S. Lakshmi Chandana
700757993

Written Assignment-3

1.

1.1) Using Block Nested Loop Join

Given no. of tuples in r = 100,000
no. of tuples per block = 10

no of blocks in relation(r)

$$b_r = \frac{100,000}{10}$$

$$= 10,000 (b_r)$$

No. of tuples in s = 20000
No. of tuples per block = 5
no. of blocks in relation(s) $b_s = \frac{20000}{5}$

$$= 4000 (b_s)$$

buffer
given no. of blocks in memory = 33(m)

total cost using block nested loop join =

$$= \left[\left(\frac{b_r}{m-1}\right) \times b_s\right] + b_r$$

$$= \left[\left(\frac{10000}{33-1}\right) \times 4000\right] + 10000$$

$$= \left[\left(\frac{10000}{32}\right) \times 4000\right] + 10000$$

$$= (313 \times 4000) + 10000$$

$$= 12,62,000.$$

1.2) Using merge Join

The total cost using merge join

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

$$= 10000 * \left(2\left\lceil \log_{33-1}\left(\frac{10000}{33}\right)\right\rceil + 2\right) + 4000\left(2\left\lceil \log_{33-1}\left(\frac{4000}{33}\right)\right\rceil + 2\right)$$

$$= 10000 * (2(2)+2) + 4000(2\lceil 2 \rceil + 2)$$

$$= 10000(6) + 4000(6)$$

$$= 84000$$

$$\text{Cost} = b_s + b_r + b_s$$

$$= 84000 + 10000 + 4000$$

$$= 98000$$

1.3) Using Hash Join (Recursive partition)

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

$$= 2(10000 + 4000)\left\lceil \log_{32} 40000 - 1\right\rceil + (10000 + 4000)$$

$$= 2(14,000)\lceil 2.3932 - 1\rceil + (14000)$$

$$= 2(14000)(1.3932) + 14000$$

$$= 53009. \text{ without totally ceiling.}$$

$$\text{Total cost} = 70,000 \text{ (By taking ceiling function)}$$

1.4 Using Hash Join (NO recursive partition)

Total cost using hash join = $3(b_r + b_s)$

$= 3(10000 + 4000)$

Total cost = 42,000.

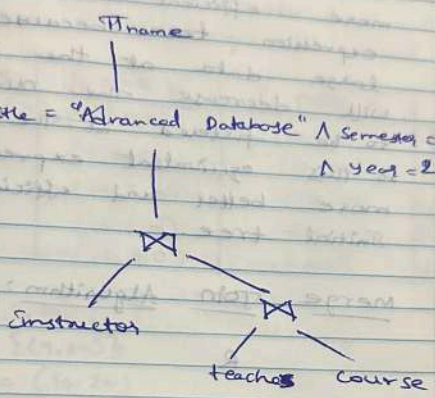
1.5 If there is infinite of memory we can use any of the join algorithm has the complexity is same.

But the efficient join algorithm is hashJoin with no recursive partition we prefer this because block nested loop join may show quadratic nature which takes more cost → when coming to merge join it is somehow expensive compare to hash join. So, we can prefer hash join which total cost will be $3(b_r + b_s)$

2.)

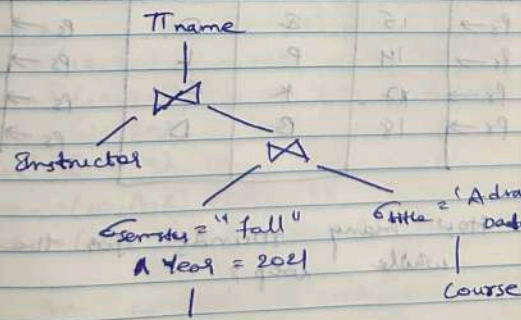
2.1) $\pi_{name}(\sigma_{title = 'Advanced\ Database' \wedge semester = 'fall' \wedge year = 2021}(Instructor \bowtie_{id} (teaches \bowtie_{course\_id} course)$

2.2) Initial expression Tree:



Equivalent expression:

$$\Pi_{name}\left(Instructor \bowtie \left(\sigma_{Semester = "fall" \wedge Year = 2021}(teaches)\right) \bowtie \left(\sigma_{title = "Advanced\ Data\ Base"}(course)\right)\right)$$



This equivalent expression tree is more efficient than the initial expression tree because of filtering large data at the initial stage will decrease the number of records to be filtered at the last stage. So this equivalent expression tree is more better and efficient than the Initial tree.

