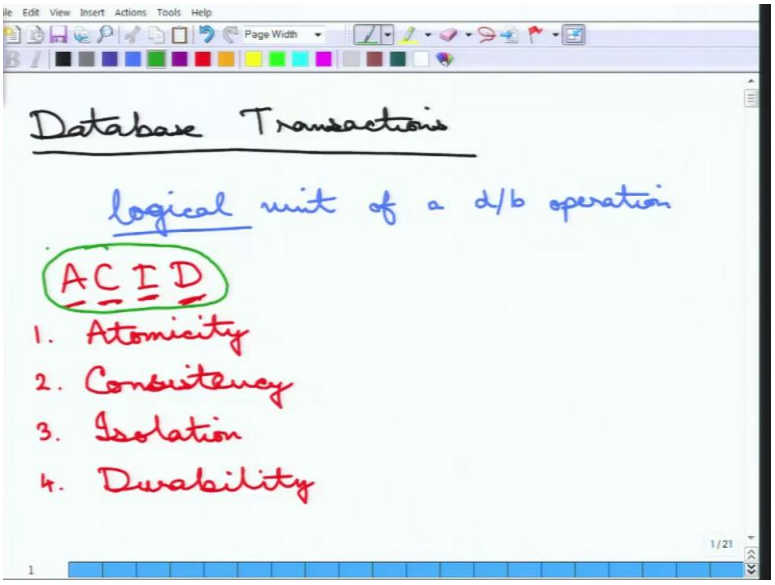
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 28
Database Transactions: Properties and Failures

Welcome, today we will be talking about Database Transactions. So, transactions if you remember at the one logical units of programs, that let us one transfer money from one bank account to the other, let us one buys rail tickets, etcetera. So, this is about database transactions.

(Refer Slide Time: 00:28)



So, the concept of transactions we will cover it in multiple sessions, but one thing to remember is that a transaction is not a program or a part of the database, it is a logical unit of a programs, it is a logical unit of a database function. So, logical unit of a database operation, so some particular set of operations can be set to be constitute in at transactions. So, the complete operation of for example, the complete operation of moving money from, transferring money from one bank account to the other is the, can be called a transaction.

So, only that part may be declared as a transaction within in the database operations. So,

transactions have four very, very important properties, these are called the acid properties, this is the very famous term. So, that in the context of databases, this is called the acid properties. So, what are the acid properties? The first property in that acid thing is A, A stands for Atomicity. So, the atomicity of a transaction essentially says that either the transaction has completely happened or it has not happened at all.

So; that means, in this banking transaction scenario, either the money has been transferred from A's account to B's account completely or it has not happened at all. Now, what do I mean by completely? See the transfer of money from A to B consists of two operations, two basic operations, debiting money from A's account and crediting to B's account. Now, it cannot happen that only the debit part has gone through or only the credit part has gone through, either both have succeeded or both have failed. So, that is the atomicity of transactions.

The next property is the C part, which is called the consistency. So, the transaction should not change the consistent order of a database. So, if the database was consistent to start with before the transaction started, it should be consistent after the transaction has entered. So, the consistency property for example, for this bank account scenario may be that the total amount of money in A's account and B's account is something which was constant, now after the transaction that should not be changing.

So, suppose A is started of A 1000 rupees and B had 500 rupees and then 100 rupees was transferred. So, the initial amount in A plus B's was 1500 and it should still remain 1500. So, the database should be in the consistent state, this is just an example and then there are other ways of actually defining what consistency is, but that is the property called consistency. The next property is the isolation and this we discussed a little bit towards the beginning of the course.

