CS 411: Database Systems

Fall 2021

Homework 4 (Due by 23:59 CT on November 12th)

Logistics

1. This homework is due on November 12th at 23:59 CT. **We DO NOT accept late homework submissions.**
2. You will be using Gradescope to submit your solutions. Answer each sub-question (e.g. "a.") on a **new page** and submit your solution as a single PDF file on Gradescope. **IMPORTANT: Please make sure to link PDF pages with the corresponding question outline on GradeScope.**
3. Please write down any **intermediate steps** to receive full credit.
4. Keep your solutions brief and clear.
5. Please use Campuswire if you have questions about the homework but do not post answers. Feel free to come to office hours.

Q1. Relational Algebra Queries (20 points)

Consider a relational database of a Medical College/Hospital system. The corresponding relational schema is described below:

Doctor (DoctorID, Name, DepartmentID, Phone, City, salary)
Patient (PatientID, Patient_Name, Gender, Phone, City, dob)
Appointments(Date, PatientID, DoctorID)
Courses(CourseID, Course_Name,DoctorID, DepartmentID)
Department (DepartmentID, Dept_Name, Building)

Please answer the following queries using **relational algebra expressions**: (do not use expression tree).

You can solve it in steps and combine your answers. For example,

R1: $A \bowtie_{A.abc=B.abc} B$
R2: $R1 \bowtie C$
R3: $\Pi_x(\sigma_{x=y} R2)$

Note: You can use Relax: Relational Algebra Calculator to try your queries: https://dbis-uibk.github.io/relax/calc/local/uibk/local/0. But please note that Relax is not meant for grading your answers. It is a tool that can help you write RA queries and visualize your answer.

We have provided the schema and instance of Question 1 here: https://docs.google.com/document/d/1LfRoVz373b_irQQzaFwB71igvModsW9uIcpXik4ecl0

You can paste the above script into 'Group Editor' and execute your queries in the 'Relational algebra' section to test your RA queries. Please watch this short tutorial (made by Abdu) on how to load the data and use Relax to write RA queries.

1. Find the names of patients who have not yet made an appointment. (5 points)
2. Find the names of all the departments that had zero male patients on 10/28/2021. Please note that the patient belongs to the same department as that of the doctor assigned to them. Ignore departments that did not have any patients on that day. (5 points)
3. Find all the dates when there was at least one appointment having either a patient from Nashville or a doctor who has never taught a course. (10 points)

Q2. Relational Algebra Equivalence (20 points)

Consider the following supermarket database schema, with primary key(s) underlined:

Doctor (DoctorID, Name, DepartmentID, Phone, City, salary)
Patient (PatientID, Patient_Name, Gender, Phone, City, dob)
Appointments(Date, PatientID, DoctorID)
Courses(CourseID, Course_Name,DoctorID, DepartmentID)
Department (DepartmentID, Dept_Name, Building)

For each pair of relational algebra expressions (E1 and E2) shown below, state whether or not the two expressions are equivalent in general. If your answer is "true", justify the equivalency by explaining which RA rule(s) can be used to transform one query expression into the other. If your answer is "false", give a sample instance of the database where the two expressions are not equal. Make no assumptions about the existence of foreign keys.

1. E1: $\pi_{\text{Doctor.Name, Doctor.DepartmentID}}(\text{Doctor} \bowtie_{\text{Doctor.DepartmentID = Courses.DepartmentID}} \text{Courses})$

   E2: $\boldsymbol{\sigma}_{\text{Doctor.DepartmentID == Courses.DepartmentID}}(\pi_{\text{Doctor.Name, Doctor.DepartmentID}}(\text{Doctor}) \times \pi_{\text{Courses.DepartmentID}}(\text{Courses}))$ (5 points)

