Written Assignment 3:

Query Plan, and Query Optimization

Name: Venkata Lakshmi Sasank Tipparaju

Student# 700738838

CS 5600 – 13892

1. Join Algorithm (15 points)

   Consider the join operation between relation r and s ($r \bowtie_\theta s$), 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 10 tuples per block.
   There are 33 buffer blocks available in Memory.
   No sorted data in relation r and s.
   Find Total cost (block transfers)
   a. Using Block Nested Loop Join (3 points)
   b. Using Merge Join (2 points)
   c. Using Hash Join (Recursive partition) (2 points)
   d. Using Hash Join (No recursive partition) (3 points)
   e. If there are the infinity of memory, which join algorithm that you prefer? And why? (5 points)

Written Assignment 3

Sasank

(1) Given Relation b/w r & s

$(r \bowtie_\theta s)$, θ is $r.A = s.B$

Relation 'r' contains 10,000 tuples & 10 tuples per block.

=> Total no. of blocks required for 'r' = $\frac{10,000}{10}$

= 1000 blocks

Relation 's' contains 2000 tuples & 10 tuples per block

=> Total no. of blocks required for 's' = 2000/10

= 200 blocks

No. of buffer blocks available in memory : 33 Blocks

∴ r needs 1000 blocks, s needs 200 blocks, M = 33 Blocks

(1.1) Block Nested Loop Join :-

$$Cost = \left(\left\lceil \frac{b_r}{M-1} \right\rceil * b_s\right) + b_r \text{ block transfers}$$

$b_r = 1000$, $b_s = 200$, $m = 33$

$$\Rightarrow \left(\left\lceil \frac{1000}{33-1} \right\rceil \times 200 \right) + 1000$$

$$\Rightarrow \left(\lceil 31.25 \rceil \times 200 \right) + 1000$$

$$\Rightarrow 32 * 200 + 1000$$

$$\Rightarrow 7400 \text{ Block transfer.}$$

1.2 Merge Join

$$B_s \text{ (Sorting cost)} = b_r * \left(2 \left\lceil \log_{m-1} \left(\frac{b_r}{m} \right) \right\rceil + 2 \right) + b_s * \left(2 \left\lceil \log_{m-1} \left(\frac{b_s}{m} \right) \right\rceil + 2 \right)$$

$b_r = 1000$, $b_s = 200$, $m = 33$

Since data is not sorted $\Rightarrow$ need to calculate B_s

$$B_s = 1000 * \left(2 \left\lceil \log_{33-1} \left(\frac{1000}{33} \right) \right\rceil + 2 \right) + 200 * \left(2 \left\lceil \log_{33-1} \left(\frac{200}{33} \right) \right\rceil + 2 \right)$$

$B_S = 1000 * \left(2\left\lceil \log_{32} 30.30 \right\rceil + 2\right) + 200 * \left(2\left\lceil \log_{32} \frac{200}{6} \cdot 06 \right\rceil + 2\right)$

$\div 1000 * (2(1)+2) + 200 * (2(1)+2)$

$\Rightarrow 4000 + 800$

$B_S \Rightarrow 4800$ sorting Cost Block transfer.

$\Rightarrow$ Total Cost $\Rightarrow B_S + b_r + b_s$

$\Rightarrow 4800 + 1000 + 200$

Cost $\Rightarrow$ 6000 Block transfer.

(1.3) · Hash Join (Recursive Partition)

Cost $\Rightarrow 2(b_r + b_s)\lceil \log_{M-1}(b_s) - 1 \rceil + b_r + b_s$ $\Rightarrow b_s \leq b_r$

Since $200 < 1000 \Rightarrow b_s \leq b_r$.

$\Rightarrow$ Cost $\Rightarrow 2(1000+200)\lceil \log_{33-1}(200) - 1 \rceil + 1000 + 200$

$\Rightarrow 2(1200)\left\lceil \log_{32}^{200} - 1 \right\rceil + 1200$

$\Rightarrow 2400\{1.5 - 1\} + 1200$

$\Rightarrow 2400(1) + 1200$

Cost $\Rightarrow$ 3600 Block transfer.

(1.4) Hash Join (No recursive partition)

Cost = 3(br+bs)
= 3(1000+200)
= 3*1200
Cost ⇒ 3600 block transfers.

(1.5) If there are infinity of memory

⇒ Block Nested Loop Join :- Cost = br+bs = 1000+200 = 1200
⟹ (M ≥ br+bs) ⇒ Seeks = 2 Seeks.

