South China Agricultural University
Computing Science
2005-2006

<u>Database Systems</u>
<u>Tutorial Set 1 (ANSWERS)</u>

Tutorial Questions (Transaction Processing)

1. List the ACID properties. Explain the usefulness of each.

Ans:
Consistency
Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database. This is typically the responsibility of the application programmer who codes the transactions.

Atomicity
Either all operations of the transaction are reflected properly in the database, or none are. Clearly lack of atomicity will lead to inconsistency in the database.

Isolation
When multiple transactions execute concurrently, it should be the case that there is no interference between transactions. Each transaction is unaware of other transactions executing concurrently with it. The property that concurrent schedules take the system from one consistent states to another.

Durability
After a transaction completes successfully, the changes it has made to the database persist even if there are system failures.

2. During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition may occur.

Ans:
The possible sequences of states are:
Active → Partially Committed → Committed
This is the normal sequence a successful transaction will follow. After executing all its transactions it enters the partially committed state. After enough information has been written to disk, the transaction finally enters the committed state.

Active → Partially Committed → Aborted
After executing the last statement of the transaction, it enters the partially committed state. But before enough recovery information is written to disk, a hardware failure may occur destroying the memory contents. In this case, the changes which it made to the database are undone, and the information enters the aborted state.

Active → Failed → Aborted
After the transaction starts, if it is discovered at some point that normal execution cannot continue (either due to internal program errors or external errors), it enters the failed state. It is then rolled back, after which it enters the aborted state.

3. Explain the distinction between the serial schedule and serializable schedule.

Ans:

A schedule in which all the instructions belonging to one single transaction appear together is called a serial schedule. A serializable schedule has a weaker restriction that it should be equivalent to some serial schedule.

4. Consider the following two transactions

```
T1:     Read(A)
        Read(B)
        If A = 0 then B := B + 1
        Write (B)

T2:     Read(B)
        Read(A)
        If B = 0 then A := A + 1
        Write (A)
```

Let the consistency requirement be A = 0 or B = 0 with A = B = 0 the initial values

a. Show that every serial execution involving these two transactions preserve the consistency of the database.
b. Show a concurrent execution of T1 and T2 that produces a non-serializable schedule.
c. Is there a concurrent schedule of T1 and T2 that produces a serializable schedule?

Ans:

a.

There are two possible executions: T1 → T2 or T2 → T1

Case 1: T1 → T2	A	B
Initially	0	0
After T1	0	1
After T2	0	1

Consistency met as A = 0

Case 2: T2 → T1	A	B
Initially	0	0
After T2	1	0
After T1	1	0

Consistency met as B = 0

b.

Any interleaving of T1 and T2 results in a non-serializable schedule

For example:

T1	T2
Read(A)	
	Read(B)
	Read(A)

```
Read(B)
If A = 0 then
B = B + 1
                        If B = 0 then
                        A = A + 1
                        Write(A)
Write(B)
```

c. There is no concurrent execution resulting in a serializable schedule.