Assignment-4 (Analysing Indexes using EXPLAIN)

E Santhosh Kumar (CS16B107)

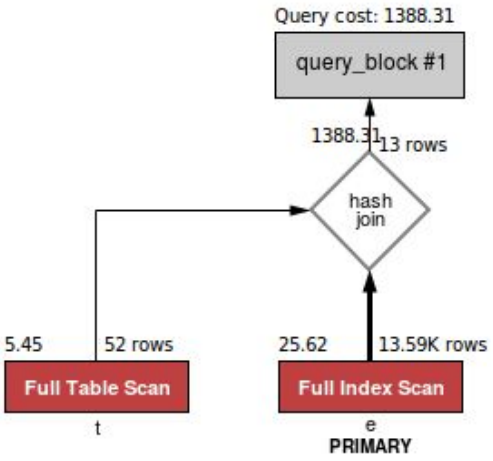
Query Used: To identify the list of students who studied in classroom R5 in year 2003
select e.rollNo from enrollment as e, teaching as t where t.classRoom = 'R5' and e.courseId = t.courseId and e.sem = t.sem and e.year = t.year and t.year = 2003;

Case 1: courseId had been defined as a foreign key in teaching and enrollment. Hence it has an non-unique index by default. We first remove these indexes in both tables (we also have to remove the foreign key constraints to be able to do this) and test the query.

Currently available indexes in the relevant tables are as follows

Table	Index	Non_unique
enrollment	(rollNo, courseId, sem, year) - PRIMARY	False
teaching	(empId, courseId, sem, year) - PRIMARY	False

Execution Plan



Query statistics indicate one expensive full table select_scan and one expensive select_full_join operations. No indexes were used. Total number of rows examined is **13638**

Timing (as measured at client side):
Execution time: 0:00:0.01195693

Timing (as measured by the server):
Execution time: 0:00:0.01179669
Table lock wait time: 0:00:0.00013000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 318
Rows examined: 13638

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 1
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

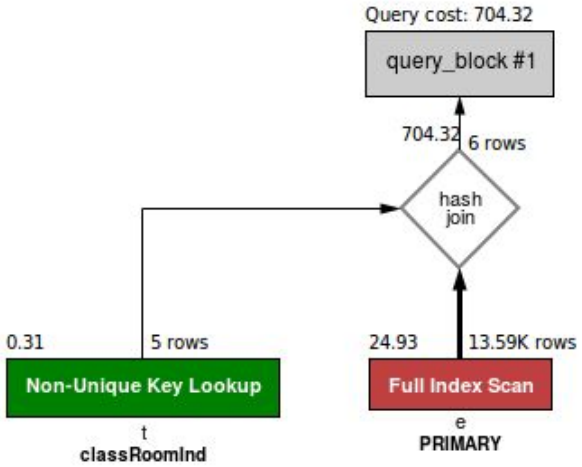
Index Usage:
No Index used

Case 2: We add a non-unique index for classRoom in teaching table, using the below query.
create index classRoomInd on teaching (classRoom);

Currently available indexes in the relevant tables are as follows

Table	Index	Non_unique
enrollment	(rollNo, courseId, sem, year) - PRIMARY	False
teaching	(empId, courseId, sem, year) - PRIMARY	False
teaching	(classRoom) - classRoomInd	True

Execution Plan



The addition of the index for classRoom replaces the full table scan select on teaching table, with a non-unique key lookup. However, since this table is not very large, it only leads to a small reduction in the number of rows examined. Total number of rows examined is still **13591**.

Timing (as measured at client side):
Execution time: 0:00:0.01350808

Timing (as measured by the server):
Execution time: 0:00:0.01325544
Table lock wait time: 0:00:0.00017800

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 318
Rows examined: 13591

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 1
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

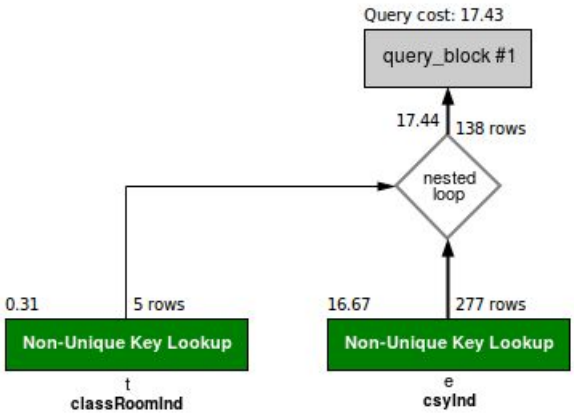
Index Usage:
At least one Index was used

Case 3: We add a non-unique index for the tuple of courseId, sem and year in the enrollment table, using the following query. The columns tracked by this index correspond to an offering of a course in a given semester.
create index csyInd on enrollment (courseId, sem, year);

Currently available indexes in the relevant tables are as follows

Table	Index	Non_unique
enrollment	(rollNo, courseId, sem, year) - PRIMARY	False
enrollment	(courseId, sem, year) - csyInd	True
teaching	(empId, courseId, sem, year) - PRIMARY	False
teaching	(classRoom) - classRoomInd	True

Execution Plan



The addition of this index in the enrollment table leads to significant improvement in efficiency. The full table scan of enrollment table is replaced with a faster non-unique key lookup. Thus there are no more costly full table scans. Total number of rows examined is just **323**.

Execution time: 0:00:0.00384402

Timing (as measured by the server):
Execution time: 0:00:0.00317267
Table lock wait time: 0:00:0.00050900

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 318
Rows examined: 323

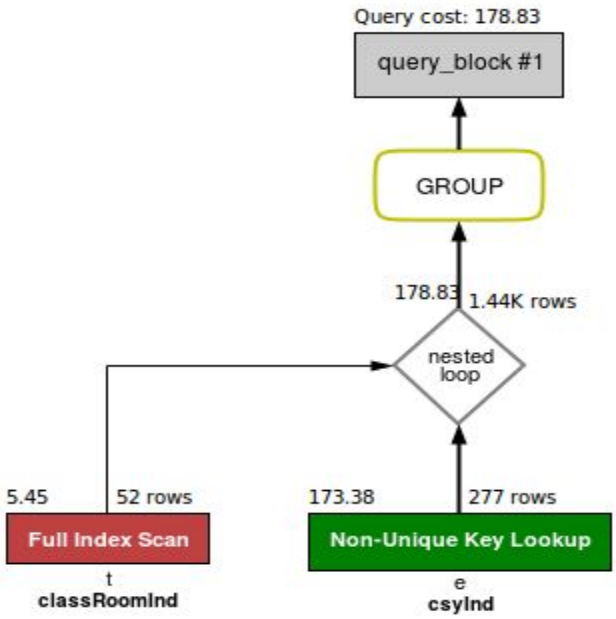
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Case 4: Using the same indexes, we try a slightly more complicated query. **We find the count of students who have studied in each classroom in the year 2003.** (Note: a student who takes up 2 different courses in a classroom in the same year will be counted twice). The query used is
select t.classRoom, count(e.rollNo) from enrollment as e, teaching as t where e.courseId = t.courseId and e.sem = t.sem and e.year = t.year and t.year = 2003 group by t.classRoom;

Execution Plan



Unlike in the previous cases, for this query, the full scan over the teaching table is unavoidable. However, we see that our csyInd index that was added to enrollment table has come in handy even here.

Execution time: 0:00:0.02029204

Timing (as measured by the server):
Execution time: 0:00:0.01991880
Table lock wait time: 0:00:0.00044900

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 12
Rows examined: 3755

Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used