Assignment (V)

Due: May 28, 2023

Problem 1 (20 points). We have a database file with N = 1000000 pages and we want to sort it using external merge sort.

1. Assume that the DBMS has 200 buffer pages. How many sorted runs are generated? Note that the final sorted file does not count towards the sorted run count.

   (a) 5024.

   (b) 5025.

   (c) 5026.

   (d) 5027.

2. Again, assuming that the DBMS has 200 buffer pages. How many passes does the DBMS need to perform in order to sort the file?

   (a) 1.

   (b) 2.

   (c) 3.

   (d) 4.

3. Again, assuming that the DBMS has 200 buffer pages. How many pages does each sorted run have after the second pass (i.e. Note: this is Pass 1 if you start counting from Pass 0)?

   (a) 199.

   (b) 200.

   (c) 39800.

   (d) 40000.

4. Again, assuming that the DBMS has 200 buffer pages. What is the total I/O cost to sort the file? (NOTE: the output cost of the final pass is included).

(a) 3000000.

(b) 5000000.

(c) 6000000.

(d) 7000000.

Problem 2 (30 points). Consider relations $R(a, b)$ and $S(a, c, d)$ to be joined on the common attribute a. Assume that there are no indexes available on the tables to speed up the join algorithms.

- There are $B = 36$ pages in the buffer.
- Table R spans $M = 1800$ pages with 100 tuples per page.
- Table S spans $N = 600$ pages with 60 tuples per page.

Answer the following questions on computing the I/O costs for the joins. You can assume the simplest cost model where pages are read and written one at a time. You can also assume that you will need one buffer block to hold the evolving output block and one input block to hold the current input block of the inner relation. You may ignore the cost of the writing of the final results.

1. Hash join with S as the outer relation and R as the inner relation. You may ignore recursive partitioning and partially filled blocks.

   1) What is the cost of the partition phase?

      (a) 1800.

      (b) 2400.

      (c) 3600.

      (d) 4800.

   2) What is the cost of the probe phase?

      (a) 1800.

      (b) 2400.

      (c) 3600.

      (d) 4800.

2. Block nested loop join with R as the outer relation and S as the inner relation:

   (a) 31800.

   (b) 32400.

   (c) 33000.

   (d) 33600.

3. Block nested loop join with S as the outer relation and R as the inner relation:

   (a) 31800.

   (b) 32400.

   (c) 33000.

   (d) 33600.

4. Merge join with S as the outer relation and R as the inner relation. Assume R and S have been sorted on attribute a before joined.

   1) What is the cost of the merge phase assuming there are no duplicates in the join attribute?

      (a) 1200.

      (b) 1800.

      (c) 2400.

      (d) 3600.

   2) What is the cost of the merge phase in the worst case scenario?

      (a) 2400.

      (b) 4800.

      (c) 1080000.

      (d) 1200000.

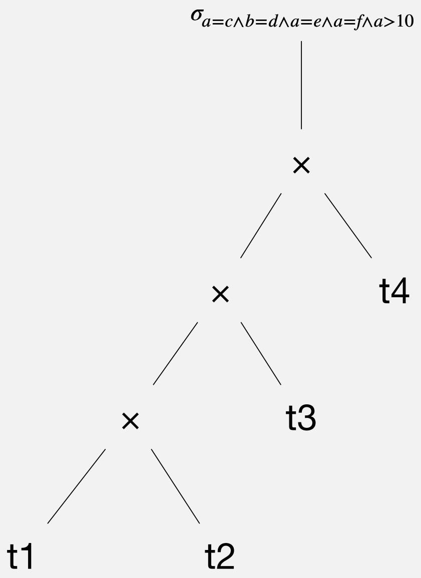
Problem 3 (20 points). Suppose we have four relations created by the following DDLs:

```
create table t1 (a int, b int);
create table t2 (c int, d int);
create table t3 (e int);
create table t4 (f int);
```

Consider the following SQL query on the above relations:

```
select
    *
from
    t1, t2, t3, t4
where
    a = c and
    b = d and
    a = e and
    a = f and
    a > 10;
```

In the DBMS, the query is transformed into the following logical plan:



Suppose we have the following groups of heuristics (i.e. rules):

- **Split Conjunctive Predicates**:
  $\sigma_{\theta_1 \wedge \theta_2}(r) \equiv \sigma_{\theta_1}(\sigma_{\theta_2}(r))$.
- **Replace Cartesian Products with Joins**:
  $\sigma_{\theta}(r \times s) \equiv r \bowtie_{\theta} s$.
- **Predicate Pushdown**:
  1. $\sigma_{\theta_1}(\sigma_{\theta_2}(r)) \equiv \sigma_{\theta_2}(\sigma_{\theta_1}(r))$.
  2. $\sigma_{\theta_1}(r \bowtie_{\theta_2} s) \equiv r \bowtie_{\theta_1 \wedge \theta_2} s$.
  3. $r \bowtie_{\theta_1 \wedge \theta_2} s \equiv (\sigma_{\theta_1}(r)) \bowtie_{\theta_2} s$ where θ_1 involves only the attributes of r.
  4. $r \bowtie_{\theta_1 \wedge \theta_2} s \equiv r \bowtie_{\theta_2} (\sigma_{\theta_1}(s))$ where θ_1 involves only the attributes of s.

Rewrite the logical plan with the above heuristics. You can apply the heuristics in whatever order you want as long as the resulting plan satisfies the following requirements:

- There is no cartesian product ($\times$) in the plan.
- All the predicates in selections (σ) and joins ($\bowtie$) are moved to the lowest applicable point in the plan.

Problem 4 (30 points). Besides inner join, left/right outer join and full outer join, there are left/right **semijoin** ($\ltimes_{\theta}/\rtimes_{\theta}$) and **antijoin** ($\overline{\ltimes}_{\theta}/\overline{\rtimes}_{\theta}$) in relational algebra. The definitions of left semijoin and left antijoin are as follows:

$$r \ltimes_{\theta} s = \{t_1 \in r \mid \exists t_2 \in s, \text{ predicate } \theta \text{ is satisfied on } t_1 \text{ and } t_2.\}$$
$$r \overline{\ltimes}_{\theta} s = \{t_1 \in r \mid \forall t_2 \in s, \text{ predicate } \theta \text{ is not satisfied on } t_1 \text{ and } t_2.\}$$

1. Transform the following SQL query into a logical plan with left semijoin:

   ```
   select course_id
   from section as S
   where semester = 'Fall' and year = 2017 and
       exists (select *
               from section as T
               where semester = 'Spring' and year = 2018 and
                   S.course_id = T.course_id);
   ```

2. Write a SQL query that can be expressed with a left antijoin.

3. Write an algorithm that computes left **antijoin** based on the block nested loop join.

Problem 5 (10 points). How long does it take you to finish the assignment? Give a score (1,2,3,4,5) to the difficulty of each problem. List all your collaborators if you have any.