Transaction is a group of SQL statement that represent a single unit of work, and it has 4 properties (AKA ACID):

1. Atomicity

All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are. For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.

1. Consistency

Data is in a consistent state when a transaction starts and when it ends. When the data modified by committed transactions need to be valid and pass all the constraints and rules. Otherwise, the transaction will roll back the data to the initial state.

Isolation

1. The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized.
2. Durability

After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.

Lock is the way that SQL manage transaction concurrency; while one transaction is accessing a data item, no other transaction can modify that data item. So, the most common method used to implement requirement is to allow a transaction to access a data item only if it is currently holding a lock on that item. There are two types of lock: shared lock and exclusive lock. Shared lock is also called read lock, and it used for reading data item only. It supports the read integrity. They ensure that a record is not in process of being updated during a read-only request. Exclusive Lock is also called write lock. It prevents any other locker from obtaining any sort of a lock on the object. Lastly, there is deadlock, which means two transaction lock each other’s resources.

Isolation affects database concurrency and determines the consistency level of the data the transaction interacts with. There are four isolation levels based on lock system only: read uncommitted, read committed, repeatable read, and serializable.

Read Uncommitted is the lowest isolation level. In this level, one transaction may read not yet committed changes made by other transaction, thereby allowing dirty reads. In this level, transactions are not isolated from each other.

Read Committed isolation level guarantees that any data read is committed at the moment it is read. Thus it does not allows dirty read. The transaction holds a read or write lock on the current row, and thus prevent other transactions from reading, updating or deleting it.

Repeatable Read is the most restrictive isolation level. The transaction holds read locks on all rows it references and writes locks on all rows it inserts, updates, or deletes. Since other transaction cannot read, update or delete these rows, consequently it avoids non-repeatable read.

Serializable is the Highest isolation level. A serializable execution is guaranteed to be serializable. Serializable execution is defined to be an execution of operations in which concurrently executing transactions appears to be serially executing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Isolation level/concurrency problem | Lost update | Dirty Reads | Non-Repeatable reads | Phantom reads |
| Read Uncommitted |  |  |  |  |
| Read Committed |  | Solve |  |  |
| Repeatable read | Solve | Solve | Solve |  |
| Serializable | Solve | Solve | Solve | Solve |