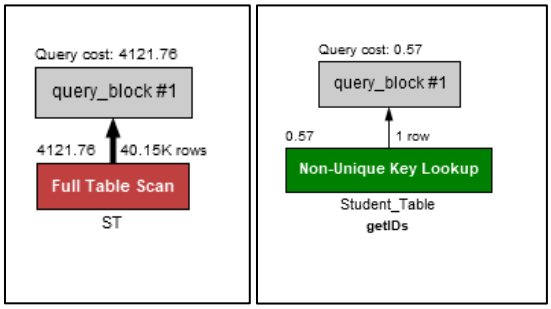
CSCI 4370 Project 3 Report

Team Name: 2019NationalChamps

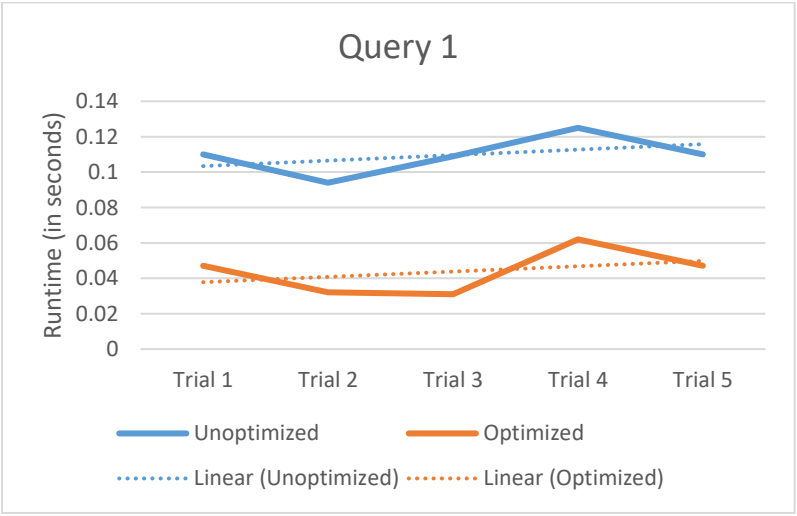
Team Members: Alex Kimbrell, Pravallika Nallamotu, Obediah Blair

Query 1: List the name of the student with id equal to v1 (id)

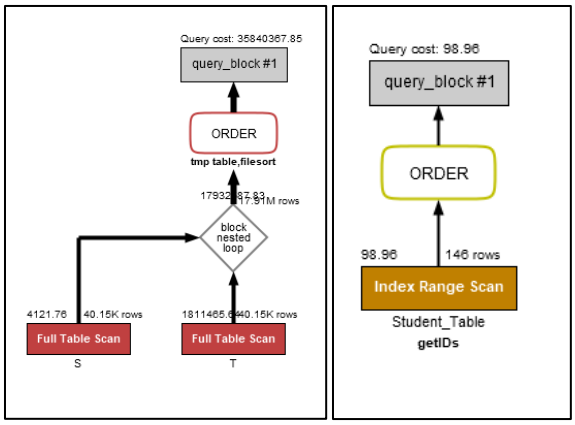


For the first query I initially decided to retrieve the Student ID by utilizing a simple WHERE clause to find the tuple with ID equal to desired ID. However, I was able to optimize this query by creating an index “getIDs” which allowed me to access the ID in approximately 40% of the time it took originally. Although the extra command to create the index took a few seconds, it made a noticeable difference when querying the Student Table.