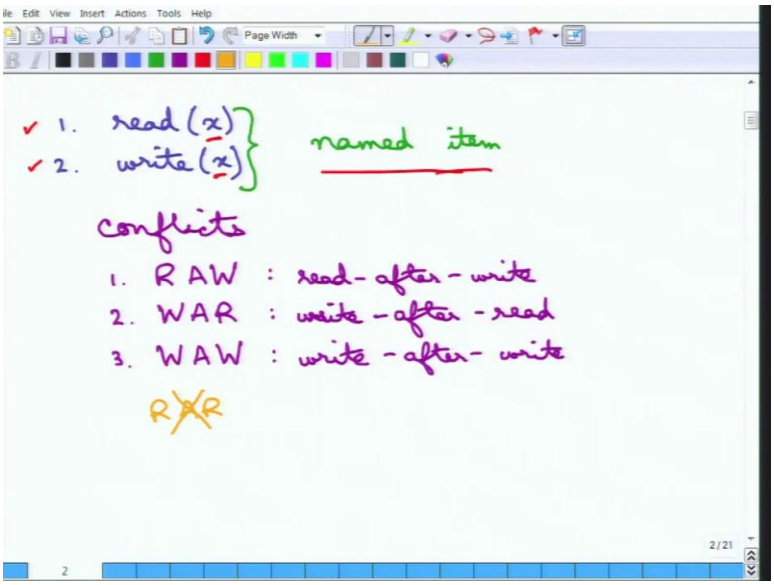
But, essentially this means that multiple transactions may be executing concurrently, but the effect on each one of them is achieved, this is obvious of any other transaction. So, each transaction is apparently happening in isolation of the other transaction. So, the example is let suppose A is transferring money to B and C is transferring money to D. Now, this should not interfere with each other and the effect of whether C is transferring

money to D has succeeded or failed, has got no effect on the transfer of money from A to be B and vice versa, so that is called an isolation.

And the fourth property is the durability. So, the durability part is that, if a transaction finishes successfully, then the effects of the transaction must be durable or must be permanent. So, after this transfer of money has been happened and if the transaction is succeeded successfully, that is even if the database crashes or there are other problems in other transactions and there are other issues, this transfer of money is deemed to be permanent.

So, if A's account has been debited and B's has been credited, after the database again comes up or again the database is being operated, the new state should reflect that A's money has been debited and B's has been credited. So, this is the durability, once it happens, it remains, it is permanent. So, these are the four important properties of transaction and together these are called the acid properties and this is very, very important to understand what the acid property is. Now, to define a little bit more of the transaction, as I say this is the logical unit of a program.

(Refer Slide Time: 05:17)



And the basic two operations of transaction is read and write. Now, what does the

transaction read or write? A transaction is said to read or write, something called a named item. Now, what is a named item? A named item is essentially any logical unit of data can be a named item. So, for example, a bank account with the particular account number, the account may be a named item. So, and in other examples a complete ticket in a ticketing system that may be a named item or etcetera, etcetera.

So, when we are starting database transactions we will forget whether the database is in the relational mode or any other mode or relational for model or any other mode, all we will see is that the database is consisting of some named item that is all. Now, the named item as I said is one logical unit of the data and there may be named items that are covering. So, that the named item may be subsets of each other, etcetera, but that we will not considered here.

So, for the transaction model that we will study, we will just say that there are named items. So, the database is just simply a collection of named items and the transaction either reads a named item or writes the named item. So, if the named item is suppose x then there are two operations, either it read x or it writes to x that is it. Now, the granularity of this named item may be a block, may be a page, may be the entire database, may be a table, etcetera, but that does not matter all we are concerned about is the named item.

And transactions are independent of how you define the named item. It is essentially independent of the granularity in which a named item is defined. So, these are the two basic of operations of read and write and using this... So, just to note that x is the named item variable, so x denotes the variable for a named item and that is all, that is a and now with this read and write there may be conflicts for read and write and we are know what the three basic types of conflicts are. The first one is read after write, then this is write after read and the third one is write after write.