3.) Merge Join Algorithm:

R

	A1	A2	A3
$P_R \rightarrow$	10	A	C
$P_R \rightarrow$	11	F	A
$P_R \rightarrow$	12	L	K
$P_R \rightarrow$	12	T	P
$P_R \rightarrow$	15	S	O
$P_R \rightarrow$	14	P	L
$P_R \rightarrow$	17	K	C
$P_R \rightarrow$	18	B	D

S

	A1	A4
$P_S \rightarrow$	9	30
$P_S \rightarrow$	10	30
$P_S \rightarrow$	12	20
$P_S \rightarrow$	14	40
$P_S \rightarrow$	14	10
$P_S \rightarrow$	17	50

i) How many rounds for the outer while loop?

fill in the tuple on R that P_R points to tuple on S that P_S points to and set S_S after the end of each round.

Round #	P_R points to	P_S points to	S_S
1	(10, A, C)	(10, 30)	{(9, 30)}
2	(11, F, A)	(12, 20)	{(10, 30)}
3	(15, I, 0)	(14, 40)	{(12, 20)}
4	(16) I, 0)	(19, 50)	{(14, 40), (14, 10)}
5	(18, B, 0)	Null	{(17, 50)}

$t_S = (9, 30)$

$S_S = \{(9, 30)\}$

$P_S = (10, 20)$

$t_S' = (10, 30)$

$t_R = (10, A, C)$

Both the conditions in the while loops didn't satisfy the Condition so we can end round 1.

Round 2

$t_S = (10, 30)$

$S_S = \{(10, 30)\}$

$P_S = (12, 20)$

$t_S' = (12, 20)$

$t_R = (10, A, C) \Rightarrow t_R = t_S$

$P_R = (11, F, A)$; $t_R = (12, F, A)$

A_1	A_2	A_3	A_4
10	A	C	30

∴ The Condition is not true which $t_R \neq t_S$

So we exit & end round 2.

Round 3

$t_s = (12, 20)$

$S_s = \{\{(12, 20)\}\}$

$P_s = (14, 40)$

$t_s' = (14, 40)$

$t_r = (11, F, A)$ [11 < 12] as $[t_r < t_s]$

$P_r = (12, L, K)$

$t_r = (12, L, K)$ Here 12 = 12 (i.e $t_r = t_s$)

$P_r = (12, T, P)$

$t_r = (12, T, P)$ Here (12 = 12) $(t_r = t_s)$

$P_r = (15, I, O)$

$t_r = (15, I, O)$

A_1	A_2	A_3	A_4
10	A	C	30
12	L	K	20
12	T	P	20

Here the condition $t_r = t_s$ does not satisfied in while loop so we should exit and end the round 3.

Round 4:

$t_s = (14, 40)$

$S_s = \{(14, 40)\}$

$P_s = (14, 10)$

$t_s' = (14, 10)$ Here 14 = 14 (i.e $t_s = t_s$)

$S_s = \{(14, 40) \cup \{(14, 10)\}\}$

$S_s = \{(14, 40), (14, 10)\}$

$P_s = (17, 50)$

$t_s' = (17, 50)$

$t_r = (15, I, 0)$

∴ Here both the condition in while loop doesn't satisfy so we can end round 4.

Round 5:

$t_s = (17, 50)$

$S_S = \{(17, 50)\}$

P_S = null

$t_r = (15, I, 0)$ [15 < 17 (i.e $(t_r < t_s)$]

$P_r = (14, P, L)$

$t_r = (14, P, L)$ (14 < 17)

$P_r = (17, K, C)$

$t_r = (17, K, C)$ (17 = 17) (i.e $t_r = t_s$)

$P_r = (18, B, D)$

$t_r = (18, B, D)$

∴ $t_r = t_s$ conditions is not satisfied here so we can exit → end round 5.

∴ Here P_S is null, so the entry condition (or) main condition is not going to satisfy. so we can exit the join.

A_1	A_2	A_3	A_4
10	A	C	30
12	L	K	20
12	T	P	20
17	K	C	50