CS-354

End Sem Paper

Name : P.V. SRIRAM

Roll No. : 1801CS37

Q11) Propose a strategy that can be used to improve the concurrency degree of a tree-based protocol. Create your own example of a schedule to illustrate the proposed strategy.

Ans) In the existant tree based protocols, the following rules are followed,

(1) Partial order on database items determines a tree like structure

(2) Only exclusive locks are allowed.

(3) The first lock by T_i may be on any data item. Subsequently, a data Q can be locked by T_i only if the parent of Q is currently locked by T_i

(4) Data items can be unlocked at any time.

The purpose of an exclusive lock is to:

(a) Prohibit access of other transactions from its unlocked data items to the subtree where it will yet Lock additional data items.

(b) To prohibit access to an individual data item that it intends to update

(c) To prevent deadlock.

In general, the restrictions for these protocols state that an item cannot be locked unless the transaction is holding locks on a set of items (fathers of item), and that this set has some functional relational to other sets of the item in order to maintain serializability.

But the idea of using exclusive locks on data items to prohibit access the rest of the graph is ill-concieved. This can be overcome by using an Edge Lock Model

Edge Lock Model

Apart from other protocols, here, a transaction can issue locks on edges as well. There are two lock modes in this model, The first lock mode is denoted EX, and prevents transactions that come after T_i in the History to enter the subgraph that T_i is operating in. The second mode allows transactions which do not come after T_i in H to enter the subgraph and is denoted ES.

EX conflict with EX, ES while ES is conflicting with only EX.

(History H is chronological sequence of Transactions
i.e) $H = \{T_0, T_1, --- T_{N-1}\}$

If transaction T_i holds a lock on a data item or edge in mode M and T_j requests a lock in conflicting mode on the same entity then J_j must wait for T_i.

An Edge Lock protocol can be used to improve the concurrency of a tree based protocol.

Transformation from a Graph protocol to Edge protocol

We initially consider a tree T using exclusive locks and the following protocol.

T_i can request a lock on vertex A iff

(i) A is the first vertex locked by T_i or the father of A is locked in X mode by T_i

(ii) A has not yet been locked by T_i

(iii) A vertex can be unlocked at any time.

The corresponding edge lock protocol derived from the previous protocol is as follows

- T_i can lock edge <A, B> in EX mode iff:

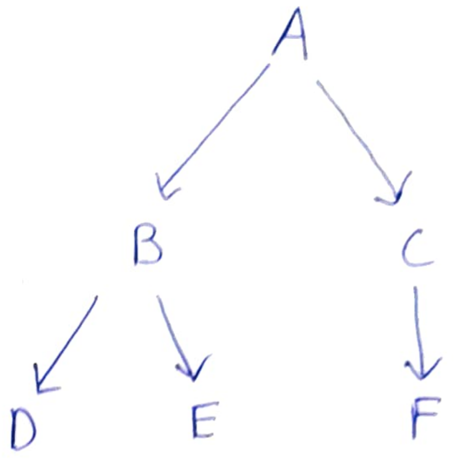
(i) <A, B> is the first edge to be locked by T_i or T_i has a lock on father of <A, B>

(ii) <A, B> has not been previously locked by T_i

T_i can lock an item A in X mode if it holds a lock on the incoming edge <Q, A> where Q is father of A.

Initial Tree protocol

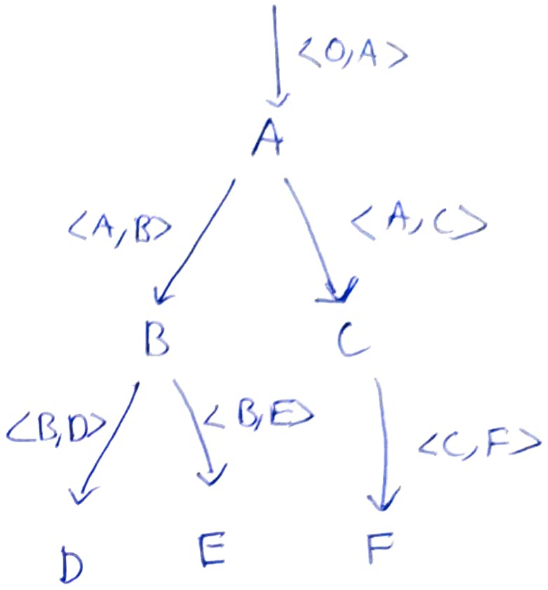
T_1	T_2	T_3
X(A)		
	X(B)	
	U(B)	
		X(C)
		U(C)
X(B)		
X(D)		
X(C)		
X(F)		
U(F)		
		X(F)
		U(F)
U(C)		
U(D)		
U(B)		