2. E1: $(\pi_{\text{PatientID}}(\boldsymbol{\sigma}_{\text{Patient.City == 'Chicago'}}\text{Patient}) \bowtie \text{Appointments}) \cup (\text{Appointment} \bowtie \pi_{\text{DoctorID}}(\boldsymbol{\sigma}_{\text{Doctor.City = 'Nashville'}}\text{Doctor}))$

   E2: $\pi_{\text{Appointments.PatientID, Appointments.DoctorID,Appointments.Date}}((\boldsymbol{\sigma}_{\text{Patient.City == 'Chicago'}}\text{Patient} \bowtie \text{Appointments}) \bowtie \boldsymbol{\sigma}_{\text{Doctor.City == 'Nashville'}}\text{Doctor})$ (5 points)

3. E1: $\pi_{\text{DoctorID}}(\boldsymbol{\sigma}_{\text{Doctor.City = 'Nashville'}}\text{Doctor}) - \pi_{\text{Doctor.DoctorID}}(\text{Doctor} \bowtie_{\text{Doctor.DoctorID = Courses.DoctorID}} \text{Courses})$

   E2: $\pi_{\text{DoctorID}}\text{Doctor} - \pi_{\text{DoctorID}}(\boldsymbol{\sigma}_{\text{Doctor.City != 'Nashville'}}\text{Doctor}) - \pi_{\text{DoctorID}}((\boldsymbol{\sigma}_{\text{Doctor.City = 'Nashville'}}\text{Doctor}) \bowtie_{\text{Doctor.DoctorID = Courses.DoctorID}} \text{Courses})$ (10 points)

Q3. Physical Operators

In this problem, you may choose between two options, both of which will provide you with full credit.

If you **choose to do the MP option** and implement it successfully, you will **receive 10 extra credit** points.

You should not submit solutions for both options, just do one.

Option 1: Implement a Physical Operator (10 points + 10 EC = 20 points)

Implement the one-pass, sort-based INTERSECTION physical operator using Python. A skeleton code file, along with test cases, can be found here.

The **intersection** operator should combine tables with the same columns (order of columns and names of columns must be the same), as described in the lecture notes and textbook. I/O operations are handled by our code skeleton, you are only responsible for the Set INTERSECTION operator.

For each test case provided, we have three input files and one expected output file which can be used to check the correctness of your output. You can use **diff** to compare your output and the expected output:

- **config.txt**: define memory size and block size.
- **table1.csv**: input table 1
- **table2.csv**: input table 2
- **expectedOutput.csv**: the results of **table1 INTERSECTION table2**.

Output Format:

1. If the intersection cannot be carried out under the constraints outlined in **config.txt**, please write "INVALID MEMORY" or ”INVALID SCHEMA/INPUT” (as shown in the skeleton) to the first cell in the output file.
2. Please include the column names and the data of the resulting table. The rows in the output table should maintain the same relative orders as the bigger input table.

Submission:

Please submit your code as a submission for “Homework 4 Q3 Option 1” **(TODO: Attach link)** on Gradescope, as a script called **mp.py**. We will run your code on each of the provided test cases, along with a number of held-out test inputs in order to verify correctness.

Option 2: Two-Pass "Multi-Way" Merge Sort (10 points)

Consider the following relation R with 12 blocks of data stored on the disk (B(R) = 12):

[63, 33, 89] [98, 21, 10] [78, 89, 51] [80, 100, 98] [66, 18, 36] [17, 80, 67]

[42, 5, 19] [34, 22, 52] [7, 96, 98] [10, 76, 88] [31, 35, 7] [8, 13, 88]

Each block holds 3 values and there are 4 blocks of memory available for the operation (M = 4). Use Two-pass "multi-way" merge sort algorithm to sort the blocks above.

Solution Format:

- Please write one memory update per line and don't leave empty lines.

