CS 122A: Introduction to Data Management – Winter 2017

Homework 7: Indexing for Performance (100 points)

Due Date: **Friday, Mar 10 (5:00 PM)**

Submission

All HW assignments should contain both your student ID and your name and must be submitted online, formatted in PDF document, through the associated dropbox on EEE. See the table below for the HW submission opportunities. Note that after 5 PM on Thursday no further HW submissions will be accepted. (We will be releasing the solution at that time.) Please strive to get all your work in on time! If possible, try to save the one dropped assignment for the end of the term when you are most likely to want/need it.

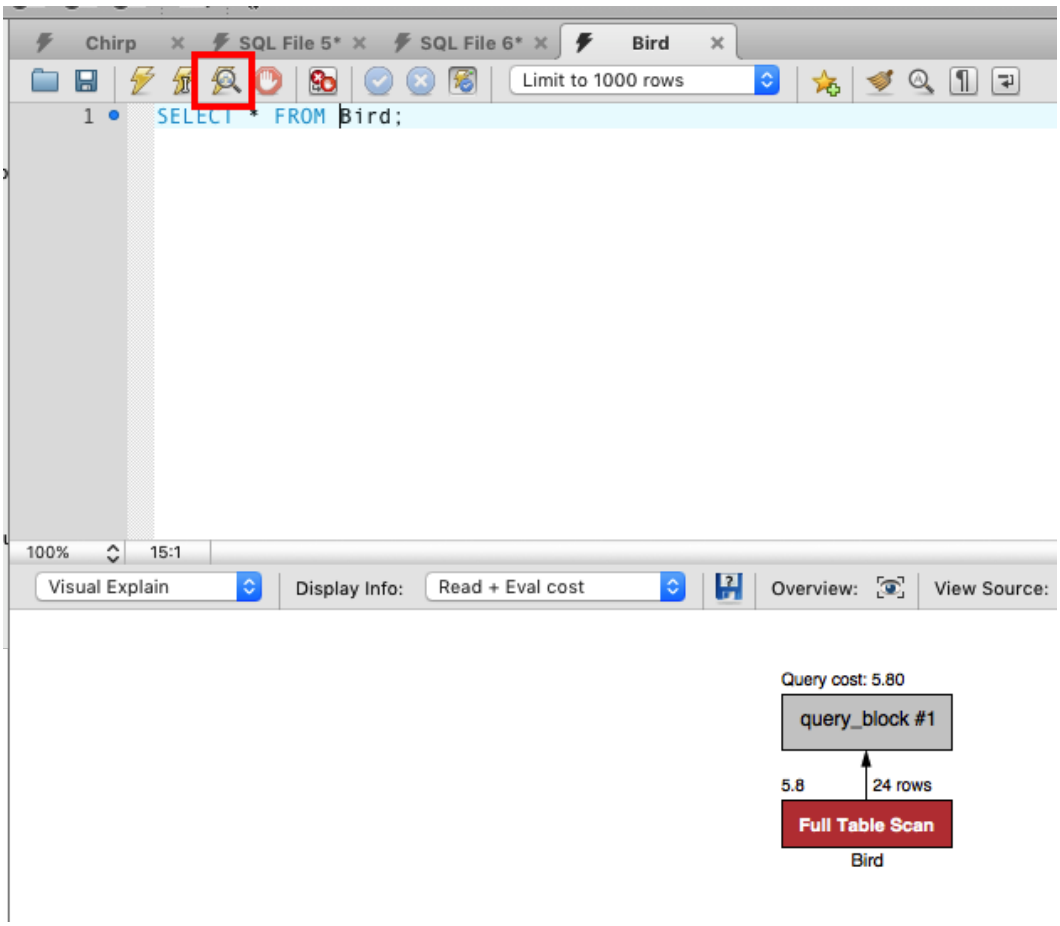
Date / Time	Grade Implications
Friday, Mar 10 (5:00 PM)	Full credit will be available
Saturday, Mar 11 (5:00PM)	20 points will be deducted
Sunday, Mar 12 (5:00 PM)	40 points will be deducted

Indexing for Performance (SQL) [100 pts]

A physical query plan is like a routine that the DBMS follows to assemble the requested query results from the underlying base tables. Different queries will have different physical plans. in fact, the same query may be translated into different physical plans depending on the physical database design. MySQL provides an 'EXPLAIN' function, as shown in the figure below, to help you to check the query plan of your query. You can use this function to examine how is your query will be executed internally, what indexes are being used, and the total cost of your query.

