EECS 484 Homework #5

(76 points)

Due: Friday, Nov 15th, 2024 at 11:45 pm (ET)

Please read the following instructions before starting the homework:

This homework must be completed individually and can be submitted on Gradescope. Use entry code **XG8VVB** to self-enroll if you don't have access to the Gradescope course page.

No late days for homework! If you miss the due date, you get 0 points. If your PDF gets modified after the due date, you get 0 points. No exceptions on this.

Honor Code

By submitting this homework, you are agreeing to abide by the Honor Code:
I have neither given nor received unauthorized aid on this assignment, nor have I concealed any violations of the Honor Code.

Question 1: Query Plan Trees (10 points)

Consider the following schemas and SQL query:

Schemas:

Courses(course_id, course_name, pid, dept)
Professors(professor_id, name, date_join)

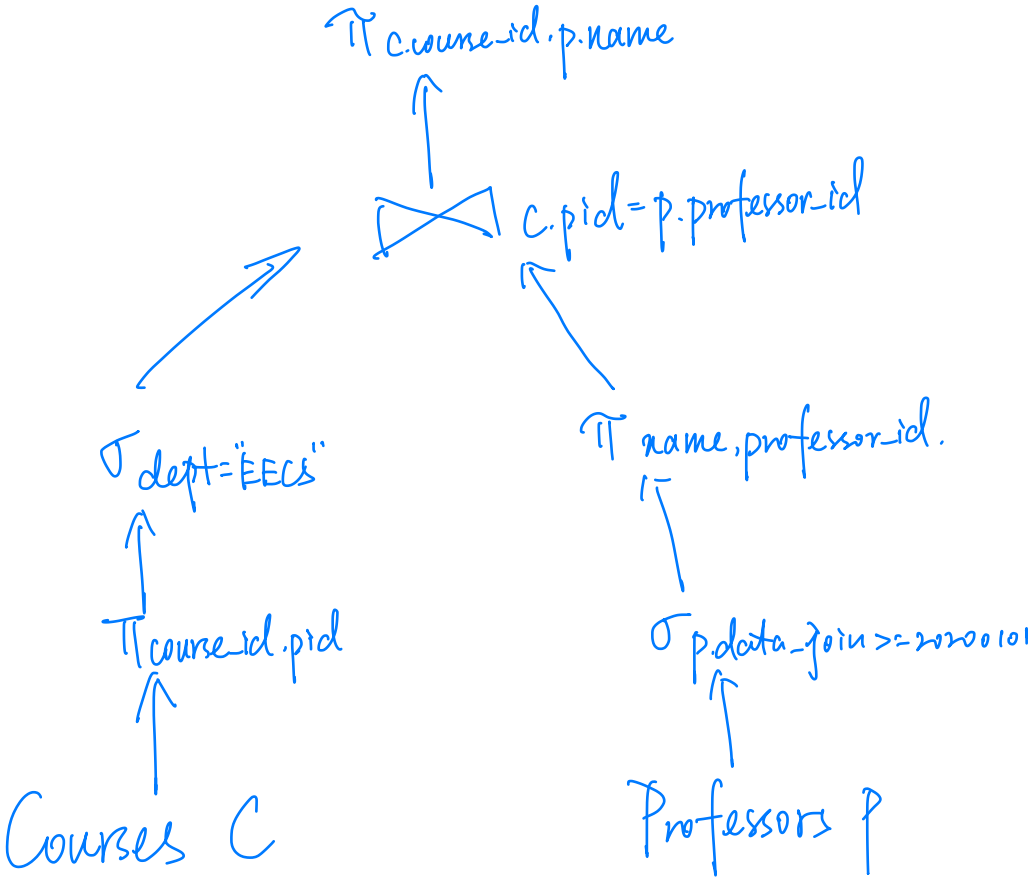
Query:

```
SELECT C.course_id, P.name
FROM Courses C, Professors P
WHERE C.pid = P.professor_id AND
    C.dept = “EECS” AND P.date_join >=20200101;
```

Using the query execution plan in the relation algebra notation discussed in class, create and optimize a query plan tree based on the following rules.

