U6 Deadlock Handling and Data Recovery

Friday, October 29, 2021 14:23

1. What is Deadlock? How to deal with Deadlock?

a. A system is in a deadlock state if there exists a set of transactions such that Every transaction in the set is waiting for another transaction in the set.

2. Give different Deadlock Prevention Strategies.

- Deadlock prevention protocols ensure that the system will never enter into a deadlock state.
- Some prevention strategies :
  - i. locks all its data items before it begins execution.
    1) The simplest scheme, requires that each transaction locks all its data items before it begins execution.
    2) either all are locked in one step or none are locked.
    3) There are two main disadvantages to this protocol:
       a) it is often hard to predict, before the transaction begins, what data items need to be locked;
       b) data-item utilization may be very low,
    4) since many of the data items may be locked but unused for a long time.
  - ii. Impose an ordering of all data items.
    1) Another approach for preventing deadlocks is to impose an ordering of all data items
    2) To require that a transaction lock data items only in a sequence consistent with the ordering.
    3) One such scheme seen in the tree protocol, which uses a partial ordering of data items.
  - iii. use preemption and transaction rollbacks.
    1) In preemption, when a transaction T2 requests a lock that transaction T1 holds,
    2) the lock granted to T1 may be preempted by rolling back of T1, and granting of the lock toT2.
    3) They use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation.
       a) Wait-Die Scheme
       b) Wound-Wait Scheme

3. Explain Deadlock Detection and Recovery mechanisms.

a. Recovery from Deadlock

i. When a detection algorithm determines that a deadlock exists,
ii. the system must recover from the deadlock.
iii. The most common solution is to roll back one or more transactions to break the deadlock.
iv. Three actions need to be taken:
   1) Selection of a victim
      a) Given a set of deadlocked transactions,
      b) we must determine which transaction (or transactions) to roll back to break the deadlock.
      c) We should roll back those transactions that will incur the minimum cost.
   2) Rollback
      - Once we have decided that a particular transaction must be rolled back,
      - we must determine how far this transaction should be rolled back.

- The simplest solution is a total rollback:
- Abort the transaction and then restart it.
- However, it is more effective to roll back the transaction only as far as necessary to break the deadlock.
- Such partial rollback requires the system to maintain additional information about the state of all the running transactions

3) Starvation
   a) In a system where the selection of victims is based primarily on cost factors,
   b) it may happen that the same transaction is always picked as a victim.
   c) As a result, this transaction never completes its designated task,
   d) Thus there is a starvation.
   e) We must ensure that transaction can be picked as a victim only a (small) finite number of times.

4. Explain the following mechanisms

i. Both the wound–wait and the wait–die schemes avoid starvation
ii. Both in wait-die and in wound-wait schemes, a rolled back transactions is restarted with its original timestamp.
iii. Older transactions thus have precedence over newer ones, and starvation is hence avoided.

a. Wait-Die

i. Non-preemptive
ii. In this scheme, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur —
   1) If TS(Ti) < TS(Tj) —
      a) is requesting a conflicting lock, it is older than Tj
      b) then Ti is allowed to wait until the data-item is available.
   2) If TS(Ti) > TS(Tj) —
      a) Ti is younger than Tj — then Ti dies.
      b) Ti is restarted later with a random delay but with the same timestamp.
      c) This scheme allows the older transaction to wait but kills the younger one

b. Wound-wait

i. In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur —
   1) If TS(Ti) < TS(Tj),
      a) then Ti forces Tj to be rolled back
      b) that is Ti wounds Tj.
      c) Tj is restarted later with a random delay but with the same timestamp.
   2) If TS(Ti) > TS(Tj),
      a) then Ti is forced to wait until the resource is available.
      b) This scheme, allows the younger transaction to wait;
      c) but when an older transaction requests an item held by a younger one,
      d) the older transaction forces the younger one to abort and release the item.
      e) It may have fewer rollbacks than wait-die scheme.
      f) It is a counterpart to the wait–die scheme.
ii. In both the cases, the transaction that enters the system at a later stage is aborted.
iii. Whenever the system rolls back transaction,
iv. it is important to ensure that there is no starvation
v. that is, no transaction gets rolled back repeatedly

vi. and is never allowed to make progress.

5. What is starvation? Give different techniques to avoid starvation.

6. What are the different types of failure?

7. Explain the following terms

a. Volatile Storage

i. Information residing in volatile storage does not usually survive system crashes.
ii. Examples of such storage are main memory and cache memory.
iii. Access to volatile storage is extremely fast,
iv. because of the speed of the memory access itself,
v. and because it is possible to access any data item in volatile storage directly.

b. Non-volatile Storage

i. Information residing in nonvolatile storage survives system crashes.
ii. Examples of such storage are disk and magnetic tapes.
iii. Disks are used for online storage, whereas tapes are used for archival storage.
iv. Both are subject to failure (for example, head crash), which may result in loss of information.
v. At the current state of technology, non-volatile storage is slower than volatile storage by several orders of magnitude.

c. Stable Storage

i. Information residing in stable storage is never lost
ii. Although stable storage is theoretically impossible to obtain,
iii. it can be closely approximated by techniques that make data loss extremely unlikely

8. Explain how stable storage can be implemented?

9. Explain Log-based Recovery Mechanism.

10. Explain Deferred database modification.

a. The deferred database modification scheme records all modifications to the log, but defers all the writes to - after partial commit.
b. Assume that transactions execute serially
c. Transaction starts by writing record to log.
d. A write(X) operation results in a log record being written, where V is the new value for X
e. Note: old value is not needed for this scheme
f. The write is not performed on X at this time, but is deferred.
g. When Ti partially commits, is written to the log
h. Finally, the log records are read and used to actually execute the previously deferred writes.
i. During recovery after a crash, a transaction needs to be redone if and only if both and are there in the log.
j. Redoing a transaction Ti (redo Ti) sets the value of all data items updated by the transaction to the new values.
k. Crashes can occur while
   i. the transaction is executing the original updates, or
   ii. while recovery action is being taken
l. example transactions T0 and T1 (T0 executes before T1):
   i.