	Unoptimized	Optimized
Trial 1	0.110	0.047
Trial 2	0.094	0.032
Trial 3	0.109	0.031
Trial 4	0.125	0.062
Trial 5	0.110	0.047
Average Runtime (seconds)	**0.1096**	**0.0438**

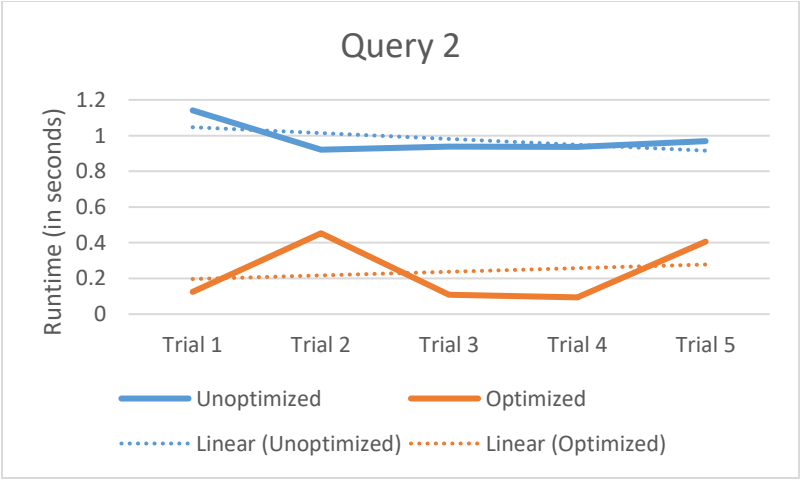


Query 2: List the names of students with id in the range of v2 (id) to v3 (inclusive)



The second query was similar to the first query in that I knew I was dealing with the ID column from the Student Table. My initial query involved two subqueries in order to find IDs above the lower bound and IDs below the upper bound. I then found the intersection of the tuples returned from subquery. My optimized version of Query 2 also utilized the index getIDs. Similar to Query 1, I was able to access the tuples associated with the upper and lower bounding values very quickly which allowed me to shrink my execution time to approximately 24% of my original query.

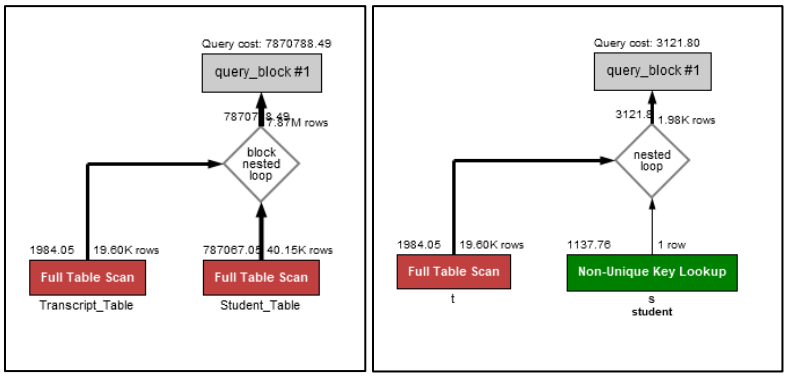
	Unoptimized	Optimized
Trial 1	1.141	0.125
Trial 2	0.921	0.453
Trial 3	0.938	0.109
Trial 4	0.937	0.094
Trial 5	0.969	0.406
Average Runtime (seconds)	**0.9812**	**0.2374**

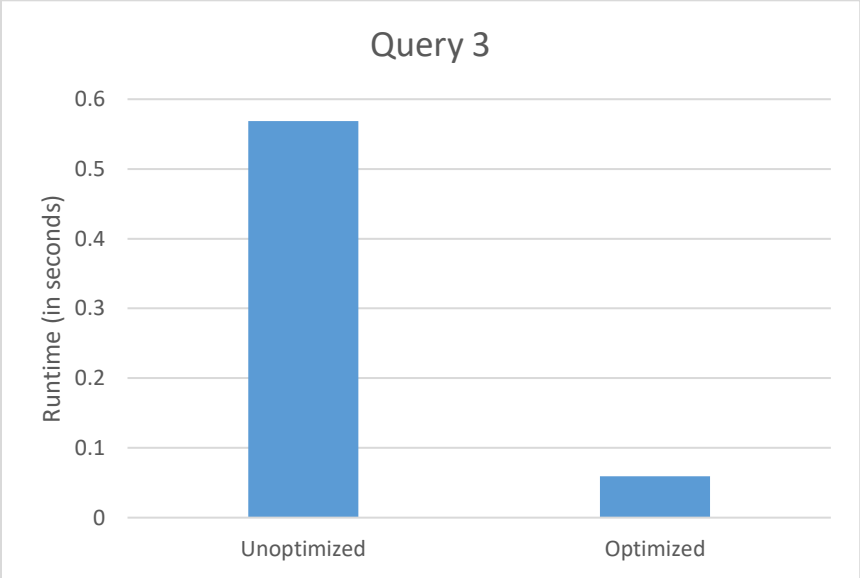


Query 3: List the names of students who have taken course v4 (crsCode).