When possible, perform:

- Joins rather than cross-products
- Selections before joins
- Projections before selections



Question 2: Join Algorithms (16 points)

For this question, suppose that you are given two relations, **R** and **S**, and that you want to perform an inner equijoin between the two. Suppose that you have B = 650 memory pages available, and suppose the following is also true of the two relations:

- Number of records in R = 20,000
- Number of pages in R = 2,500
- Number of records in S = 16,531
- Number of pages in S = 3,600

Given the above information, answer the following questions. In all cases, please ignore the cost of writing the final result back to secondary storage. Please make sure to show your work for full credit.

2.1) (4 points) **Block Nested Loops Join**
Choose the outer relation such that the total cost is minimized. How many (a) total reads and (b) total writes are required?

Total read: $2500 + \lceil \frac{2500}{650-2} \rceil \times 3600 = 16900$

Total write: 0

2.2) (4 points) **Sort Merge Join**
How many (a) total reads and (b) total writes are required?

Total R read: $2500 \times (1 + \lceil \log_{649} \frac{2500}{650} \rceil) = 5000$

Total S read: $3600 \times (1 + \lceil \log_{649} \frac{3600}{650} \rceil) = 7200$

Merge: $2500 + 3600 = 6100$

Total read: $5000 + 7200 + 6100 = 18300$.

Total write: $2 \times 2500 + 2 \times 3600 = 12200$

2.3) (4 points) **Grace Hash Join**
How many (a) total reads and (b) total writes are required? Assume the relation is **evenly distributed** in the partitioning phase.

$\frac{|R|}{(B-1)P} \leq B-2 \Rightarrow P = 1$.

Total read: $2500 \times 2 + 3600 \times 2 = 12200$

Total write: $2500 + 3600 = 6100$

2.4) (4 points) Suppose we give you a new disk, and that reading a page from this disk will take 1 µs, and writing a page to disk will take X µs. Given your above analysis, for which values of X would you choose the block nested loops join? For which values of X would you choose the hash join? Round X to the nearest hundredth if necessary (e.g., if X = 7.346 use 7.35 as your final value for X).

$16900 \leq 12200 + 6100 * x$.

$\Rightarrow x \geq 0.77$

$x \geq 0.77$: BNJ

$x < 0.77$: GHJ

Question 3: Query Execution (14 points)

Consider the following schemas and SQL query:

Schemas:

Users(uid, uname, level)
Earned(uid, aid, timestamp, redeemed_status)
Achievement(aid, aname, category)

Query:

```
SELECT U.uid, A.aname
FROM Users U, Earned E, Achievement A
WHERE U.uid = E.uid AND A.aid = E.aid AND
    U.level > 80 AND A.category = 'Legendary'
```

Then consider a query plan that works as follows:

Join(Join(Selection(U), E), Selection(A)) with projections and selections cascaded and pushed in as far as possible.

Suppose we use Grace Hash Join for both joins, an unclustered hash index with a search key of **category** to access A, a clustered B+ tree index with a search key of **level** to access U, and a file scan to access E.

Assume the following:

- The B+ tree index on level is of height 2 with 64 leaf nodes. Each node occupies one page, and only the leaf nodes are on disk (i.e., all of the internal nodes are stored in memory).
- In the B+ tree, each leaf node may not be completely full (in order to improve the average case efficiency of insertions). Assume that each leaf node in the B+ tree is 62.5% full (this is also known as the **fill factor**). The space used in each node in the B+ tree should not exceed 62.5% of the total size of a page but should also fit in as many entries as possible.
- There are 100 **distinct values** for the level attribute, with the lowest level attribute being 1 and the highest being 100. Assume that level is evenly distributed (i.e., all values are equally likely: 1… 100).
- Each attribute or search key has a size of 16 bytes.
- Each record ID (pointer) has a size of 16 bytes.
- Our system has a page size of 4096 bytes and a memory **buffer size of 8 pages**. The I/O block size is a single page.
- There are **100K** (i.e., 100 * 1024) rows in the Earned relation.
- All indexes use Alternative 1, as discussed in class.
- Ignore the size of pointers between leaf nodes.
- We fill in as many records/tuples as possible in all record pages.

For each join, you may choose either operand as the inner one (whichever results in a lower cost!)

Given the above, answer the following questions. Please make sure to show your work for full credit.

3.1) (2 points) How many records are in U?

Hint: Each leaf node entry consists of both a search key and an RID (pointer to record).