In this homework, we want you to examine the impact of indexing on several queries on the TopicalBirds dataset. Pay close attention to what you see both with and without indexes - this is what you will need to know how to do someday if you end up "doing databases for dollars" and need to performance tune your database application(s)! For example, the figure below shows that the example query is going to be executed using a full table scan - i.e., by iterating over all of the birds - since there's not faster "access path" available. (In this case, that's precisely what the query is asking for - show everything about all birds - so that query isn't further tunable. Others are, however, as you will soon see.)

Note: For this assignment, you can turn in a PDF by cutting/pasting from MySQL into a copy of this document and then PDF-printing the results. (No more pesky scripting this quarter!)



1. To start down the path of exploring physical database designs (indexing) and query plans, start by checking the queries plans for each of the following queries without any indexes using the EXPLAIN function. Report the query plan after each query by taking snapshots and pasting them into your copy of this document.

a) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth';

b) SELECT * FROM Bird WHERE Bird.first_name LIKE '%abeth';

c) SELECT * FROM Bird WHERE Bird.first_name LIKE 'Eliz%';

d) SELECT * FROM Bird WHERE Bird.btag = 'laura43';

e) SELECT * FROM Bird WHERE Bird.birthdate < '1961-1-1';

2. Now create indexes (which are B+ trees, under the hood of MySQL) on the Bird.first_name attribute and Bird.birthdate attribute separately - i.e., create two indexes, one per attribute. Paste your CREATE INDEX statements below.

3. Re-explain the queries in Q1. Report on the query plan after each query, as before. Be sure to look carefully at each plan - think about what you are seeing there.

a) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth';

b) SELECT * FROM Bird WHERE Bird.first_name LIKE '%abeth';

c) SELECT * FROM Bird WHERE Bird.first_name LIKE 'Eliz%';

d) SELECT * FROM Bird WHERE Bird.btag = 'laura43';

e) SELECT * FROM Bird WHERE Bird.birthdate < '1961-1-1';

f) Examine and compare the query plans for queries a-c above - in one brief sentence, what can you observe about B+ tree indexes and their usefulness for equality and different LIKE queries? (Enter your observation below.)

g) Examine and compare the query plans for queries d-e above - in one brief sentence, what can you observe about B+ tree indexes and their usefulness for equality and range queries? (Enter your observation below.)

4. It's time to go one step further and explore the notion of a "composite Index", which is an index that covers several fields together.

a) Check the query plans of following queries. Report the query plan after each query.

i) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth' and Bird.last_name = 'Smith';

ii) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth';

iii) SELECT * FROM Bird WHERE Bird.last_name = 'Smith';

b) Create a composite index on the attributes last_name and first_name (in that order!) of the Bird table. Paste your CREATE INDEX statement below.

c) Re-explain these queries and report the query plan after each query.

i) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth' and Bird.last_name = 'Smith';

ii) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth';

iii) SELECT * FROM Bird WHERE Bird.last_name = 'Smith';

d) Examine and compare the query plans for this composite indexing example. As before, in one brief sentence, what can you observe about composite B+ tree indexes and their applicability to multi-attribute queries? (Enter your observation below.) If you notice anything "familiar", based on things you saw earlier, feel free to mention that too. (:-))

EXTRA CREDIT I [10 points] If you would like to delve even further into the wild world of query plans and indexing, you could try dropping the composite index from Q4 part b (above) and instead creating two separate indexes, one per attribute, and examine the query plans for your queries under that alternative physical database design. If you wish to do so, your task (detailed below) will be to repeat steps Q4 b-d above but with those two indexes instead of the single composite index.

a) Drop the index that you created in Q4 using MySQL's DROP INDEX statement. Paste your DROP INDEX statement below.

b) Create the two separate indexes. Paste your CREATE INDEX statements below.

c) Re-explain the queries in Q4 with the created indexes.

i) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth' and Bird.last_name = 'Smith';

ii) SELECT * FROM Bird WHERE Bird.first_name = 'Elizabeth';

iii) SELECT * FROM Bird WHERE Bird.last_name = 'Smith';

d) Examine and compare the new query plans for these queries with the previous query plans from Q4; briefly summarize your observations below.

EXTRA CREDIT II [5 points] Going back to Q3 e, execute following query **with** the index that you created on birthdate:

SELECT * FROM Bird WHERE Bird.birthdate > '1975-1-1';

Compare the query plan with what you get in Q3 e, and briefly explain your observation.