purpose and conclusion

# SQL – Transactions

# Cau hoi

A

B

C

D

Table CH

Id, A, B, C, D, DaD, IdBCH

**What Is a Transaction?**

A transaction is a set of operations performed so all operations are guaranteed to succeed or fail as one unit.  
  
**Transaction is all or none**  
  
A common example of a transaction is the process of transferring money from a checking account to a savings account.   
  
This involves two operations:

1. Deducting money from the checking account and   
     
   **Note**: in the USA a checking account is like a current account in India
2. Adding it to the savings account.

Both must succeed together and the changes must be committed to the accounts, or both must fail together and rolled back so that the accounts are maintained in a consistent state. Under no circumstances should money be deducted from the checking account but not added to the savings account (or vice versa), you would at least not want this to happen with the transactions occurring with your bank accounts.   
  
By using a transaction concept, both the operations, namely debit and credit, can be guaranteed to succeed or fail together. So both accounts remain in a consistent state all the time.

**When to Use Transactions**  
You should use transactions when several operations must succeed or fail as a unit. The following are some frequent scenarios where use of transactions is recommended:

* In batch processing, where multiple rows must be inserted, updated, or deleted as a single unit
* Whenever a change to one table requires that other tables be kept consistent
* When modifying data in two or more databases concurrently
* In distributed transactions, where data is manipulated in databases on various servers

When you use transactions, you put locks on data that is pending for permanent change to the database. No other operations can take place on locked data until the acquired lock is released. You could lock anything from a single row up to the entire database. This is called concurrency, which means how the database handles multiple updates at one time.  
  
In the bank example above, locks will ensure that two separate transactions don't access the same accounts at the same time. If they do then either deposits or withdrawals could be lost.  
  
**Note:** it's important to keep transactions pending for the shortest period of time. A lock stops others from accessing the locked database resource. Too many locks, or locks on frequently accessed resources, can seriously degrade performance.  
  
**Understanding ACID Properties**A transaction is characterized by four properties, often referred to as the ACID properties: atomicity, consistency, isolation, and durability.  
  
**Note:** The term ACID was coined by Andreas Reuter in 1983.  
  
**Atomicity**: A transaction is atomic if it's regarded as a single action rather than a collection of separate operations. So, only when all the separate operations succeed does a transaction succeed and is committed to the database. On the other hand, if a single operation fails during the transaction then everything is considered to have failed and must be undone (rolled back) if it has already taken place. In the case of the order-entry system of the Northwind database, when you enter an order into the Orders and Order Details tables, data will be saved together in both tables, or it won't be saved at all.  
  
**Consistency**: The transaction should leave the database in a consistent state, whether or not it completed successfully. The data modified by the transaction must comply with all the constraints placed on the columns in order to maintain data integrity. In the case of Northwind, you can't have rows in the Order Details table without a corresponding row in the Orders table, since this would leave the data in an inconsistent state.  
  
**Isolation**: Every transaction has a well-defined boundary; that is, it is isolated from another transaction. One transaction shouldn't affect other transactions running at the same time. Data modifications made by one transaction must be isolated from the data modifications made by all other transactions. A transaction sees data in the state it was in before another concurrent transaction modified it, or it sees the data after the second transaction has completed, but it doesn't see an intermediate state.  
  
**Durability**: Data modifications that occur within a successful transaction are kept permanently within the system regardless of what else occurs. Transaction logs are maintained so that should a failure occur the database can be restored to its original state before the failure. As each transaction is completed, a row is entered in the database transaction log. If you have a major system failure that requires the database to be restored from a backup then you could then use this transaction log to insert (roll forward) any successful transactions that have taken place.  
  
Every database software that offers support for transactions enforces these four ACID properties automatically.

## **Syntax**

BEGIN TRANSACTION  
Marks the starting point of an explicit transaction.

COMMIT [ WORK ]  
Marks the end of an explicit or autocommit transaction. This statement causes the changes in the transaction to be permanently committed to the database. The statement COMMIT is identical to COMMIT WORK, COMMIT TRAN, and COMMIT TRANSACTION.

ROLLBACK [ WORK ]  
Rolls back a transaction to the beginning of the transaction. No changes for the transaction are committed to the database. The statement ROLLBACK is identical to ROLLBACK WORK, ROLLBACK TRAN, and ROLLBACK TRANSACTION.

**Transaction State**

In the absence of failures, all transactions complete successfully. However, a transaction may not always complete its execution successfully. Such a transaction is termed aborted.

A transaction that completes its execution successfully is said to be committed. Figure 1-1 shows that if a transaction has been partially committed then it will be committed but only if it has not failed and if the transaction has failed, it will be aborted.