Leaf nodes * fillfactor * 4096 = 64 * 62.5% * 4096 bytes.
Search key + RID = 16 + 16 = 32 bytes.
Total records: 64 * 62.5% * 4096 ÷ 32 = 5120

3.2) (2 points) How many record pages does U have?

Hint: The size of each record relates to the number of attributes for that record and the size of each of those attributes.

Users (uid, name, level) => 3 * 16 = 48 bytes.
Record Page = 5120 * 48 ÷ 4096 = 60.

3.3) (2 points) How many record pages does E have?

Hint: The size of each record relates to the number of attributes for that record and the size of each of those attributes.

E (uid, aid, timestamp, redeemed_status) : 4 * 16 = 64 bytes.
Record page = 100 * 1024 * 64 ÷ 4096 = 1600

3.4) (4 points) How many read I/Os are required to read the portion of relation U required for the above query? How many write I/Os are required to materialize only the SELECTED rows in the results? Please clearly show what each cost is (e.g., # of reads = ##, # of writes = ##) along with your work.

Hint: When materializing the relation U we only store the rows that satisfy U.level > 80 and each row only contains the attributes that will be used later for the join and projection operators.

$P(U.level > 80) = \frac{1}{5}$

leaves $= \lceil \frac{1}{5} \times 64 \rceil = 13$

pages $= \frac{1}{5} \times 60 = 12$

Total reads $= 12 + 13 = 25$

Total write $= \frac{1}{5} \times 5120 \times 16 \div 4096 = 4$

3.5) (4 points) What would be the number of I/Os of a Grace Hash Join in (a) the partitioning phase and (b) the probing phase for the join(selection(U), E) part of the query plan for the above query? Assume selection(U) has already been materialized (from previous subproblem). Also assume you do not do any projection or selection on E. Please clearly show both costs (e.g., partitioning phase = ##, probing phase = ##) along with your work.

U: $N = 5120 \times \frac{1}{5} \times 16 \div 4096 = 4$

$n = 5120 \times \frac{1}{5} = 1024$.

E: $M = 1600$

$m = 100 \times 1024 = 102400$.

partition: $2b(M+N) = 3 \times 10 \times (4 + 1024) = 3 \log$

probing: $M + N = 1604$.

$\frac{|N|}{(B-1)p} \leq B - 2 \Rightarrow p = 1$.

Question 4: Left Deep Plans (36 points)

Consider the following schemas:

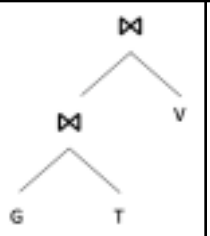
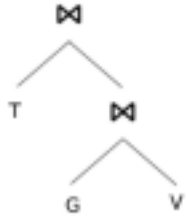
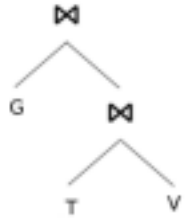
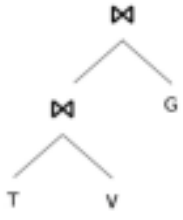
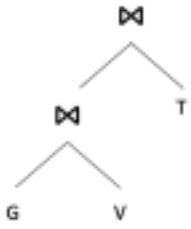
```
Team(tid, tname, sport, ranking)
Game(tid, vid, date, result)
Venue(vid, vname, city, capacity)
```

Consider the following query for question 1:

```
SELECT T.tname
FROM Team T, Game G, Venue V
WHERE T.tid = G.tid AND
      G.vid = V.vid AND
      V.city = 'Los Angeles' AND
      T.sport = 'basketball';
```

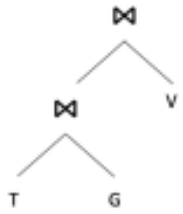
AE

4.1) (3 points) Given the above schema and no information on data sizes, select the plans that are valid left-deep plans for the query (avoid plans that cause a join to degenerate into a cross-product) from the plans given below.

A	B ✓	C ✗	D ✗	E
				

4.2 (33 points) Consider the following plan for the questions that follow later in this problem. This plan only shows the join operators. Assume that selections and projections are applied on the edges as appropriate so that they are moved inside to reduce table sizes as much as possible prior to any joins.

