**Assignment#4**

**MSIT 630 Database Systems (Summer, 2019)**

**Total: 30 points**

***Due: 7/28/2019 11:59PM***

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

**Atomic – The entire transaction is submitted to the database or it all fails. Almost like a one for all and all for one mentality. Either it all works or none of it works. There is no partial execution of a transaction only the whole occurrence.**

**Consistent- This means that transactions should always return the same data and is consistent. Such as a query of a primary key of table will always return the same primary key. Another example of keeping data consistent would be transferring assets where the sender could have validation constraints verifying pre and post transaction balances and if invalid void the transaction. Keeping both the sender and receivers balances both consistent.**

**Isolated- Transactions don’t effect and are un-effected by other transactions. Transactions are isolated from each other and two transactions occurring at the same time can damage database integrity. To keep transactions isolated they should be interpreted by the database sequentially. Avoiding mid-air collisions.**

**Durable – This means that data is written and committed to the database even in an act of failure. How durable is your database to hardware and network failure? Even if a failure is to occur data should be written to a disk till the system is back up.**

**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 [T13 🡪 T14] consistency of A as a value of 1.**

|  |  |  |
| --- | --- | --- |
| **Values** | **A** | **B** |
| **Initial Values** | **1** | **1** |
| **After T13** | **1** | **0** |
| **After T14** | **1** | **0** |

**Consistency of A as a value of 1 is met.**

**Case 2 [T14 🡪 T13] consistency of B as a value of 1.**

|  |  |  |
| --- | --- | --- |
| **Values** | **A** | **B** |
| **Initial Values** | **1** | **1** |
| **After T14** | **0** | **1** |
| **After T13** | **0** | **1** |

**Consistency of B as a value of 1 is met.**

**Proving the consistency of the database when using these required values A and B.**

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

**For concurrent executions of T13 and T14 that makes a non-serializable schedule the system should execute transactions in a mirrored manner. Such as this.**

|  |  |
| --- | --- |
| **T13** | **T14** |
| **Read A** | **Read B** |
| **Read B** | **Read A** |
| **If A= 1, then B = B-1** | **If B=1, then A=A-1** |
| **Write B** | **Write A** |

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

**There is no concurrent execution of T13 and T14 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= A-1** |
|  | **Write A** |
| **Write B** |  |
| **No parallel execution.** | |

**3, What is the two-phase locking protocol?**

**This protocol has two phases, a growing phase in which a transaction may accept new locks but cannot unlock. As well as a shrinking phase where transactions can unlock but may not except new locks. This is how transactions issue lock and unlock requests.**

**What is the strict two-phase locking protocol?**

**Takes the two phase locking but also adds that exclusive mode locks be held until the transactions commitment. This ensures that data from uncommitted transactions are in exclusive mode until the transaction is ran.**

**What is the rigorous two-phase locking protocol?**

**Takes the two phase locking but also holds all exclusive mode locks until the transaction has been completed or aborted. Used to serialize transactions in order of their committal.**

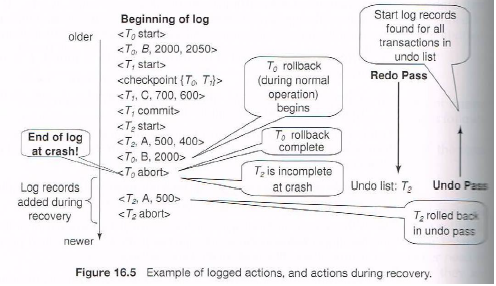
**What benefit does strict two-phase locking protocol provide?**

**This produces only schedules without cascades. It provides a quick and easy recovery process along with a low rollback overhead. It will allow any acceptable concurrency level.**

**What benefit does rigorous two-phase locking protocol provide? (8 points)**

**Provides a feature to operate two simultaneous transactions basing their commitment order directly with their serializability order. Effectively taking care of conflicting transactions.**

**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)**

****

**There will be a redo phase and an undo phase. The redo phase focuses on the records after that last checkpoint while the undo phase scans the logs backwards and creates a redo only log.**

**Redo phase**

**- Undo T0, T1.**

**- Start checkpoint and redo operations.**

**- C= 600 from redo record <T1,C,700,600>.**

**- Remove T1 from the Undo.**

**- T2 Is added to the undo list after log record identified <T2 start>.**

**- A = 400 and B = 2000.**

**Undo Phase.**

* **Under T0, T2.**
* **- Scan log backwards**
* **A= 500 from redo record <T2,A,500>**
* **Output <T2 abort>**
* **B = 2000**
* **Output <T0 abort>**

**After recovery process is finished.**

**A= 500**

**B=2000**

**C=600**

**The added log records.**

**<T2,A,500>**

**<T2 abort>**

**<T0,B,2000>**

**<T0 abort>**