accounts WHERE branch_name = 'Redwood' AND account_type = 'Checking';	100,000 records	Without Indexes	44461.3
	Retrieve the number of checking accounts that are from Redwood.		With Index (e.g. account_num, branch_name, account_type, and balance)	23826.9
Point Query 3	SELECT count(*) FROM accounts WHERE account_type = 'Savings' AND balance = 70000;	150,000 records	Without Indexes	67728.8
	Retrieve the number of savings accounts that have a balance of exactly $70,000.		With Index (e.g. account_num, branch_name, account_type, and balance)	121.1
Range Query 1	SELECT count(*) FROM accounts WHERE branch_name = 'Downtown' AND balance BETWEEN 10000 AND 50000;	50,000 records	Without Indexes	21318.5
	Retrieve accounts from the "Downtown" branch with a balance between $10,000 and $50,000.		With Index (e.g. account_num, branch_name, account_type, and balance)	8020.9
Range Query 2	SELECT count(*) FROM accounts WHERE account_type = 'Checking' AND balance BETWEEN 0 AND 30000;	100,000 records	Without Indexes	47015.7
	Retrieve the number of checking accounts that have a balance between $0 and $30,000.		With Index (e.g. account_num, branch_name, account_type, and balance)	43667.1
Range Query 3	SELECT count(*) FROM accounts WHERE account_type = 'Savings' AND branch_name = 'Mianus' AND balance BETWEEN 55000 AND 91000.	150,000 records	Without Indexes	66176.6
	Retrieve the number of savings accounts from the "Mianus" branch that have a balance betweeen $55,000 and $91,000.		With Index (e.g. account_num, branch_name, account_type, and balance)	25574.3

Extra Credit:

Line graph showing the timing results for each query: