CS 411: Database Systems

Spring 2023

Homework 5 (Due by 23:59 CT on April 13th)

Logistics

1. This homework is due on April 13th 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 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, Name, Gender, Phone, City, dob)
Appointment (PatientID, DoctorID, Date)
Course (CourseID, Course_Name, DoctorID, DepartmentID)
Department (DepartmentID, Dept_Name, Building)
```

Please answer the following queries using **relational algebra expressions**: (do not use an 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:
UIUC_CS411_SP23_HW5 schema

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 IDs of patients who have never been treated by a doctor from Chicago or those who have been treated by doctor from Chicago who only works in the Cardiology department. (10pt)
2. Find the IDs of patients who have never been seen by a doctor before the age of 50 and have always been treated by doctors from the Orthopedic department. Assume that subtracting a date-type value will give you the number of years of difference. (5pt)
3. Find the ID of the doctors who have treated at least one patient with a name that matches the doctor's name or those who have never taken courses outside of their department. (5pt)

Q2. Relational Algebra Equivalence (15 points)

Consider the following medical college/hospital database schema, with primary key(s) underlined:

```
Doctor (DoctorID, Name, DepartmentID, Phone, City, salary)
Patient (PatientID, Name, Gender, Phone, City, dob)
Appointment (PatientID, DoctorID, Date)
Course (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:** $(\pi_{\text{DepartmentID}}(\pi_{\text{DoctorID, DepartmentID}}\text{Doctor} - \pi_{\text{DoctorID, DepartmentID}}(\text{Department} \bowtie \text{Doctor}))) \cup (\pi_{\text{DepartmentID}}((\pi_{\text{DepartmentID}}\text{Department} \cap \pi_{\text{DepartmentID}}\text{Doctor}) \bowtie \text{Doctor}))$
   E2: $\pi_{\text{DepartmentID}}(\text{Doctor} \bowtie \text{Department}) \cup (\pi_{\text{DepartmentID}}\text{Doctor} - \pi_{\text{DepartmentID}}(\text{Doctor} \bowtie \text{Department}))$
   Assume Doctor.DepartmentID is a non-null foreign key. (5pt)
2. **E1:** $(\pi_{\text{PatientID}}(\boldsymbol{\sigma}_{\text{Patient.City == 'Chicago'}}\text{Patient}) \bowtie \text{Appointment}) \cup (\text{Appointment} \bowtie \pi_{\text{DoctorID}}(\boldsymbol{\sigma}_{\text{Doctor.City = 'Nashville'}}\text{Doctor}))$
   E2: $\pi_{\text{Appointments.PatientID, Appointments.DoctorID,Appointments.Date}}((\boldsymbol{\sigma}_{\text{Patient.City == 'Chicago'}}\text{Patient} \bowtie \text{Appointment}) \bowtie_{\text{Appointment.DoctorID = Doctor.DoctorID}} \boldsymbol{\sigma}_{\text{Doctor.City == 'Nashville'}}\text{Doctor})$ (10pt)

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.

If you submit solutions for both options, we will grade only Option 1.

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

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 (the order of columns and names of columns must be the same) by finding common rows in both tables. Our code skeleton handles I/O operations, 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 set intersection cannot be carried out under the constraints outlined in **`config.txt`**, please write "INVALID MEMORY" in the output file.
2. Please include the column names and the data in 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 5 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):

[54,13,81], [64,82,78], [48,32,74], [66,12,24], [69,53,58], [18,39,71]

[95,47,10], [53,87,75], [74,43,8], [55,90,62], [88,88,43], [5,5,18]

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,_]
- If no more memory reloads are possible for a block, then discard that block and keep merging with the remaining blocks.

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

The following three tables from a museum system database are stored on the disk:

```
Paintings(PaintingID, PaintingName, ArtistName, FinishYear)
Collections (PaintingID, MuseumID, MuseumSection, FloorNum)
Museums(MuseumID, MuseumName, Address)
```

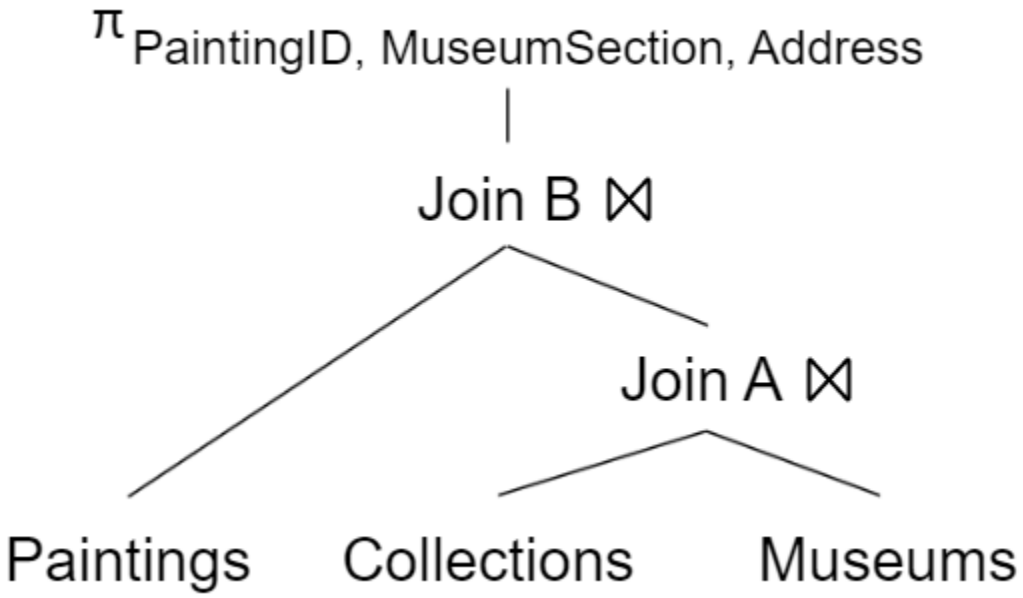
The statistics of these tables are shown below:

Table	Total Tuples	Total Occupied Blocks
Paintings	3600	36
Collections	9600	320
Museums	600	50

Consider the following query:

```
SELECT PaintingID, MuseumSection, Address
FROM Paintings, Collections, Museums
WHERE Collections.MuseumID=Museums.MuseumID
   AND Paintings.PaintingID=Collections.PaintingID
```

The Joining strategy is as follows:



Assume the number of blocks in memory is **M = 42**. Answer the following questions:

1. Calculate the **optimal** cost to perform the join in "Join A". Please answer the following questions: (10 points)
    a. Is one-pass join feasible for this question? Justify your answer.
    b. If YES, calculate the cost using one-pass join.
    If NO, calculate the cost using block-based nested loop join.

    In this question, when calculating the cost of block-based nested loop join:
- Please clearly state your choice of outer and inner relation, which should **minimize** the cost.
- **Ceil** the intermediate result of B(R)/(M-2), where B(R) is the outer relation.

2. Assume that after "Join A", the joining result would be stored back to the disk first and the resulting table has 7200 tuples. Each block can contain up to 80 tuples in the resulting joined relation. Assume that the tuples are densely stored (i.e. clustered file organization). Calculate the **optimal** cost to perform the join in "Join B". Please answer the following questions: (10 points)
    a. What is the total number of blocks for the result after "Join A"?
    b. Is one-pass join feasible for “Join B”? Justify your answer.
    c. If YES, calculate the cost using one-pass join.
    If NO, calculate the cost using block-based nested loop join.

    In this question, when calculating the cost of block-based nested loop join:
- Please clearly state your choice of outer and inner relation, which should **minimize** the cost.
- **Ceil** the intermediate result of B(R)/(M-2), where B(R) is the outer relation.

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(w,y,z)	**B(w,x)**	**C(w,x,z)**	**D(w,x,y,z)**
T(A) = 2000	T(B) = 8000	T(C) = 6000	T(D) = 5000
V(A, w) = 50	V(B, w) = 40	V(C, w) = 30	V(D, w) = 40
	V(B, x) = 25	V(C, x) = 50	V(D, x) = 20
V(A, y) = 40			V(D, y) = 50
V(A, z) = 40		V(C, z) = 10	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. Use the dynamic programming algorithm we learn in lecture to calculate as few rows as possible. Each step in the dynamic programming algorithm should be one row.

Subquery	Size	Cost	Plan

Extra credit: Physical Query Plan (15 EC points)

This question is optional and will be worth 15 extra credit points in total.

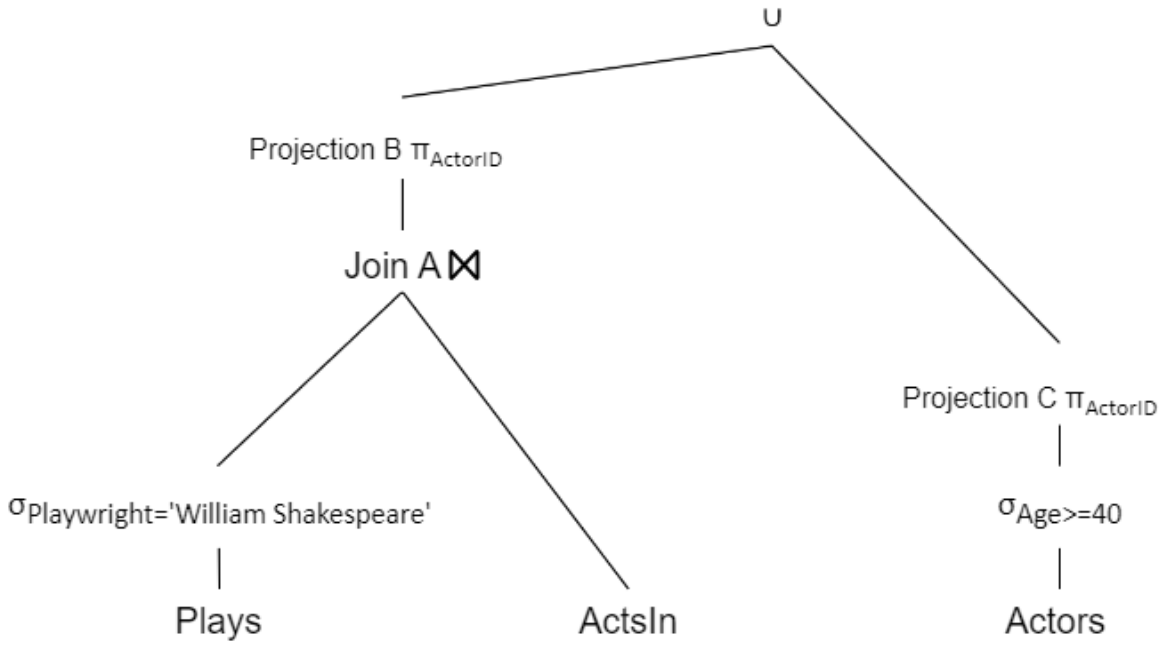
Consider the following schema for a Play library. Assume we have the following statistics for the tables and columns with the primary key(s) underlined:

```
Actors (ActorID, ActorName, Age)
ActsIn (ActorID, PlayID, PerformanceDate, Theatre)
Plays (PlayID, PlayName, Playwright)
```

Select the ActorIDs of Actors who are at least 40 years old or have acted in at least one play written by “William Shakespeare”.

Table	# tuples	# blocks
ActsIn	25000	1250
Plays	200	25
Actors	6000	100

Column	# distinct
Actor.Age	[18, 100)
Plays.Playwright	25
ActsIn.ActorID	6000
Acting.PlayID	200



Assume that the Plays, Actors, and ActsIn tables have a clustered index on their respective primary keys. Assume the distribution of Age and Playwright values are uniform, to compute estimated sizes after selection.

a. What is the cost and size of the selection Age>=40? Assume we have no index on Age. (2 points)

b. What is the cost and size of the selection Playwright= “William Shakespeare”? Assume we have an unclustered index on Playwright. (2 points)

c. What is the cost of executing Join A if we use an index-based nested loop join? Use the result of the Playwright selection as the inner relation and follow the assumptions from (b). Ceil the intermediate result. (3 points),

d. Suppose we plan to use a hash-based algorithm for JOIN A and the two-pass hash-based join. What is the cost of executing JOIN A? What is the **precise** range for the possible memory size M? **Ceil** the lower boundary. (3 points)

e. Assuming that
    - The number of tuples after Projection B is 4000
    - The block size for the result tables after projection B/C is 200 tuples
    - The memory capacity is 25 blocks

    can we perform a one-pass algorithm for the union operation? If yes, estimate the cost of the union. If not, explain why and estimate the total cost of the union with the appropriate algorithm. (Hint: First, estimate the number of blocks for the result tables after projections B and C). (5 points)