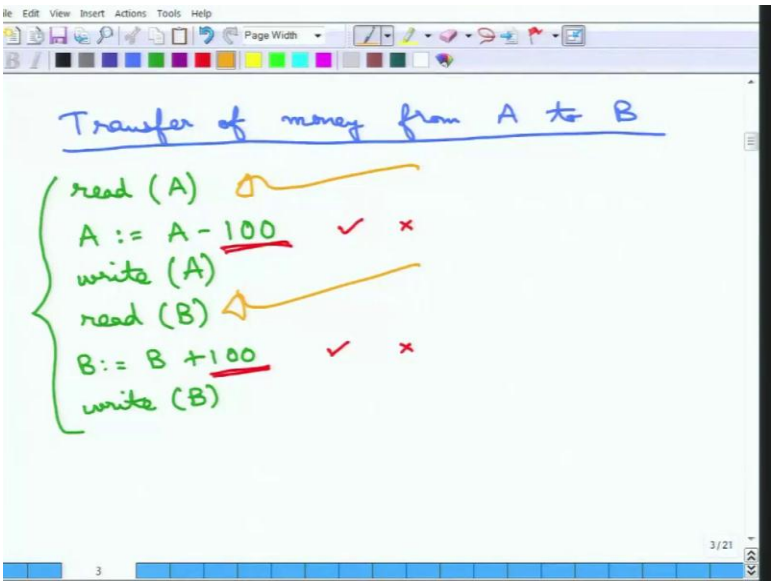
So, this just to complete, this is read after write. So, when this is read after write conflict happens is that, when there are two or more transactions that reads. So, there are two transactions, two or more transactions, one of them reads and the other writes and then whatever the reading, whenever the first transaction tries to read, it should read the value

before the second transaction has retrained. So, that is the read after write conflict and then similarly there is a write after read conflict and we will go over this conflict in much more detail later, but this is basically this three conflicts are essentially saying which operations can conflict.

Now, note that very importantly there is no read after read conflict, there is no read after read conflict, because if there are two transactions that are reading the same item, the x. It does not matter in which order the read, both of them reads the same value, none of them changes the value of the item. So, it does not matter which order the read, so it is not a conflict, so the transactions one and transaction two even if they read the same item, it is not a conflict.

But, otherwise whenever there is a write operation inward we need, which is this read after write, write after read or write after write, they are may be conflicts. Because, the reading and writing has to go in the correct order; otherwise, the read value or the written value may be wrong and we will go over this in much more detail now. But, this is essentially the concept of what the conflicts are. Before we go over the conflicts, let us complete to one example of the transaction to understand this acid properties.

(Refer Slide Time: 09:12)



So, suppose this transfer of money from A's account to B's account. So, transfer of money from A to B, this is the example, this is the transaction that we are doing. Now, what are the properties that matters is that, the first thing is the how do we denote these transaction, the way to denote this transaction is that what will happen is that, the amount... So, whatever amount is in the A's account that will be read, so that is the first operation called read.

Then, suppose A's account will be debited by let us say 100 or this thing is been debited from the A's account. So, that is the new value of A becomes this thing, now this new value needs to be retain. So, this is the write A, then similarly this read the B's account will be read that will be enhanced with that amount 100 and then there will be a write to this B. So, this is the complete description of the transaction in this manner and now we see let us say, what is the idea of atomicity.

The first thing is, how do we define atomicity? Very simply, if this happens then these must happen or if this does not happen, then this does not happen. So, atomicity as I am saying that either A's account is debited and B's account is credited or none of them is happened, it cannot that only one of them as happen that is the atomicity. Otherwise, you see what the problem is, the bank can either take away money indefinitely or generate money indefinitely, so both of which problem occurs.

So, the atomicity ensures that those problems do not happen. So, that is the part about atomicity. Now, consistency, consistency is something about the semantic, is that now you see 100 rupees is taken from A and that the same 100 rupees is deposited to B. So, that is why the sum of A plus B remains constant after the transaction is happening. If 100 rupees was taken from A and only 90 rupees or 110 rupees were credited to B, then that the consistency may suffered.