T_0: **read** (A)	T_1 : **read** (C)
A: - A - 50	C:-C- 100
Write (A)	**write** (C)
read (B)	
B:- B + 50	
write (B)	

ii. Below we show the log as it appears at three instances of time.

iii.

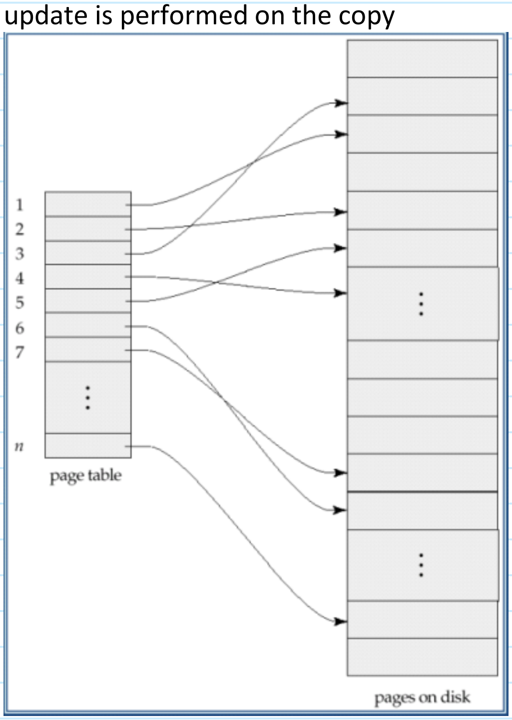
(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 950>	<T_0, A, 950>	<T_0, A, 950>
<T_0, B, 2050>	<T_0, B, 2050>	<T_0, B, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 600>	<T_1, C, 600>
		<T_1 commit>

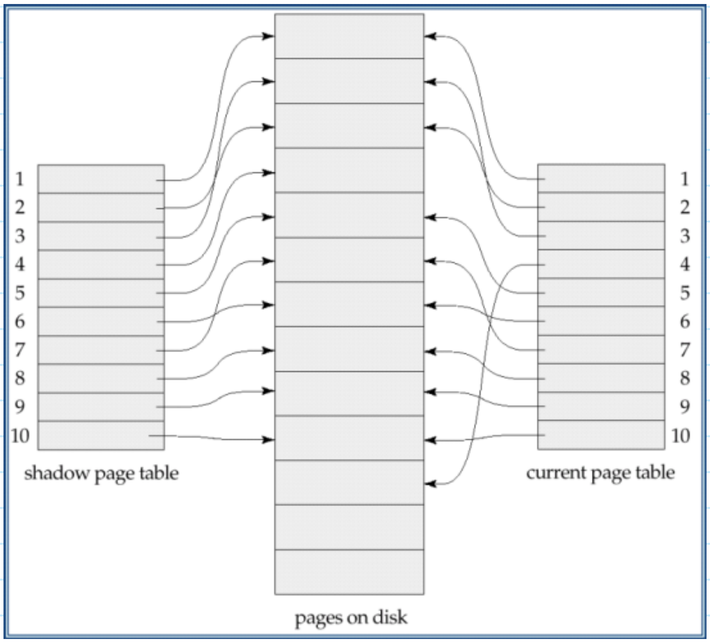
iv. If log on stable storage at time of crash is as in case:
1) No redo actions need to be taken
2) redo(T0) must be performed since is present
3) redo(T0) must be performed followed by redo(T1) since and are present

11. Explain Immediate database modification.

12. Explain Shadow Paging for Data Recovery. Give advantages and disadvantages of shadow paging.

a. Shadow paging is an alternative to log-based recovery;
b. this scheme is useful if transactions execute serially
c. Idea: maintain two page tables during the lifetime of a transaction –the current page table, and the shadow page table
d. Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered.
   i. Shadow page table is never modified during execution
e. To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.
f. Whenever any page is about to be written for the first time – A copy of this page is made onto an unused page. – The current page table is then made to point to the copy – The update is performed on the copy
g.



h. 

i. To commit a transaction :
    i. Flush all modified pages in main memory to disk
    ii. Output current page table to disk
    iii. Make the current page table the new shadow page table, as follows:
        1) keep a pointer to the shadow page table at a fixed (known) location on disk.
        2) to make the current page table the new shadow page table, simply update the pointer to point to current page table on disk
    iv. Once pointer to shadow page table has been written, transaction is committed.
    v. No recovery is needed after a crash — new transactions can start right away, using the shadow page table.
    vi. Pages not pointed to from current/shadow page table should be freed (garbage collected).

j. Advantages of shadow-paging over log-based schemes –
    i. no overhead of writing log records
    ii. recovery is trivial

k. Disadvantages
    i. Copying the entire page table is very expensive
        1) Can be reduced by using a page table structured like a B+ -tree
        2) No need to copy entire tree, only need to copy paths in the tree that lead to updated leaf nodes
    ii. Commit overhead is high even with above extension
        1) Need to flush every updated page, and page table
    iii. Data gets fragmented (related pages get separated on disk)
        1) After every transaction completion, the database pages containing old versions of modified data need to be garbage collected
    iv. Hard to extend algorithm to allow transactions to run concurrently
        1) Easier to extend log based schemes

l.

13. Compare Deferred database modification with Immediate database modification

14. Explain use of checkpoint for data recovery