

Questions for Final DBMS Examination Chapter wise

From Chapter 6

Introduction to Transaction Processing Concepts and Theory

Q1. ----- of data - a field, a record , or a whole disk block that measure the size of the data item. [A]

A. Granularity B. Locking C. Concurrency D. None

Q2. A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed/aborted is called ----- [D]

A. Serializability B. Conflict Serializability C. Scheduling
D. Strict scheduling

Q3. ----- property, once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

A.Atomicity B.Consistency C. Isolation D.Durability [D]

Q4. Consider the following schedule S of transactions T1, T2, T3, T4:

Which one of the following statements is CORRECT?

T1	T2	T3	T4
Writes(X) Commit	Reads(X) Writes(Y) Reads(Z) Commit	Writes(X) Commit	 Reads(X) Reads(Y) Commit

- (A) S is conflict-serializable but not recoverable
- (B) S is not conflict-serializable but is recoverable
- (C) S is both conflict-serializable and recoverable
- (D) S is neither conflict-serializable nor is it recoverable

Answer: (C)

Q5. If a transaction obtains an exclusive lock on a row, it means that the transaction wants to that row. [B]

A. select B. update C. view D. read

Chapter-7

Concurrency Controlling Techniques

Q1. If transaction has ----- lock on item, can read but not update item.

A. Exclusive B. Shared C. Timestamp D. Checkpointing [B]

Q2. A transaction unlocks its locked data items one at a time and releases locks but cannot acquire any new locks is ----- phase.

A. Growing B. 2PL C. Shrinking D. ALL [c]

Q4. ----- occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further while other transaction continue normally.

A. Deadlock B. Starvation C. Abort D. Rollback [B]

Q5. When the transaction finishes the final statement the transaction

enters into-----

- A. Active state B. Committed state C. Partially committed state
D. Abort state [C]

Part-II

Q1. What is the purpose of Database Recovery? And Explain types of failures briefly. (Short Ans.Quyestion)

Ans:-

To bring the database into the last consistent state, which existed prior to the failure.

- To preserve transaction properties (Atomicity & Durability).

The recovery manager of a DBMS is responsible to ensure

atomicity by undoing the action of transaction that do not commit

Durability by making sure that all actions of committed transaction survive system crash

Types of Failures

The database may become unavailable for use due to :

Transaction failure: Transactions may fail because of incorrect input, deadlock.

System failure: System may fail because of addressing error, application error, operating system fault, RAM failure, etc.

Media failure: Disk head crash, power disruption, etc.

Q2. If no concurrency mechanism in the database system?

Ans:-

Problems of Concurrency Control:

When concurrent transactions are executed in an uncontrolled manner, several problems can occur.

The concurrency control has the following three main problems:

Lost updates.

Dirty read (or uncommitted data).

Unrepeatable read (or inconsistent retrievals).

Lost Update Problem :

A lost update problem occurs when two transactions that access the same database items have their operations in a way that makes the value of some database item incorrect.

In other words, if transactions T1 and T2 both read a record and then update it, the effects of the first update will be overwritten by the second update.

Dirty Read Problem :

A dirty read problem occurs when one transaction updates a database item and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value. In other words, a transaction T1 updates a record, which is read by the transaction T2.

Then T1 aborts and T2 now has values which have never formed part of the stable database.

Inconsistent Retrievals Problem :

Unrepeatable read (or inconsistent retrievals) occurs when a transaction calculates some summary (aggregate) function over a set of data while other transactions are updating the data.

The problem is that the transaction might read some data before they are changed and other data after they are changed, thereby yielding inconsistent results.

In an unrepeatable read, the transaction T1 reads a record and then does some other processing during which the transaction T2 updates the record. Now, if T1 rereads the record, the new value will be inconsistent with the previous value.

Q3. Write short notes on Wait-die, Wound-wait.

Ans:-

Wait-die:-

- If T_i has higher priority, it is allowed to wait; otherwise it is aborted.
- An older transaction is allowed to wait on a younger transaction.
- A younger transaction requesting an item held by an older transaction is aborted
- If $TS(T_i) < TS(T_j)$, then (T_i older than T_j) T_i is allowed to wait.

Wound-wait:-

The opposite of wait-die

If T_i has higher priority, abort T_j ; otherwise T_i waits.

A younger transaction is allowed to wait on an older one

An older transaction requesting an item held by a younger transaction preempts the younger transaction by aborting it.

If $TS(T_i) < TS(T_j)$, then (T_i older than T_j), Abort T_j (T_i wounds T_j) and restart T_j later with the same timestamp

Otherwise (T_i younger than T_j) T_i is allowed to wait

Q4. 3 limitations of serial schedule

- Serial schedule:
 - A schedule S is serial if, for every transaction T participating in the schedule, all the operations of T are executed consecutively in the schedule. Otherwise, the schedule is called nonserial schedule.
 - *For example*, in the banking example suppose there are two transaction where one transaction calculate the interest on the account and another deposit some money into the account. hence the order of execution is important for the final result.
- Serializable schedule:
 - a schedule whose effect on any consistent database instance is identical to that of some complete serial schedule over the set of *committed* transactions in S .

Q5. Sc: r1(X); w1(X); r2(X); r1(Y); w2(X); c2; a1;

Ans:-

- Consider the two (partial) schedules *Sc* and *Sd* that follow:
Sc: r1(X); w1(X); r2(X); r1(Y); w2(X); c2; a1;
Sd: r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); c1; c2;
Se: r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); a1; a2;
- *Sc* is not recoverable because *T2* reads item *X* from *T1*, but *T2* commits before *T1* commits.
- The problem occurs if *T1* aborts after the *c2* operation in *Sc*, then the value of *X* that *T2* read is no longer valid and *T2* must be aborted *after* it is committed, leading to a schedule that is *not recoverable*. For the schedule to be recoverable, the *c2* operation in *Sc* must be postponed until after *T1* commits, as shown in *Sd*.
- If *T1* aborts instead of committing, then *T2* should also abort as shown in *Se*, because the value of *X* it read is no longer valid.
- In *Se*, aborting *T2* is acceptable since it has not committed yet, which is not the case for the non recoverable schedule *Sc*.

They belongs to different transaction

They access the same data item X

At least one of the operation is a write_Item(X)

Eg. Sa: r1(X); r2(x); w1(X); r1(Y); W2(X); W1(Y);

r1(X) and w2(X)

$r_2(X)$ and $w_1(X)$; } Conflicting
operations
 $W_1(X)$ and $w_2(X)$

$r_1(X)$ and $r_2(X)$
 $W_2(X)$ and $w_1(Y)$

} No Conflict,
why?
 $R_1(x)$ and $w_1(x)$

Sc: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(x)$; c_2 ; a_1 ; not recoverable

Part-III

Q1. What is query optimization? And Explain briefly every step when processing a high-level query with neat block diagram.

Ans:-

The activity of choosing a single “efficient” execution strategy (from hundreds) as determined by database catalog statistics.

Which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?

How do operations pass data (main memory buffer, disk buffer,...)?

Will this plan minimize resource usage? (CPU/Response Time/Disk