- First, write sorted runs and then merge sorted runs.
- For sorted runs, write as: [1,3][5,4][7,9]=>[1,3][4,5][7,9]
- For every merge run, mark in bold the position of pointers in each block at the beginning of the run. Eg: [**1**,3][**2**,4]=>[1,2]=>out
- If two slots have the same number, choose the leftmost first by default.
  For example: [***2***,3][**2**,4]=>[***2***,**2**]=>out
- If the output block cannot be outputted and needs a memory reload, write as [1,**2**][**2**,4]=>[2,_]

Q4. Block-based Nested Loop Join (20 points)

The following three tables from a recruiting database of a Company are stored on the disk:

Candidates (CandidateID , FirstName, LastName, PhoneNumber)
WorkExperience (CandidateID, CompanyID, Role, Years)
Companies (CompanyID, CompanyName, StateName, CountryName)

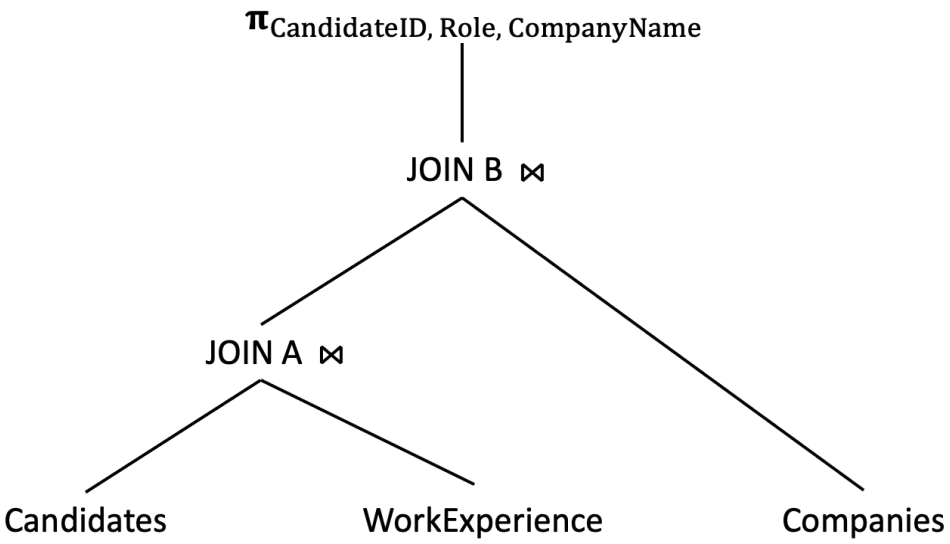
The statistics of these tables are shown below:

Table	Total Tuples	Total Occupied Blocks
Candidates	3000	40
WorkExperience	5000	70
Companies	200	30

Consider the following query:

```
SELECT CandidateID, Role, CompanyName
FROM Candidates, WorkExperience, Companies
WHERE Candidates.CandidateID=WorkExperience.CandidateID
    AND WorkExperience.CompanyID=Companies.CompanyID
```

The Joining strategy is as follows:



Assume the number of blocks in memory is **M = 20**. Answer the following questions:

(i) Calculate the cost to perform block-based nested loop join in "Join A". You can choose either **Candidates** or **WorkExperience** as the outer relation for your computation. Please clearly state your choice in the solution.

(ii) Assume that after "Join A", the joining result would be stored back to the disk first and the resulting table has 8000 tuples. Each block can contain up to 50 tuples in the resulting joined relation. Assume that the tuples are densely stored. Calculate the estimated cost to perform a block-based nested loop join in "Join B". You can choose either Universities or the 'JOIN A' relation as the outer relation for JOIN B. Please clearly state the order in which you joined Universities and the 'JOIN A' relation in your solution. [Hint: You need to compute the total number of blocks for "Join A" to be able to estimate "Join B"] (10 points)

Q5. Cost-based Optimization (15 points)

Note: On your exam, the auto-grader only rounds (not ceiling/floor) the final result and not the intermediary results. For this question, follow the same process, and do not round intermediate results.