In this example, there are two redundancies (circled above). originally T_1 required only exclusive locking on D, F but due to the protocol, Even B,C are locked. But as T_2, T_3 have locked B,C respectively T_1 has to wait for both even through it never needed B,C. This anamoly has to be fixed.

Edge lock protocol

T_1	T_2	T_3
Ex(0,A)		
	Ex(A,B)	
X(A)		
	X(B)	
	UE(A,B)	
		Ex(A,C)
		X(C)
		UE(A,C)
EX(A,B)		
EX(B,D)		
X(D)		
EX(A,C)		
EX(C,F)		
X(F)		
		X(C)
		UE(A,C)
		X(F)
		U(F)
U(A,B)		
U(D)		
U(A)		
U(0,A)		
U(B,D)		
U(C,F)	U(B)	
		U(C)



Here, for each edge, we add a dummy item. Eg. For edge connecting A, B it is < A, B>. Now, if we look at the instances which caused anamolies in first place, i.e) X(B), X(C) of T_1 are not even present now. Instead, they are replaced by EX(A,B) and EX(C,F) respectively. Now, irrespective of how long T_2 holds B (Gap between X(B), U(B)) and T_3 holds C (Gap between X(C), U(C)), T_1 can occur independently hence promoting concurrency or parallelism.

CS-354

End-Sem Quiz (Numericals)

Q1) Given,

Disk rotates at 10,000 rpm

50,000 tracks per surface,

120 sectors per track,

4 platters,

Avg. Seek time = 5 ms

Sector size = 512 Bytes

Block size = 2 sectors = 1 KB.

Position on correct track

Position on correct sector

Required: Avg Seek Time + Avg. Rotational latency time

+ Avg. Block transfer time.

Seek Time = 5ms (Given)

Rotational latency time => Average is for moving half of the sector

$$1 \text{ rotation} = \frac{1}{10,000} \text{ min} = \text{Full track}$$

$$\frac{1}{2} \text{ rotation} = \frac{1}{2} \times \frac{60}{10,000} \times 1000 \text{ ms} = \text{Half track}$$

$$\Rightarrow \underline{\underline{3\text{ms}}}$$

Data transfer time :

Generally, the transfer rate ranges from 25 MB/s to 40 MB/s

And a block here is 1 KB.

$\therefore \quad 1\,MB = \frac{1}{25}$ sec

$1\,KB = \frac{1}{25} \times \frac{1000}{1024}$ ms ≈ 0.03 ms

Total time $= 5 + 3 + 0.03 = \boxed{8.03 \text{ ms}}$

3) In a B^+ tree, a node is usually fitted into a block. We also know, that a node should contain (N-1) values and N keys.

$$\therefore \quad 4N + 8(N-1) = 512$$

$$\Rightarrow \quad 12N - 8 = 512 \quad \Rightarrow \quad 12N = 520$$

$$N = \frac{520}{12} \approx 43.33$$

$\therefore \boxed{43}$ is the ideal degree of node.

9)

	T_1	T_2	
10 units	X(A)		

		X(B)
		U(B)
			150 units

10 units	X(B)
	X(C)		

Given, T_1 requires X(A), X(C) while T_2 requires X(B)

Both of these are non-conflicting. And it is also given that data dependency is A → B → C. Now T_1 will execute operations on A taking 10 units. Next T_2 takes 150 units for B. T_1 takes 10 from then onwards for C. T_1 waits for U(B) as protocal requires lock on B for X(C).