Distributed Database System (ECS-242/602)

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 03087792217**

Objectives: **Learning objectives of this lecture are**

- **Properties of Transaction**
  - **Atomicity**
  - **Consistency**
    - **Degree/Level of Consistency**
  - **Isolation**
    - **Degree/Level of Isolation**
  - **Durability**

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 03087792217**

Overview:

In this lecture we will learn what are the properties of transaction.

Properties of Transactions

Following are four properties of transaction which are also known as ACID (atomicity, consistency, isolation, durability).

1. **Atomicity**
2. **Consistency**
3. **Isolation**
4. **Durability.**

Together, these are commonly referred to as the ACID properties of transactions. They are not entirely independent of each other; usually there are dependencies among them as we will indicate below. We discuss each of these properties in the following sections.

1- Atomicity

Atomicity refers to the fact that a transaction is treated as a unit of operation. Therefore, either all the transaction's actions are completed, or none of them are. This is also known as the "all-or-nothing property". Notice that we have just extended the concept of atomicity from individual operations to the entire transaction. Atomicity requires that if the execution of a transaction is interrupted by any sort of failure, the DBMS will be responsible for determining what to do with the transaction upon recovery from the failure. There are, of course, two possible courses of action: it can either be terminated by completing the remaining actions, or it can be terminated by undoing all the actions that have already been executed. One can generally talk about two types of failures. A transaction itself may fail due to input data errors, deadlocks, or other factors. In these cases, either the transaction aborts itself or the DBMS may abort it while handling deadlocks

2- Consistency

The consistency of a transaction is simply its correctness. In other words, a transaction is a correct program that maps one consistent database state to another. Which means database must be in consistent state after completion of transaction. Verifying that transactions are consistent is the concern of integrity enforcement. Ensuring transaction consistency as defined at the beginning of this chapter, on the other hand, is the objective of concurrency control mechanisms.

Distributed Database System (ECS-242/602)

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 03087792217**

Dirty Data:

There are four levels of consistency. Dirty data is a term we will use to define level of consistency.

"Dirty data refers to data values that have been updated by a transaction prior to its commitment". Any Data item in a transaction will be considered "Dirty Data" which is written by a transaction until it is committed by transaction. By data item we mean those values which are read and written during a transaction. For example; Read(x), here x is a data item which will be considered as "Dirty Data" when transaction will write x. and x will be dirty till commit statement is issued by transaction.

Before going into the Degrees of consistency, we will discuss some points on the basis of which degree will be defined. Below are four points on which we will define degrees of consistency.

Then, based on the concept of dirty data, the four levels are defined as follows:

1- **T does not overwrite dirty data of other transactions.**

2- **T does not commit any writes until it completes all its writes [i.e., until the end of transaction (EOT)].**

3- **T does not read dirty data from other transactions.**

4- **Other transactions do not dirty any data read by T before T completes.**

NOTE: "T" is the transaction for which we will be checking degree of consistency. Which means each transaction will have its own degree of consistency. Degree of consistency of a transaction does not depend upon consistency of other transaction.

On the basis of above four points, transaction will be at level

- Inconsistent (Point 1) ✘
- Degree 0 (point 1) ✓, (point 2) ✘
- Degree 1 (point 1,2) ✓, (point 3) ✘
- Degree 2 (point 1,2,3) ✓, (point 4) ✘
- Degree 3 (point 1,2,3,4) ✓

Of course, it is true that a higher degree of consistency encompasses all the lower degrees. The point in defining multiple levels of consistency is to provide application programmers the flexibility to define transactions that operate at different levels. Consequently, while some transactions operate at Degree 3 consistency level, others may operate at lower levels and may see, for example, dirty data.

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 03087792217**

3- Isolation

Isolation is the property of transactions that requires each transaction to see a consistent database at all times. In other words, an executing transaction cannot reveal its results to other concurrent transactions before its commitment. There are a number of reasons for insisting on isolation. One has to do with maintaining the inter-consistency of transactions. If two concurrent transactions access a data item that is being updated by one of them, it is not possible to guarantee that the second will read the correct value.