Consider the following relations:

A(x,y,z)	**B(w,x)**	**C(w,x,z)**	**D(y,z)**
T(A) = 2000	T(B) = 4000	T(C) = 6000	T(D) = 5000
	V(B, w) = 100	V(C, w) = 30	
V(A, x) = 50	V(B, x) = 20	V(C, x) = 50	
V(A, y) = 20			V(D, y) = 50
V(A, z) = 40		V(C, z) = 20	V(D, z) = 20

Determine the most efficient way to join the 4 relations, i.e., A⋈B⋈C⋈D. Note that T(R) is the number of tuples in relation R and V(R, a) is the number of distinct values of attribute a in relation R.

Assume the distribution of values of each attribute in all relations are uniform and both Containment Of Values and Preservation Of Value Sets hold in the relations above. Show your work by completing the following table. Each step in the dynamic programming algorithm should be one row.

Subquery	**Size**	**Cost**	**Plan**

Q6. Physical Query Plan (15 points)

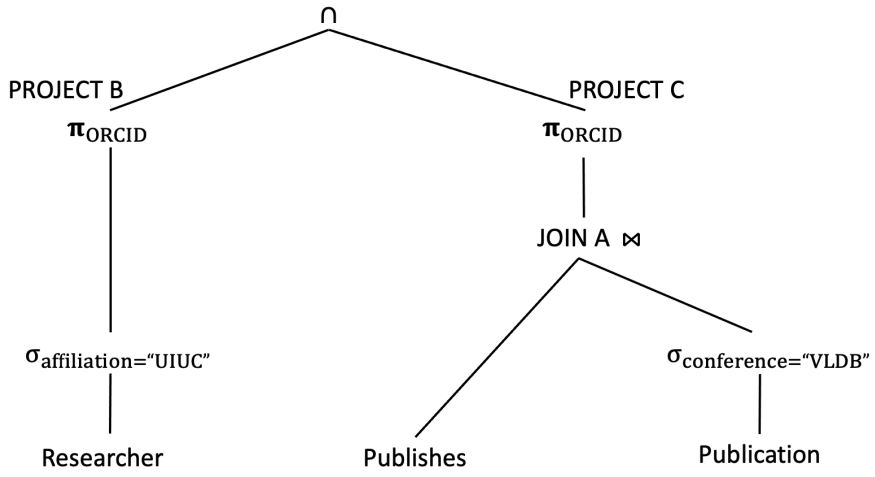
Consider the following schema for a bibliographic library (such as DBLP). Assume we have the following statistics for the tables and columns with the primary key(s) underlined:

Researcher (ORCID, name, affiliation, email)
Publishes (ORCID, DOI, year)
Publication (DOI, title, conference)

Select the ORCIDs of researchers who are affiliated with 'UIUC' and have published at least one publication in conference 'VLDB'.

Table	**# tuples**	**# blocks**
Publishes	200000	2500
Publication	15000	1500
Researcher	5000	100

Column	**# distinct**
Researcher.Affiliation	10
Publication.Conference	30
Publishes.DOI	10000



Assume that the Researcher, Publishes, and Publication tables have a clustered index on their respective primary keys. Assume the distribution of Affiliation and Conference values are uniform, to compute estimated sizes after selection.

(a) What is the cost and size of the selection conference = “VLDB”? Assume we have an unclustered index on “conference”. (2 points)

(b) What is the cost and size of executing selection affiliation = “UIUC”? (2 points)

(c) What is the cost of executing Join A if we use an index-based nested loop join? Follow other assumptions from (a). Use the smaller relation as the inner relation. (3 points)

(d) What is the cost of executing Join A if we use a two-pass hash-based join? Follow other assumptions from (a). (3 points)

(e) Assuming the block size for the result table, after projection B/C (Use part d for Join A) is 100 tuples and memory capacity is 50 blocks, can we perform a one-pass algorithm for the intersection operation? If yes, estimate the cost of intersection. If not, explain why and estimate the total cost of the intersection with the appropriate algorithm. (Hint: First, estimate the size of result tables after projections B and C). (5 points)