**Written Assignment 3: Query Plan, and Query Optimization**

1. Join Algorithm (15 points)

Consider the join operation between relation r and s ( r θ s ), θ is r.A = s.B with the following information:

* Relation r contains 10,000 tuples and has 10 tuples per block.
* Relation s contains 2,000 tuples and has 5 tuples per block.
* There are 17 buffer blocks available in Memory.
* No sorted data in relation r and s.

Find Total cost (block transfers)

* 1. Using Block Nested Loop Join (3 points)
  2. Using Merge Join (2 points)
  3. Using Hash Join (Recursive partition) (2 points)
  4. Using Hash Join (No recursive partition) (3 points)
  5. If there are the infinity of memory, which join algorithm that you prefer? And why? (5 points)

1. Equivalent Expression (5 points)

There are four relations as following:

instructor (id, name, dept\_name, salary),

department (dept\_name, building, budget),

teaches (id, course\_id, sec\_id, semester, year),

course(course\_id, title, credits).

Query:

Find the instructor’s name who teaches ‘Advanced Database’ in Fall semester, 2021.

SQL command:

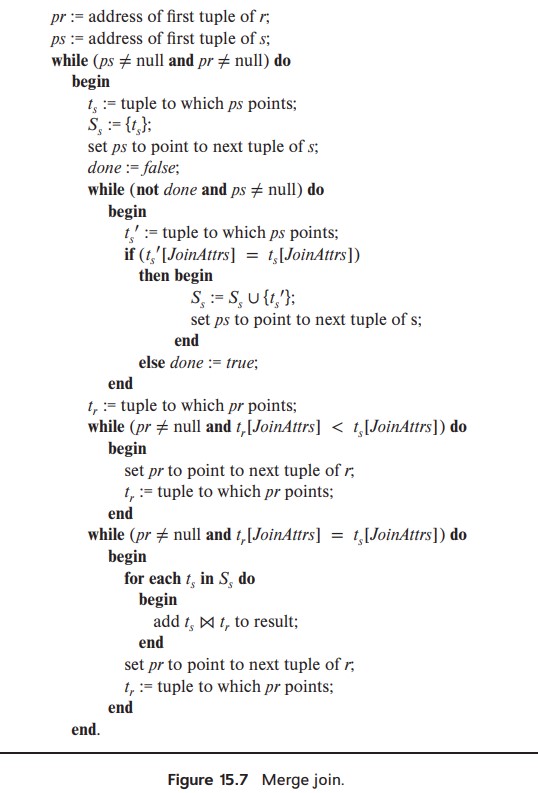
SELECT name FROM instructor, teaches, course WHERE

teaches.course\_id = course.course\_id AND instructor.id = teaches.id

AND title = ‘Advanced Database’ AND semester = ‘Fall’ AND year = 2021;

* 1. Find the relational algebra expression of this SQL command. (2 points)
  2. According to 2.1, find the equivalent expression? And show how this equivalent expression is better than the expression on 2.1 (3 points)

1. Merge Join Algorithm (10 points)



Using Figure 15.7 Merge join with the following samples relation R and S

**R S**

|  |  |  |
| --- | --- | --- |
| A1 | A2 | A3 |
| 11 | A | C |
| 12 | F | A |
| 12 | L | K |
| 14 | T | P |
| 15 | I | O |
| 16 | P | L |
| 17 | K | C |

|  |  |
| --- | --- |
| A1 | A4 |
| 10 | 30 |
| 11 | 30 |
| 12 | 20 |
| 14 | 40 |
| 14 | 10 |
| 17 | 50 |

How many rounds for the outer while loop? = \_\_\_\_\_\_\_\_\_ Rounds

Fill in the tuple on R that tR points to, tuple on S that tS points to and set SS after the end of each round.

|  |  |  |  |
| --- | --- | --- | --- |
| Round# | tR points to | tS points to | Ss |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Block Nested-Loop Join (r Θ s)**

The cost = block transfers (disk accesses)

**Merge Join (r Θ s)**

=

The cost = block transfers (disk accesses)

**Hash Join (r Θ s): No recursive partition**

The cost = block transfers (disk accesses)

**Hash Join (r Θ s): Recursive partition**

The cost = block transfers (disk accesses); where