So, the consistency has to be define, what is the exactly the consistency criterion. Here the criterion is that the total amount of money in A's and B's account is constant. So, that is being maintained, so that is the consistency part, next comes the isolation. So, the isolation is essentially is that if there is another transaction that either reads A or reads B, then to that transaction, now this is a little bit different concept from what we have been

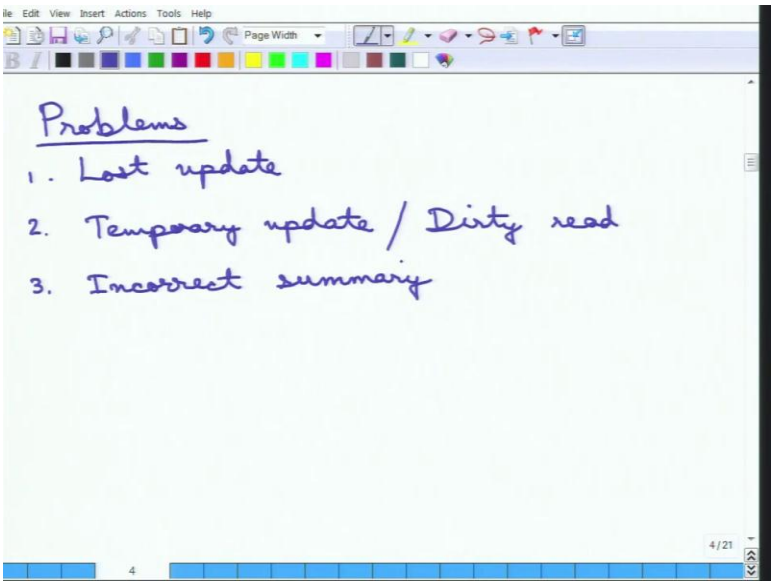
describing isolation.

So, if there is another transaction that either reads A or reads B, then that transaction should be oblivious of what is happening in this thing. So, either it must read the value before any of these two operations have taken places or it must read the value after both of these operations have taken places. So, that is the effect, that is the way to ensure isolation, but isolation essentially means is that, that the other transaction does not need to know that there is another transaction going on here and it is does not care whether this transaction, what stage this transaction is whether it is succeeded or fail. So, that is the isolation property. So, that is the thing about isolation.

And finally, that thing is about durability, the last thing is about durability. Now, if B's account has been credited and B has been sent of a notice by the ((Refer Time: 12:25)) whatever it is and then the database crashes etcetera, after the database recovers it should not be that B's seeing less money B should see the new money that the new balance should be reflected in B's account no matter what after the transaction has been completed.

So, after the transaction is completed, there is a generally as signal given there is generally a state which is declared. So, in the transaction as completed successfully and after that it should be durable it should not it can be reversed. So, that is the point of transactions.

(Refer Slide Time: 13:01)



Now, the transactions may end up in certain problems, there may be certain problems in transactions. So, the first problem is called the problem of last update, so what essentially is last update is that suppose there are two transactions that are both writing to the same data item. So, that what may happen is the update by the first transaction may be over written by the update of the second transaction? So, the essentially the update that is done by the first transaction is lost, because it has been proceeded by some another transaction, so that the last update problem.

The second problem is the temporary update, this is also sometimes called the dirty read. So, what the dirty read problems is that suppose a transaction writes to a particular data item. So, suppose the previous example the transaction writes the new value to A and then it fails before B is being updated. So, then as we have being saying the atomicity should guaranty that A should roll back to the original amount. Now, before this roll back happens, suppose that is another transaction that reads the value of A. So, that is the dirty read.

Why it is the dirty read? Because, the value that is read is the new value which is dirty, because the transaction that updated the value of A has not yet committed. So, there is the concept of committing, committing meaning essentially completing successfully that