⇒ Hash Join ⇒ If entire table can be kept in main memory, no partitioning require
⇒ Cost ⇒ br+bs ⇒ 1000+200 ⇒ 1200 block transfers
∀ Hash Join ⇒ No recursive pattern is independent of M
⟹ Block NestedLoopJoin is preferred.

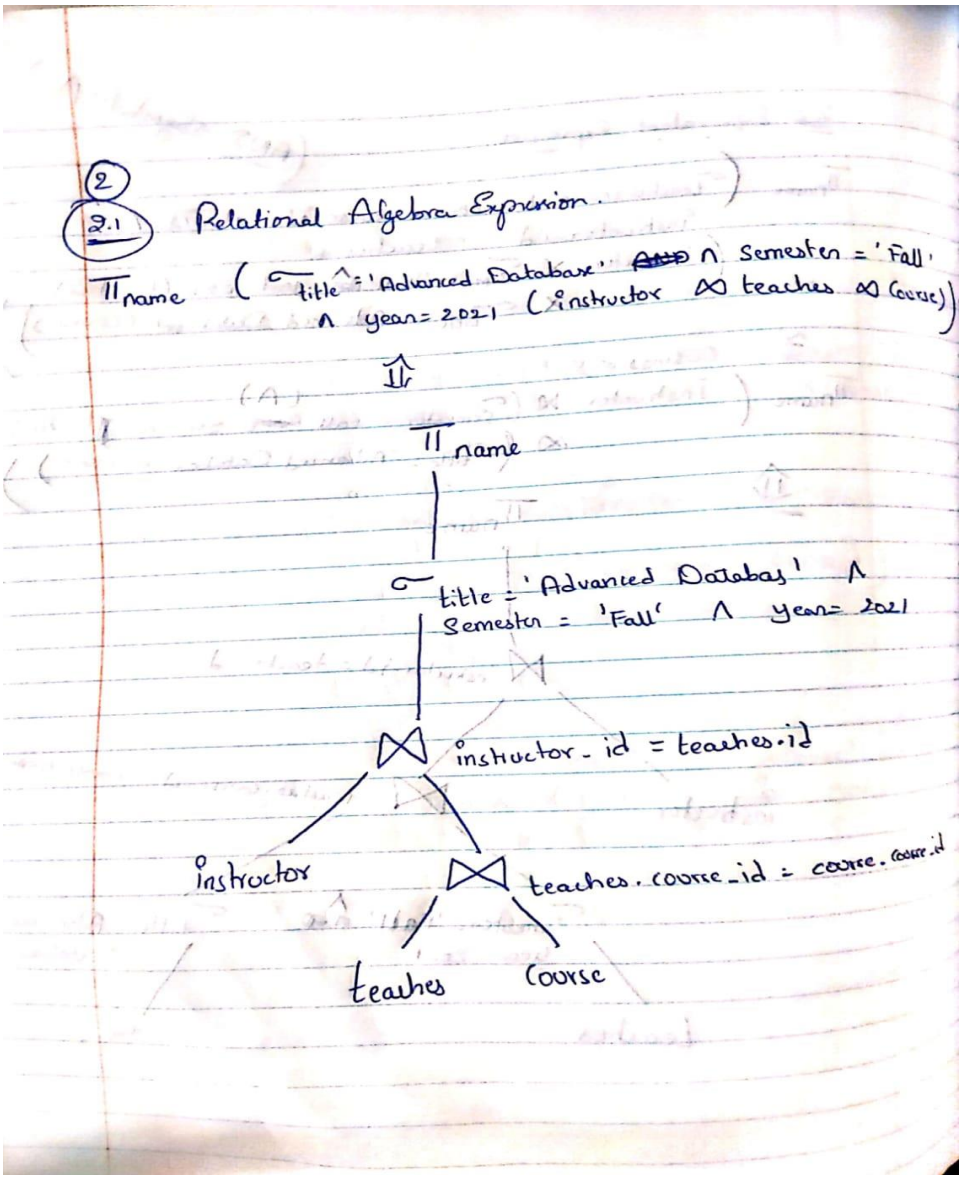
(*) Block Nested Loop Join is preferable, if relations fits entirely in memory, cost will be br+bs
i.e.. 1200 blocks transferable.
Because:- we can use longer relation as outer relation and read the block. The buffer will hold entire smaller relation one page for reading longer one for outer

2. 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).

