**28.Write a short essay talking about your understanding of transactions, locks and isolation levels.**

If I want to just say one sentence about transaction, it is a single unit of work which would be either committed or not committed (rollback). In the other word, it is a set of command, everything would execute, or nothing executed that means it has ONLY two possible outcomes: Committed or Rollback.

The purpose of transaction is avoiding concurrency in the database. Concurrency means when multiple users are working on the same database at the same time.

There are 2 types of transaction: 1- Local transaction 2- distributed transaction.

1. It is working with the same local database system
2. It can have transaction over distributed clusters.

Each transaction should exhibit four properties: Atomicity, Consistency, Isolation, Durability (ACID).

Atomicity: a transaction is an atomic unit of work; we cannot split transaction.

Consistency: transaction database must follow all the rules to maintain data integrity. All data in a consistent state.

Isolation: modification made by different users at the same time on the same database should be isolated(lock).

Durability: after executing a transaction, the transaction’s result is permanently in place.

Also, we have explicit transaction and implicit transaction.

Explicit transaction: user can define the start and end of the transaction.

Implicit transaction: It’s an auto commit transaction. The instance of the SQL server database engine automatically starts a new transaction.

@@trancount means a transaction can be nested and @@trancount will define the level of transaction a user currently at.

Unnamed transaction and using nested transaction: each time commit one layer of transaction but rollback the entire transaction.

Named transaction and using nested transaction: each time commit one layer of transaction but can rollback to a named transaction or save point. Save point and named transaction are very useful in error handling.

We have 2 types of concurrency control:

* Optimistic Concurrency Control:
* It doesn’t use any lock.
* It is faster in retrieving data because there is no lock to wait for it. However, when transaction updates some resources it doesn’t know the status of other transaction about what they are doing with the same resource.
* Chance of rolling back is higher.
* