School of Computer Science, McGill University

COMP-421/764 Database Systems, Winter 2021

Written Assignment 2: Query Evaluation

Due Date March 25, 05:00 pm EST

This is an individual assignment. You are required to work on your own to create the solution.

This assignment is worth 15% of your course grade. The total points in this assignment is 45.

SUBMISSION FORMAT:
Submit your solution as **assignment2.pdf**. Unless stated elsewhere in the question description, no handwritten components are allowed in your solution. Most of your solutions involve writing equations and simple math computations that can be easily typed into a document. Handwritten equations and numbers are often difficult for the TAs to evaluate accurately for which points can be lost. There will be a 25% penalty if this is not followed. Where handwritten components are allowed (relational algbera expression and execution tree), you are allowed to (if you do not want to type or draw) hand write / draw and include the image into the document (or upload as a separate file and include the name in the document). However, all such handwritten submissions must be very legible. This is not an exam, there is sufficient time and therefore, you shouldn't be submitting works that are scrached and smeared all over, which are difficult for the TAs to evaluate. You maybe familiar with your hand writing, others are not. If TAs have a hard time to read your "script", you will not get points.

For each day late, 15% of the maximal achievable points will be subtracted from the achieved points. Maximum of 2 late days allowed.

In this assignment we evaluate some queries used by an application of a healtcare network. We look at the three relations whose schema and a sample record is given below:

Patient (hcnum:CHAR(12), first_name: VARCHAR(30), last_name: VARCHAR(30), email:VARCHAR(42), phonenum:CHAR(11))
'CARNE98063452', 'Emily', 'Carnel', 'ecarnel@someemail.com', '19877654409'

Doctor (practiceid:CHAR(10), name:VARCHAR(48), startdate:DATE, speciality:VARCHAR(33))
MEGJ781232, 'Jessy Meghan', '2018-08-21', 'Oncology'

Diagnosis (diagid:INT, notes:VARCHAR(300), diagdate:DATE, followup:BOOLEAN, followupdate:DATE, hcnum:CHAR(12), practiceid:CHAR(10)) hcnum references Patient, practiceid references Doctor
68122, 'Need to perform biopsy...', '2020-02-22', TRUE, '2020-04-20', 'CARNE98063452', 'MEGJ781232'

INT occupies 8 bytes and each CHAR is 1 byte (that is, a CHAR(10) would occupy 10 bytes). Assume for all VARCHAR attributes that the average size of a value is 2/3 the total capacity allocated for that attribute (e.g speciality is on an average 22 characters). BOOLEAN is 1 byte and DATE occupies 10 bytes.

There are 30,000 patients and 200 doctors. Assume that each patient on an average has had 10 diagnosis. The database has 2 full years (365 × 2 days) of diagnosis stored for the years 2019 and 2020 (we will pretend 2020 is not a leap year for simplicity), and diagnosis are equally likely to have been made on any of the days in a year (another simplication to avoid handling weekends and holiday). There are 25 different speciality for doctors (assume uniform distribution). We will also assume that the diagnosis records can be evenly distributed between the doctors in the network. About 80% of diagnosis ends up with a followup visit required. The followup date is set to NULL if there is no followup visit scheduled. This is the only attribute in the schema that can be NULL.

In general, all database pages are 4000 bytes with an average 75% fill factor. For indexes, a single data entry may be spread over more than one leaf page. Each rid has 10 bytes and each internal pointer has 8 bytes. Leaf/data pages are filled on an average of 75%. An index page has 4000 bytes. The root may have any fill factor. Assume that the root and all intermediate nodes of an index are in memory. For all questions, assume 100 free buffer pages in memory. If it is convenient, you may assume a couple more for simplifying your calculations. Minor rounding errors are acceptable. **However, keep in mind that it is important you write down the correct steps. If you got a close enough solution by chance eventhough your steps and equations are wrong, you will still lose points**.

Do not assume any indexes other than the ones mentioned for the specific questions. (Index mentioned in one question does not automatically carry over to the next one). You can also assume that the root and intermediate nodes of an index is already available in memory (not counted towards our free memory pool).

Tip:- Most of the questions below will require the number of records for the tables in our model. The information needed to compute these is given in the description above.

Ex. 1 — (11 Points)

A typical query would be the one below, which searches the `Diagnosis` table for all the diagnosis written by Dr. `D`, with a diagnosis date between `X` and `Y`. Here `D`, `X` and `Y` represent parameters that might differ for each execution. `X` and `Y` can be any date in the years `2019` and `2020`. We also assume `Y` > `X`.

```
SELECT *
FROM Diagnosis
WHERE practiceid = D AND diagdate BETWEEN X AND Y
```

(a)(3 Points) Find the number of data pages to store each of the 3 tables.

(b)What is the I/O cost of the above query when:

(i)(1 Points) You do not use any index.

(ii)(3 Points) You use an unclustered type I index on practiceid?

(iii)(4 Points) You use a clustered type II index on diagdate?

Turn in:- Your solution (typed) in **assignment2.pdf**, make sure to write down each sub question number, etc.
Note:- To simplify your calculations, you can assume basic date arithmetic, i.e, for example, if you do ′2019−01−31′ – ′2019 − 01 − 01′ you get 30 (number of days between them) as your result.
In some cases you maybe able to compute a "concrete" number as the solution, in other cases it will be an equation that depends on the actual values used in the query.

Ex. 2 — (0 Points)
Note:- This is a warmup problem, please do not turn in the solution to this or cross reference solution steps in this problem. This will not be graded.

Given an unclustered type II index on `hcnum` on `Diagnosis`, calculate the estimate I/O and give an estimate of the number of output tuples for:

(a)index nested loop join between `Diagnosis` and `Patient`.

(b)block nested loop join between `Diagnosis` and `Patient` and `Patient` is the outer relation.

(c)sort merge join between `Diagnosis` and `Patient` (assume relations are not already sorted on the join attribute). Hint:- Remember we used a technique in class to reduce the cost of merge-join so that it is less than the sum of the individual costs of merge sort and join.

Ex. 3 — (19 Points)

Consider the following query:

```
SELECT P.phonenum, D.followupdate, Doc.practiceid, Doc.name
FROM Patient P, Diagnosis D, Doctor Doc
WHERE P.hcnum = D.hcnum
AND D.practiceid = Doc.practiceid
AND D.diagdate >= '2020-12-01'
AND Doc.speciality = 'Oncology'
```

Referring to the `Patient` table as `P`, the `Diagnosis` table as `D`, and the `Doctor` table as `Doc`, a non-optimized relational expression for this query is:

$\rho(J(phcnum, phonenum, dhcnum, dpracticeid, diagdate, followupdate, docpracticeid, speciality, name),$
$\Pi_{P.hcnum, P.phonenum, D.hcnum, D.practiceid, D.diagdate, D.followupdate, Doc.practiceid, Doc.speciality, Doc.name}(P \times D \times Doc))$

$\Pi_{phonenum, followupdate, docpracticeid, name}($
$\sigma_{phcnum=dhcnum \wedge dpracticeid=docpracticeid \wedge diagdate>='2020-12-01' \wedge speciality='Oncology'}(J))$

(a)(3 Points) Optimise this expression according to the rules of **algebraic optimisation** discussed in class. Your answer need not take into account data cardinalities for this sub-question, and you need not draw a tree.

Turn in:- You may type this solution / include this as a screen shot, or a scan/photo of handwritten relational algebra expression.

(b)(16 Points)

Given an unclustered Type I index on `hcnum` on the `Patient` table, give an execution plan for your expression and indicate how to execute each operator. Draw the execution plan tree. You need not include the number of rows and bytes passing through the operators in the tree. Give a rough estimation of the best I/O cost for your execution plan. Do not assume any other indexes on the tables.

Turn in:- The steps, discussion and any computations must be typed in the **assignment2.pdf**. For the execution plan tree, you may draw / include as a screen shot, or a scan/photo of hand-drawn execution plan tree.

Note:- Remember to apply the techniques we saw in class such as pipelining and "combining" operations (projection with join, etc) when useful, to reduce the overall I/O cost.

Ex. 4 — (15 Points)

Consider the following query that finds the patients who have been diagnosed by doctors from more than one speciality since the beginning of the year 2020.

```
SELECT P.hcnum, COUNT(DISTINCT speciality)
FROM Diagnosis D, Doctor Doc
WHERE D.practiceid = Doc.practiceid
AND D.diagdate >= '2020-01-01'
GROUP BY P.hcnum
HAVING COUNT(DISTINCT speciality) > 1
```

Find the optimal execution plan and its cost. Also draw the execution plan tree.
Turn in:- The steps, discussion and any computations must be typed in the **assignment2.pdf**. For the execution plan tree, you may draw / include as a screen shot, or a scan/photo of hand-drawn execution plan tree.

Hints & Guidelines

- You do not have to "budget" for slot directory, sibling pointers, row header, etc., when doing your page size calculations - only data entries, index entries and record sizes need to be considered.
- Assume that a NULL value takes no extra space in the data record (table, intermediate results, etc.).
- To simplify the computation of index logistics, you can assume that the size of NULL values stored in the index is the same as the size of regular values of that column.
- Assume uniform distribution of values, unless the information given indicates otherwise.
- You can use the information (record length, pages, etc.) computed in one Exercise (except Ex.2, which is not graded), in the following exercises.
- Use pipelining when possible to save I/O cost between multiple steps (operators) in the execution plan. Not doing this could result in points lost (depending on how straightforward it would have been to integrate).

- When calculating I/O, take into account the I/O cost savings due to pipelining in your equations.
- You can leave fractions as it is, if you would like to (as in number of data entries per page, records per page, etc.). However, keep in mind that there are some steps that cannot produce fractions. For example, if you have a specific block nested join step in your execution plan, its cost cannot be 24.6 pages, it has to be 27 pages - because that is the unit in which we do I/O (same with memory buffers).
- Minor rounding errors due to approximating data entries per page, records per page, etc., are acceptable. In general look at the magnitudes of the numbers you are rounding and how big a number you are multiplying it with later to see how significant the impact is. (for example, rounding 1.5 to 2 and rounding 1000.5 to 1001 introduces very different magnitude shifts to the numbers with which they are multiplied later). At the end of the day, remember that our objective is to have a reasonably good computational methodology to help us choose between multiple possible execution plans based on their costs.
- Partial points are given to steps and methodology even if you making errors doing math. But try to use a calculator to do your math to avoid making mistakes.

Questions ?

Please use Piazza for any clarifications you need. Do not email the instructor or TAs as this leads to a lot of duplicate questions and responses (not an efficient system). Please check the pinned post “A2 general clarifications” before you post a new question. It might have been already addressed there, in which case we will not address it again.

There will be specific TA office hours for the assignment that will be announced closer to the due date.