For example, between relation T and the join operator, you would want to do a selection on T.sport = basketball and also a projection to remove columns that are not going to be used. You can assume that projections and selections can be done together as tuples of T are being read into the buffer. Similarly, you will want to insert selections and projection operators along other edges as needed.



Consider the following information.

Team Table: 400 data pages

- There are 8 types of sports, one of which is basketball. All teams play a sport and each sport has the same number of teams that play it.
- Index on Team Table: Clustered B+ tree index on attribute sport. The entire tree, including the leaves, lives in memory.

Venue Table: 200 data pages

- There are 10 unique cities in which venues can be, each of which have the same number of records. (Note that each city can have multiple vid's)
- No indexes are available on Venue.

Game Table: 2000 data pages

- Every team has the same number of venues where it participated in a game as any other team.
- Every venue has the same number of teams that played in it as any other venue.
- No indexes are available on Game.

System Info:

- Page size = 512 bytes
- Each attribute is 64 bytes

Additional assumptions:

1. If there are selections and/or projections after a join operation, they will be pipelined (done in memory as the final result is being produced). However, you should assume that results from a previous join (with the pipelined selection and projection having been applied) are materialized to disk for any subsequent join operation.
2. Similarly, if a selection and/or projection is done on input tables (e.g., T) prior to the first join with that table, assume that you would materialize the result after the selection and projection.
3. Use an index for selection whenever there is an index match on selection predicates (not join predicates).
4. There is sufficient memory so that the read cost to join any two relations A and B can be assumed to be |A| + |B| (Think about why this is a reasonable estimate).
5. Ignore the cost of writing the final result (but not intermediate results).

Answer the following questions regarding the query plan:

1. (3 points) After applying a selection predicate on the edge from Team to the Join, what is the number of qualified tuples? If no selection predicate is needed, assume that the selection predicate is true.

   team: 4 attributes.

   # entry: $400 \times 512 \div 4 \div 64 = 800$.

   qualified tuples: $800 \times \frac{1}{8} = 100$

2. (3 points) After applying a projection operation (if any) on the edge from the Team to the Join, what is the size of each tuple in the resulting intermediate relation? If no projection is needed, then give the size of the entire tuple as your answer.

   T.tid & T.tname => 2 attributes.

   $2 \times 64 = 128$ bytes.

3. (6 points) What is the total cost (read and write) of applying both selection and projection to the Team relation before the first join that involves it? Note that the intermediate table after selection and projection is required to be materialized prior to the join (see additional assumption #2), so consider the cost of materialization in your answer.

   (Hint: See additional assumption #3)

   read: $400 \times \frac{1}{8} = 50$

   write: $50 \times \frac{64 \times 1}{64 \times 4} = 25$

   Total IO: $50 + 25 = 75$

4. (6 points) What is the total cost (read and write) of applying selection and projection (if any) to the Venue relation before any join, including the cost of materializing the

result if any operation was applied?

Selection: 2000

projection: $2000 \times \frac{1}{10} \times \frac{64 \times 1}{64 \times 4} = 5$.

Total: $2000 + 5 = 2005$

5. (3 points) If we join Team and Game together based on tid after selections and projections on both tables (if any), how many estimated matching tuples are there in the join result?

Reading G: 2000.

game tuples: $2000 \times 512 / 4 \times 64 = 4000$.

matching tuples: $4000 \times \frac{1}{8} = 500$

6. (6 points) What is the additional cost (read and write) of joining Team and Game as the first step of the plan, assuming that all the required intermediate inputs to the join are already on disk? Include the cost of materializing the intermediate result, which will be used in the second join.

Team: $N = 25$

Game: $M = 2000 \times \frac{64 \times 2}{64 \times 4} = 1000$.

Join Team and Game: Read: $M + N = 1025$.
write: $500 \times 64 \times 2 \div 512 = 125$

Total IO: $1025 + 125 = 1150$.

7. (6 points) What is the total estimated cost (without the final write) of the plan ((Team ⋈ Game) ⋈ Venue)?

selection and projection:
Team IO: 75.
Game IO: $2000 + 1000 = 3000$.
Venue IO: 205.

Join 1: 1150
Join 2: $125 + 5 = 130$

Total: $75 + 3000 + 205 + 1150 + 130 = 4560$.