**Assignment#4**

**MSIT 630 Database Systems (Summer, 2018)**

**Total: 30 points**

***Due: 7/27/2018 11:59PM***

1, List the ACID properties. Explain the usefulness of each. (**4 points**)

Atomicity - Either all operations of the transaction are reflected properly in

the database, or none are. The database system keeps track of old values of any data on which a transaction performs a write. If the transaction does not complete its execution, the database system restores the old values from the log to make it appear as the transaction never

executed. The atomicity property requires that we execute a transaction to completion. It is the responsibility of the transaction recovery subsystem of a DBMS to ensure atomicity.

Consistency - Execution of a transaction in isolation preserves the consistency of the database. Without the consistency requirement, money could be created or destroyed by the transaction. If the database is consistent before the execution of the transaction, the database remains consistent after the execution of the transaction. This allows for the completion of transactions without any interruptions.

Isolation - Even if the consistency and atomicity properties are ensured for each transaction, if several transactions are executed concurrently, their operations may interleave in some undesirable way, resulting in an inconsistent state. Isolation property ensures that a transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently.

Durability - After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures. Durability ensures that changes applied to the database by a committed transaction must persist in the database.

2, Consider the following two transactions: (**10 points**)

*T*13: read(*A*);

read(*B*);

**if** *A* = 1 **then** *B* := *B* - 1;

write(*B*).

*T*14: read(*B*);

read(*A*);

**if** *B* = 1 **then** *A* := *A* - 1;

write(*A*).

Let the consistency requirement be *A* = 1 or *B* = 1, with *A* = 1 and *B* = 1 as the initial values.

1. Show that every serial execution involving these two transactions preserves the consistency of the database.

|  |  |  |
| --- | --- | --- |
| **Case 1** | **A** | **B** |
| Initially | 1 | 1 |
| After T13 | 1 | -1 |
| After T14 | 1 | -1 |

|  |  |  |
| --- | --- | --- |
| **Case 2** | **A** | **B** |
| Initially | 1 | 1 |
| After T13 | -1 | 1 |
| After T14 | -1 | 1 |

1. Show a concurrent execution of *T*13 and *T*14 that produces a nonserializable schedule.

T13

T14

T15

T16

T17

1. Is there a concurrent execution of *T*13 and *T*14 that produces a serializable schedule?

|  |  |
| --- | --- |
| T13 | T14 |
| Read (A) |  |
|  | Read (B) |
|  | Read (A) |
| Read (B) |  |
| If A = -1 Then B = B -1 |  |
|  | If B = -1 Then A = -1 |
|  | Write (A) |
| Write (B) |  |
| Answer: No concurrent Execution | |

3, What is the two-phase locking protocol? The two-phase protocol requires that each transaction issue lock and unlock requests in two phases: Growing phase. (A transaction may obtain locks but may not release any lock). Shrinking phase. (A transaction may release locks but may not obtain any new locks).

What is the strict two-phase locking protocol? The strict two-phase protocol requires not only the locking be two phase, but also that all exclusive mode locks to be taken by a transaction

be held until that transaction commits. This requirement ensures that any data

written by an uncommitted transaction are locked in exclusive mode until the

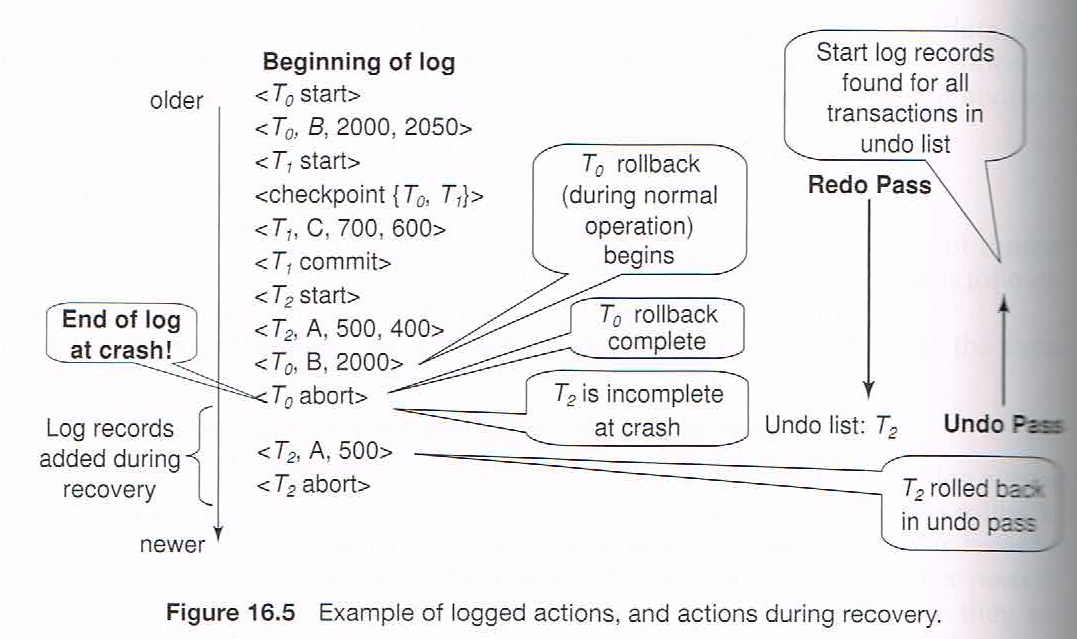
transaction runs, preventing any other transaction from reading.

What is the rigorous two-phase locking protocol? Rigorous two-phase locking is the same as two-phase locking, but it will hold all locks until the transaction has been completed or aborted.

What benefit does strict two-phase locking protocol provide? As it only produces cascadeless schedules, the recovery can be achieved fairly easy.

What benefit does rigorous two-phase locking protocol provide? As mentioned above, rigorous two-phase locking has the same advantages as two-phase locking plus it has the properties that two conflicting transactions, the commit order is their seriazability order. (**8 points**)

4. Consider the log in Figure 16.5 (page 738). Suppose there is a crash just before the < *T*0 abort> log record is written out. Explain what would happen during recovery. (**8 points**)

****

**Redo Phase**

1. **Undo T0, T1**
2. **Start checkpoint and perform redo operation**
3. **C = 600**
4. **Remove T1 from Undo**
5. **T2 is added to Undo list encountering <T2 start> record.**
6. **A = 400**
7. **B = 2000**

**Undo Phase**

1. **Undo T0, T2**
2. **Scan log backwards**
3. **A = 500 Redo Record <T2, A, 500>**
4. **Output <T2 abort>**
5. **B = 2000**
6. **Output <T0 abort>**

**A = 500**

**B= 2000**

**C = 600**

**Log Records added during the recovery process are as follow:**

**<T2, A, 500>**

**<T2 Abort>**

**<T0, B, 2000>**

**<T0 Abort>**