	Unoptimized	Optimized
Trial 1	0.609	0.078
Trial 2	0.546	0.078
Trial 3	0.594	0.031
Trial 4	0.578	0.047
Trial 5	0.516	0.063
Average Runtime (seconds)	**0.5686**	**0.0594**

In the original SQL query I used a JOIN statement to combine all rows in the Transcript table that contain the v4 course code. For the optimized version I decided to add an index for the student table to make accessing the appropriate IDs better. This index made the JOIN on transcript much more efficient and resulted in a time reduction on 50 milliseconds.

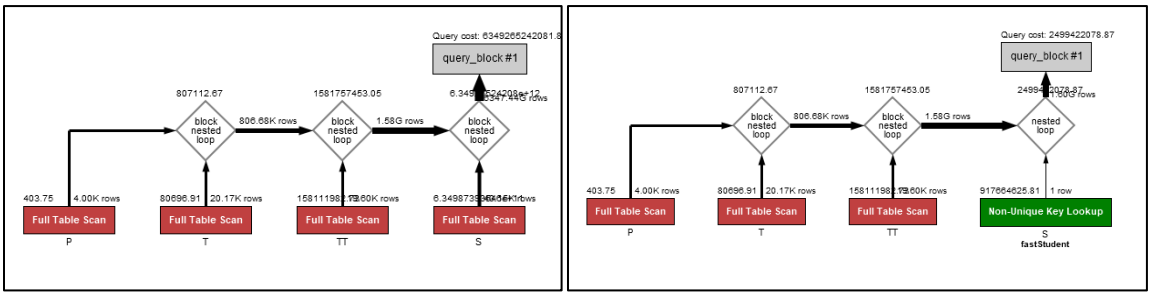


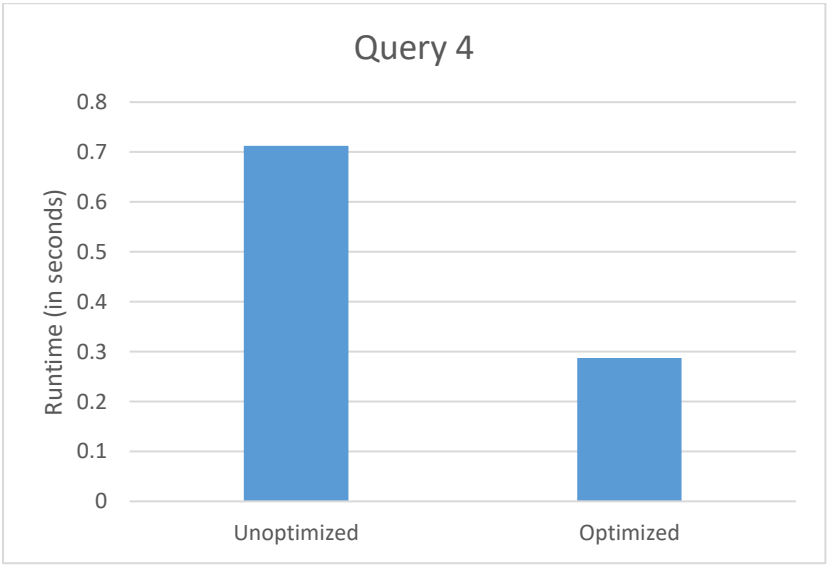


Query 4: List the names of students who have taken a course taught by professor v5 (name).