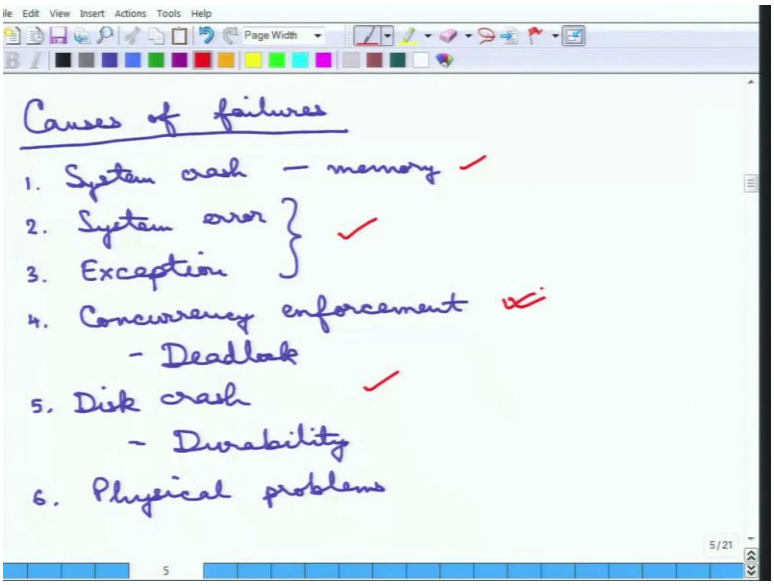
has not yet done correctly that is not yet completed correctly, but another transaction has come and read that value. So, the update in the first cases the temporary, because the transaction has not yet said that this update is correct and it has not gone through completely.

So, it is the update has been temporary, but another transaction read this. So, that is why it is called a temporary update at the dirty read problem and the third problem is called an incorrect summary. So, that incorrect summary problem is that suppose one of the transaction is just trying to find out the aggregate or some aggregate operation on all the bank accounts, etcetera and what may happen is that it reads the A and then A is changed.

So, the statistics the aggregate that the first transaction completely wrong, because A has change. So, by the time the first transaction finishes the aggregate A has changed, but the value that it shows across the let us say the some of the accounts of A and B etcetera or the average etcetera will be wrong, because it is rate something wrongly. So, that is an incorrect summary, so the summarization that it provides the essentially the aggregation etcetera is wrong.

And this happens, because of the temporary update of the dirty read problem, but this is another very common problem that the database practitioner's faced earlier when they were looking at transactions. So, that is all that is fine, but makes the question comes is why your transactions fails.

(Refer Slide Time: 16:06)



So, what are the causes of failures, so why would transactions fail. Now of course, the first very simple way to understand is that there is a crash. So, there is the system crash that is it. So, either the power of the database machine has gone or there is some errors, some exception happen, etcetera. So, the memory is loss all those things happen, so there is the system crash that is fine, then there is a system error.

So, how can error happen is that suppose A has 1000 rupees and it is trying to transfer 2000 rupees it cannot right that is an error that is a logical error though does not think wrong in that the sense of read write operations from the transaction. So, but that is an error and the transaction has cannot succeed in any way. So, because there is no way that 2000 rupees can be transferred out of 1000 rupees and they can be other program error such as division by 0 etcetera, but that is the system error.

And again I mean this division by 0 etcetera can also be called which is called the exception. So, that is may be many other x kinds of exception, etcetera, so insufficient account balances. So, these two are mostly the same system error or exception then the fourth one is that there is a concurrency. So, we will define concurrency and all those things carefully, but so the error that happens, because of trying to enforce the concurrency and this is happening, because of something called a deadlock.

We will see this definition of deadlock etcetera definition concurrency in much more detail later. But, the idea is the following is that transaction one request to read some item B which has been updated by transaction 2 and transaction 2 reads to needs to read transaction A some data item A that is been retain by transaction. So, very simply transaction 1 depends on transaction 2 and transaction 2 depends on transaction 1. So, there is a deadlock meaning, nobody can proceed either the other completing, but nobody can completely either.

So, the system essentially stops and none of the transactions can finish and none of the transactions can therefore, finish successfully. So, that is a deadlock. And we will define in much more detail later, but that is one. Then of course, system crash then I mean the system crash can be memory failure, this is generally a memory failure and of course, I can we are disk crash the entire database as crashes that is a distance at these must take care of the durability.

So, if the database crashes the durability can be happens, so there are may be problem with the transaction ((Refer Time: 18:36)) thing and then there are maybe I mean other physical problems, I mean fire or theft, etcetera whatever physic this is essentially why the transaction may be failed. But, the important things is the disk crash or the system crash and these two and then this is we will study much more later.