Solutions to Exercises on Final Review

Exercise 3: JOIN operation
Consider the company database discussed in the textbook and class with the following catalog information for the Employee and Department relations:

- $n_{\text{employee}} = 10{,}000$
- $b_{\text{employee}} = 2{,}000$
- $n_{\text{department}} = 125$
- $b_{\text{department}} = 13$
- Clustering index on the attribute salary in employee
- Secondary index on SSN (key attribute) in employee using B$^+$-tree with $HTi = 4$
- Secondary index on dno (non-key attribute) in employee using B$^+$-tree with $HTi = 2$, $LBi = 4$, and V(dno, employee) = 125
- Primary index on the attribute dnumber in department using B$^+$-tree with $HTi = 1$
- Secondary index on the attribute mgr_ssn in department using B$^+$-tree with $HTi = 2$

1. What join algorithm makes sense for joining Employee and Department on department number?

   Answer

   Calculate it using the different join options (nested block, index, sort, and hash-join)

   - Nested block with employee as outer loop:

   $$b_E b_D + b_E = 2000(13) + 2000 = 28,000$$

   - Nested block with department as outer loop:

   $$b_D b_E + b_D = 13(2000) + 13 = 26,013$$

   - Indexed nested block with employee as outer loop:

   $$b_E + n_E c = b_E + n_E(HTi + 1) = 2000 + 10000(2 + 1) = 32,000$$

   - Indexed nested block with department as outer loop:

   $$b_D + n_D c = b_D + n_D(HTi + \text{SC(dno, employee)}) = 13 + 125(2 + 80) = 10,263$$

   - Sort-merge join: need to sort employee table but not department

   $$\begin{aligned} 2b_E(\lceil \log_{M-1}(b_E/M) \rceil + 1) + b_E + b_D &= 2(2000)(\lceil \log_{M-1}(2000/M) \rceil + 1) + 2000 + 13 \\ &= 4000(\lceil \log_{M-1}(2000/M) \rceil + 1) + 2013 \end{aligned}$$

- Hash join with no recursive partitioning

$$3(b_E + b_D) = 3(13 + 2000) = 6,039$$

- Hash join with recursive partitioning

$$2(b_E + b_D)\lceil \log_{M-1}(b_D) - 1 \rceil + b_E + b_D = 2(13 + 2000)\lceil \log_{M-1}(13) - 1 \rceil + (13 + 2000)$$

If we have enough memory to fit all the partitions into available memory, we would use recursive partitioning. However, assuming that we don't have enough memory, then we would choose between hash join, sort-merge, and indexed nested block with department as outer loop depending on the value of M.

2. What join algorithm makes sense for joining Employee and Department on manager ssn?

Answer

Calculate it using the different join options (nested block, index, sort, and hash-join). Note that many of the calculations are repeated from above. Nested block (both employee and department as outer loop), indexed nested block with employee as outer loop, sort-merge join, hash join with no recursive partition, and hash join with recursive partitioning, so we only need to done new calculation.

- Indexed nested block with department as outer loop:

$$b_D + n_D c = b_D + n_D(HTi + 1) = 13 + 125(2 + 1) = 388$$

For this query, it should be obvious that we would use an indexed nested block with department as the outer loop.

Exercise 4: Serializability

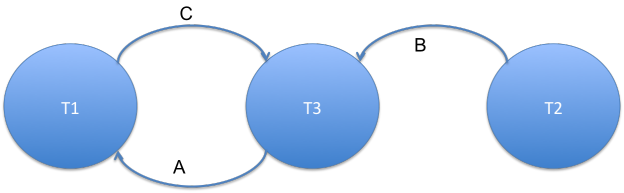
Consider the schedule given in the table below of three transactions T1, T2, and T3.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
T_1			R(A)		W(A)		R(C)		W(C)		
T_2				R(B)		W(B)					
T_3	R(A)	W(A)						R(B)	W(B)	R(C)	W(C)

1. Draw the precedence graph

Answer

The precedence graph contains nodes that denote each of the transactions and an edge if one of the operations in a transaction appears before a conflicting operation in another transaction. In this case, our graph will have 3 nodes, T1, T2, T3. Since T3 writes A before T1 reads and writes it, there will be an edge from T3 to T1. Since T2 writes B before T3 reads and writes B, there will be an edge from T2 to T3. Finally, since T1 writes C before T3 reads C, we have an edge from T1 to T3.



2. Is this schedule serializable?

Answer Since there is a cycle in the precedence graph, this is not a serializable schedule.