	Unoptimized	Optimized
Trial 1	0.984	0.297
Trial 2	0.625	0.297
Trial 3	0.719	0.328
Trial 4	0.578	0.281
Trial 5	0.656	0.234
Average Runtime (seconds)	**0.7124**	**0.2874**

In the optimized SQL query, I used several multiple JOINs just like in the original. However, this time I decided to add two Indexes to the student table and transcript table. The index to the student table will make accessing the appropriate students faster for the given professor. The index to the transcript table will also speed up the query since it is a necessary intermediate step to obtaining the professor. By looking at the table above you can see that the query execution time for the unoptimized version was 0.7124 seconds. This was reduced down to 0.2874 seconds by using the student table index. That's a nearly a third the amount of time to execute the original query.

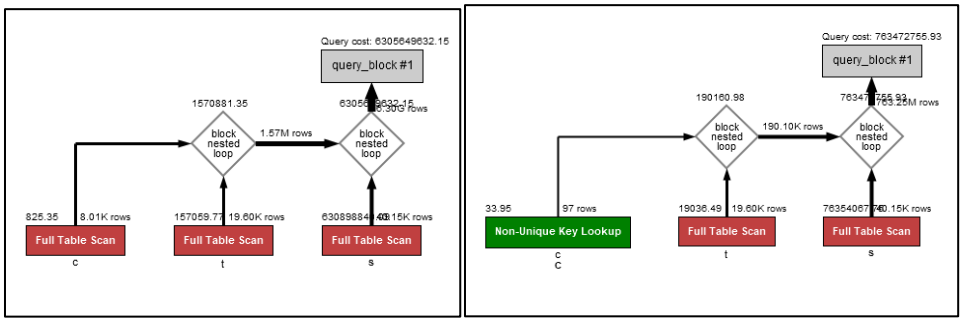


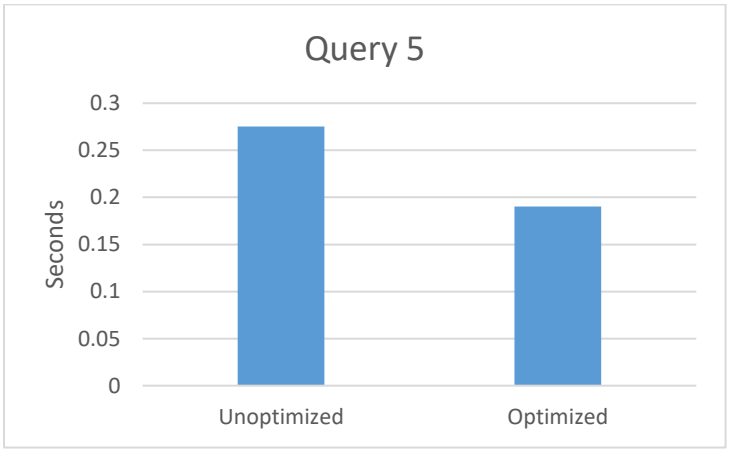


Query 5: List the names of students who have taken a course from department v6 (deptId), but not v7.

	Unoptimized	Optimized
Trial 1	0.422	0.203
Trial 2	0.219	0.188
Trial 3	0.234	0.203
Trial 4	0.235	0.187
Trial 5	0.266	0.171
Average Runtime (seconds)	**0.2752**	**0.1904**

I was able to increase the runtime of the query by removing one unnecessary JOIN and changing the SELECT statement from * to only the name and department ID. I also implemented and an index to make filtering through the Course table a little faster. Overall the runtime was reduced by 18 milliseconds.





Query 6: List the names of students who have taken all courses offered by department v8 (deptId).

	Unoptimized	Optimized
Trial 1	0.281	0.201
Trial 2	0.312	0.177
Trial 3	0.328	0.238
Trial 4	0.297	0.190
Trial 5	0.312	0.203
Average Runtime (seconds)	**0.306**	**0.2018**

Increased time by selecting only the relevant rows to the query and adding an index to help find the correct deptId. This sped up runtime by roughly 10 milliseconds.