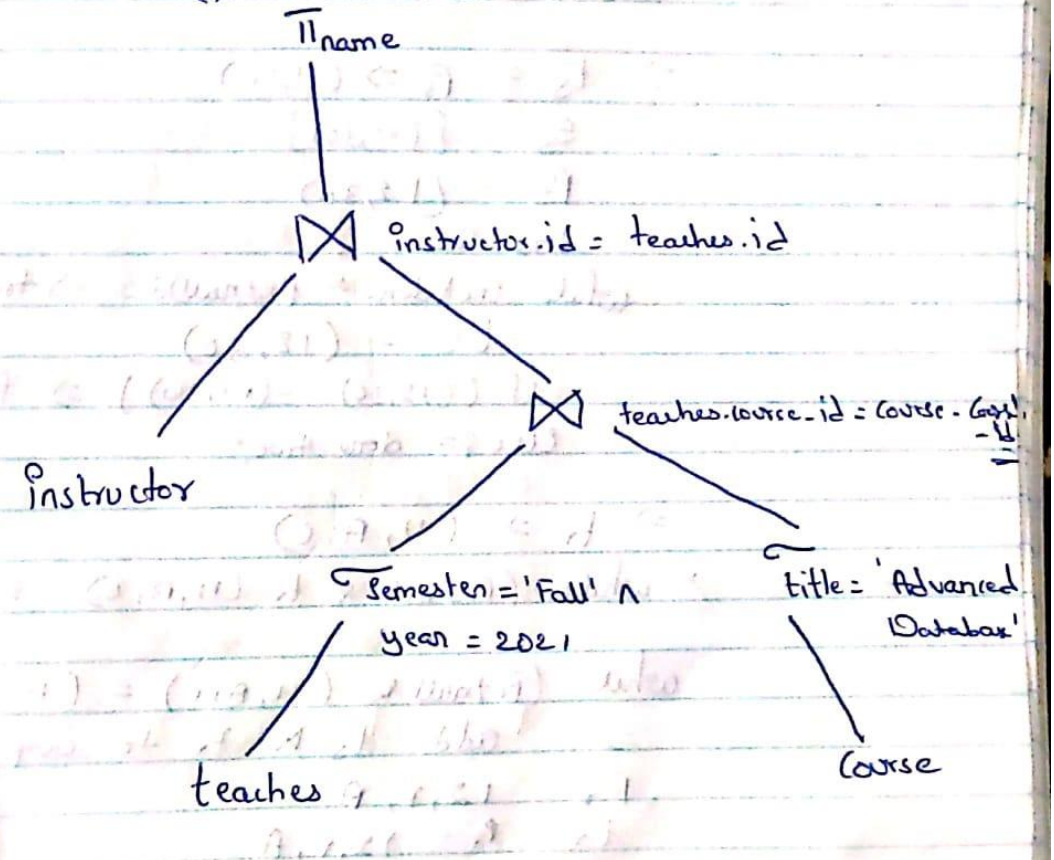
Query: Find the instructor's name who teaches 'Advanced Database' in Fall semester, 2021.

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



2.2 Equivalent Expression

$$\Pi_{name}\left(\text{instructor} \bowtie \left(\sigma_{Semester='Fall' \wedge year=2021}(teaches)\right) \bowtie \left(\sigma_{title='Advanced\ Database'}(course)\right)\right)$$



With Cartesian (X)

$$\Pi_{name}\left(\sigma_{teaches.course\text{-}id = course.course\text{-}id \wedge instructor.id = teach.id}\left(\text{instructor} \times \left(\sigma_{Semester='Fall' \wedge year=2021}(teaches)\right) \times \left(\sigma_{title='Advanced\ Database'}\right)\right)\right)$$

3. Merge Join Algorithm (10 points)

```
pr := address of first tuple of r;
ps := address of first tuple of s;
while (ps ≠ null and pr ≠ null) do
   begin
      t_s := tuple to which ps points;
      S_s := {t_s};
      set ps to point to next tuple of s;
      done := false;
      while (not done and ps ≠ null) do
         begin
            t_s' := tuple to which ps points;
            if (t_s'[JoinAttrs] = t_s[JoinAttrs])
               then begin
                        S_s := S_s ∪ {t_s'};
                        set ps to point to next tuple of s;
                    end
               else done := true;
         end
      t_r := tuple to which pr points;
      while (pr ≠ null and t_r[JoinAttrs] < t_s[JoinAttrs]) do
         begin
            set pr to point to next tuple of r;
            t_r := tuple to which pr points;
         end
      while (pr ≠ null and t_r[JoinAttrs] = t_s[JoinAttrs]) do
         begin
            for each t_s in S_s do
               begin
                  add t_s ⋈ t_r to result;
               end
            set pr to point to next tuple of r;
            t_r := tuple to which pr points;
         end
   end.
```

Figure 15.7 Merge join.