Let us take example of previous lecture.

x=100, y=50 (Database Values)

Time	T1	T2
t0	Read (x)	
t1	x = x + 20	Read(x)
t2	Read (y)	x = x + 50
t3	x = x + y	Write(x)
t4	Write(x)	Commit
t5	Commit	

x=170, y=50 (Database Values)

Here we see value of x is 170, which must be “220” because “120” is being added into x but due to not isolated, value of x is corrupted because partial write of T1 is visible to T2. If we execute above two transaction independently then situation will be different.

x=100, y=50 (Database Values)

T1
Read (x)
x = x + 20
Read (y)
x = x + y
Write(x)
Commit

x=170, y=50 (Database Values)

T2

Mr. Abdul Rehman email id: abdul@northern.edu.pk Whatsapp# 03087792217

Read(x)
x = x + 50
Write(x)
Commit

x=220, y=50 (Database Values)

Here we can see that value of x is "220". Which is correct because both transactions are executed independently of each other. So, isolation requires that transactions partial written data must not be visible to other transaction at all.

Ensuring isolation by not permitting incomplete results to be seen by other transactions, as the previous example shows, solves the lost updates problem. This isolation has been called cursor stability. A second reason for isolation is cascading aborts. If a transaction permits other to see its incomplete results before committing and then decides to abort, any transaction that has read its incomplete values will have to abort as well. This chain can easily grow and impose considerable overhead on the DBMS. It is possible to treat consistency levels discussed in the preceding section from the perspective of the isolation property (thus demonstrating the dependence between isolation and consistency). As we move up the hierarchy of consistency levels, there is more isolation among transactions. Degree 0 provides very little isolation other than preventing lost updates. However, since transactions commit write operations before the entire transaction is completed (and committed), if an abort occurs after some writes are committed to disk, the updates to data items that have been committed will need to be undone. Since at this level other transactions are allowed to read the dirty data, it may be necessary to abort them as well. Degree 2 consistency avoids cascading aborts. Degree 3 provides full isolation which forces one of the conflicting transactions to wait until the other one terminates. Such execution sequences are called strict and will be discussed further in the next chapter. It is obvious that the issue of isolation is directly related to database consistency and is therefore the topic of concurrency control.

There are some phenomena which will be used to define levels of isolation. These phenomena are discussed below.

Dirty Read:

As defined earlier, dirty data refer to data items whose values have been modified by a transaction that has not yet committed. Consider the case where transaction T1 modifies a data item value, which is then read by another transaction T2 before T1 performs a Commit or Abort. In case T1 aborts, T2 has read a value which never exists in the database.

A precise specification4 of this phenomenon is as follows (where subscripts

indicate the transaction identifiers)

Non-repeatable or Fuzzy read:

Transaction T1 reads a data item value. Another transaction T2 then modifies or deletes that data item and commits. If T1 then attempts to reread the data item, it either reads a different value or it can't find the data item at all; thus, two reads within the same transaction T1 return different results.

Phantom:

The phantom condition that was defined earlier occurs when T1 does a search with a predicate and T2 inserts new tuples that satisfy the predicate. Again, the precise specification of this phenomenon is (where P is the search predicate)

Based on these phenomena, the isolation levels are defined as follows. The objective of defining multiple isolation levels is the same as defining multiple consistency levels.

Read uncommitted: For transactions operating at this level all three phenomena are possible.

Read committed: Fuzzy reads and phantoms are possible, but dirty reads are not.

Repeatable read: Only phantoms are possible.

Serializable: None of the phenomena are possible.

4- Durability

Durability refers to that property of transactions which ensures that once a transaction commits, its results are permanent and cannot be erased from the database. Therefore, the DBMS ensures that the results of a transaction will survive subsequent system failures. The durability property brings forth the issue of database recovery, that is, how to recover the database to a consistent state where all the committed actions are reflected.