CS 411: Database Systems

Summer 2021

Homework 4 (Due by 23:59 CT on July 28th)

Logistics

1. This homework is due on July 28th 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.

Section 1. Relational Algebra

Q1. Relational Algebra Queries (20 pts)

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(AptID, 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.

(a) Find names of patients who consulted doctors from both 'Cardio' and 'Ortho' departments. (5 points).

(b) Find the PatientID of all patients who were treated by a doctor who teaches **at least** one course in a department that is different from the department he works in. (10 points)

(c) Find the Name and Department of all Doctors who **do not** teach any course.(10 points)

Q2. Relational Algebra Equivalence (21 points)

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

```
Customer (CustomerID, FirstName, LastName, Location, BirthDate, BirthCountry)
Item (ItemID, ItemName, Price, Category, Brand, ExpirationDate)
Purchase ( CustomerID, ItemID, PurchaseDate, PurchaseLoc)
```

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 (7 x 3 = 21 points).

a) E1: $\Pi_{\text{FirstName,LastName,BirthDate}}(\sigma_{\text{Item.Brand='Apple'} \wedge \text{Item.Category = 'Phone'} \wedge \text{Customer.CustomerID = Purchase.CustomerID} \wedge \text{Purchase.ItemID = Item.ItemID} \wedge \text{Purchase.PurchaseLoc = 'USA'}}(\text{Customer X Purchase X Item}))$

E2: $\Pi_{\text{FirstName,LastName,BirthDate}}(\sigma(\text{Customer}) \bowtie \sigma_{\text{PurchaseLoc = 'USA}}(\text{Purchase}) \bowtie \sigma_{\text{Category = 'Phone'}}(\text{Item}))$

b) E1: $\Pi_{\text{Location}}(\sigma_{\text{Purchase.PurchaseDate < '2021-07-01'} \wedge \text{Purchase.PurchaseDate > '2021-06-01'}}(\text{Customer} \bowtie \text{Purchase})) \cap \Pi_{\text{Location}}(\sigma_{\text{Item.Category = 'Frozen'} \wedge \text{Item.ExpirationDate < '2021-07-20'}}(\text{Customer} \bowtie \text{Item}))$

E2: $\Pi_{\text{Location}}(\sigma_{\text{Purchase.PurchaseDate < '2021-07-01'} \wedge \text{Purchase.PurchaseDate > '2021-06-01'}}(\text{Customer} \bowtie \text{Item} \bowtie \text{Purchase}) \cap \sigma_{\text{Item.Category = 'Frozen'} \wedge \text{Item.ExpirationDate < '2021-07-20'}}(\text{Customer} \bowtie \text{Item} \bowtie \text{Purchase}))$

c) E1: $\Pi_{\text{FirstName,LastName}}(\Pi_{\text{CustomerID}}(\sigma_{\text{PurchaseLoc='Chicago'}} \text{Purchase} \bowtie \sigma_{\text{Brand='Nestle'}}(\text{Item})) \bowtie \text{Customer})$

E2: $\Pi_{\text{FirstName,LastName}}(\sigma_{\text{PurchaseLoc='Chicago'} \vee \text{Brand='Nestle'}}(\text{Purchase} \bowtie \text{Item} \bowtie \text{Customer}))$

Section 2. Query Processing

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 pts + 10 EC = 20 pts)

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

The **union** 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 UNION 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 UNION table2`.

Output Format:

1. If the union cannot be carried out under the constraints outlined in `config.txt`, please write "INVALID MEMORY" to the first cell in the output file.
2. Please include the column names and the data of the resulting table. The smaller table should be loaded first.

Submission:

Please submit your code as a submission for "Homework 4 Q3 Option 1" 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 pts)

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

[73, 63, 25] [70, 68, 82] [94, 97, 34] [45, 13, 84] [45, 37, 22] [58, 63, 58]
[16, 10, 77] [52, 42, 61] [86, 61, 28] [51, 34, 97] [90, 68, 31] [52, 47, 61]

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 merging sorted runs
- For sorted runs, write as: [1,3][5,4][7,9]=>[1,3][4,5][7,9]
- For merging sorted runs, write as: [1,3][2,4]=>[1,2]=>out
- If the output block cannot be outputted and needs memory reload, write as: [1,3][2,4]=>[3,_]
- If two slots have the same number, choose the leftmost first by default. For example: [***2***,3][**2**,4]=>[***2***,**2**]=>out

Section 3. Query Optimization

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

The following 3 tables from a Company database are stored on the disk:

```
Students(StudentID , FirstName, LastName, PhoneNumber)
Transcripts(TranscriptID , StudentID, UniversityID, GPA,Year)
Universities(UniversityID, UniversityName, StateName)
```

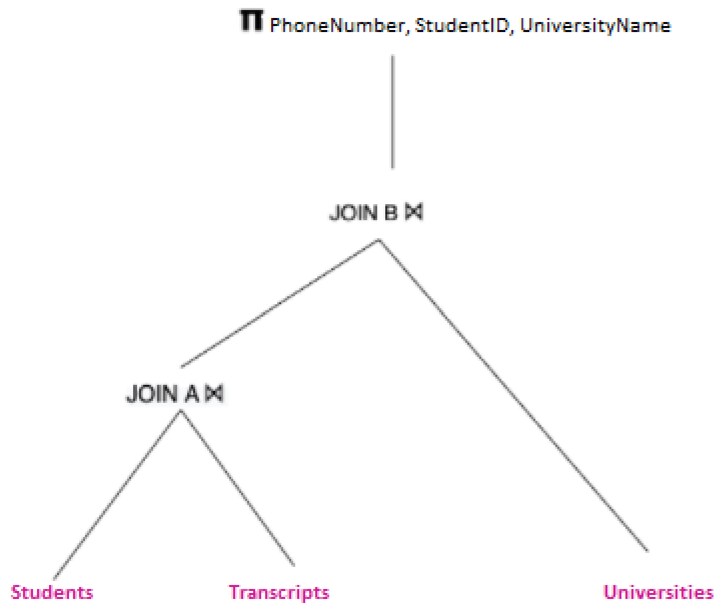
The statistics of these tables are shown below:

Table	Total Tuples	Total Occupied Blocks
Students	3000	50
Transcripts	2000	70
Universities	500	30

Consider the following query:

```
SELECT PhoneNumber, StudentID, UniversityName
FROM Students, Transcripts, Universities
WHERE Students.StudentID=Transcripts.StudentID
   AND Transcripts.UniversityID=Universities.UniversityID
```

The Joining Strategy is as follows:



Assume the number of blocks in memory is **M = 30** . Answer the following questions:

(i) Calculate the cost to perform block-based nested loop join in "Join A" . You can choose either `Students` or `Transcripts` as an outer relation for the 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 10000 tuples. Each block can contain up to 50 tuples in the resulting joined relation . Suppose 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 pts)

Q5. Cost-based Optimization (20 points)

Note: On your exam, the auto-grader rounds (not ceiling/floor) only 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(w,x,y)	**B(y,z)**	**C(x,y)**	**D(w,y,z)**
T(A) = 2000	T(B) = 1500	T(C) = 2500	T(D) = 3000
V(A, w) = 70			V(D, w) = 100
V(A, x) = 50		V(C, x) = 70	
V(A, y) = 80	V(B, y) = 100	V(C, y) = 50	V(D, y) = 50
	V(B, z) = 40		V(D, z) = 80

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**