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

R

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

S

A1	A4
11	30
12	30
12	20
14	40
14	10
16	50

How many rounds for the outer while loop?

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#	tuple on R	tuple on S	Ss

3)

Round 1

P_r = (11, A, C)
P_s = (11, 30)
while ($P_s \neq$ null and $P_r \neq$ null) do ⇒ true

⇒ $t_s = P_s$ ⇒ (11,30)
S_s = {(11,30)}
P_s = (12,30)
done = false;
while (not done & $P_s \neq$ null) do ⇒ true.
$t's$ = (12,30)
if ((12,30) = (11,30)) ⇒ false
else ⇒ done = true;

⇒ t_r ⇒ (11, A, C)
✗ while ($P_s \neq$ null & t_r (11,A,C) < t_s (11,30)) ⇒ false

while ($P_r \neq$ null & (11,A,C) = (11,30)) ⇒ true
⇒ add t_s & t_r to output.
P_r = (12, F, A)
t_r = P_r = (12, F, A.
end of Round 1

Round 2 while ($P_s \neq$ null, $P_s \neq$ null) do ⇒ true
t_s = (12,30)
S_s = {(12,30)}
P_s = (12,20)
done = false;
while (not done & $P_s \neq$ null) do ⇒ true
$t's$ ⇒ (12,20)
if ((12,20) = (12,30)) ⇒ true
S_s = {(12,30)} ∪ {(12,20)}
S_s = {(12,30) & (12,20)}
P_s ⇒ (14,40)
t_r = (12, F, A)
✗ while ($P_r \neq$ null $t_r[1]$12 < $t_s[1]$·12) false
while ($P_r \neq$ null t(12,F,A)[3] = (12,30)[join)] true
break add t_s & t_r to result.

P_r = (12, L, K)
t_r = (12, L, K)
end of Round 2

Round 3 while ($P_s \neq$ null & $P_r \neq$ null) true
t_s = (14,40)
S_s = {(14,40)}
P_s = (14,10)
done = false
while (true) ⇒ ✓
$t's$ = (14,10)
if ((14,10)[1] = (14,40(1)) true ✓
S_s = S_s ∪ {$t's$}
= {(14,40), (14,10)}
P_s = (16, 50)
t_r = (12, L, K)
while ($P_r \neq$ null & 12 < 14) ⇒ True
P_r = (14, I, P)
t_r = (14, I, P)
while ⇒ false ✗
while ($P_r \neq$ null & 14 = 14) True
add t_s & t_r to result

P_r = (15, I, O) = t_r
end of Round 3

Round 4 t_s = (16,50)
S_s = {(16,50)}
P_s = Null.
done = false.
while ($P_s \neq$ null) ⇒ false.

t_r = (15, I, O)

while ($P_r \neq$ null & 15 < 16) ⇒ true
⇒ P_r = (16, P, L)
⇒ t_r = (16, P, L)
while (false)
while ($P_r \neq$ null 16 = 16) true ✓
for ⇒ add t_s & t_r to result
P_r = (17, K, L)
t_r = (17, K, L) ✓
End of Round 4

Round 5 P_s = null End
while (false)

Round #	tuple on R	tuple on S	S_s
1	(12, F, A)	(12, 30)	{(11, 30)}
2	(12, L, K)	(14, 40)	{(12, 30), (12, 20)}
3	(15, I, O)	(16, 50)	{(14, 40), (14, 10)}
4	(17, K, C)	NULL	{(16, 50)}
—	—	—	—

A total of 4 Rounds for outer while loop.

Output

A_1	A_2	A_3	A_4
11	A	C	30
12	F	A	30
12	F	A	20
14	T	P	40
14	T	P	10
16	P	L	50

Block Nested-Loop Join (r ⋈ Θ s)

The cost = $\left(\left\lceil \frac{b_r}{M-1} \right\rceil * b_s\right) + b_r$ block transfers (disk accesses)

Merge Join (r ⋈ Θ s)

$B_s\ (Sorting\ Cost) = b_r * \left(2\left\lceil log_{M-1}\left(\frac{b_r}{M}\right)\right\rceil + 2\right) + b_s * \left(2\left\lceil log_{M-1}\left(\frac{b_s}{M}\right)\right\rceil + 2\right)$

The cost = $B_s + b_r + b_s$ block transfers (disk accesses)

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

The cost = $3(b_r + b_s)$ block transfers (disk accesses)

Hash Join (r ⋈ Θ s): Recursive partition

The cost = $2(b_r + b_s)\lceil log_{M-1}(b_s) - 1\rceil + b_r + b_s$ block transfers (disk accesses); where $b_s \leq b_r$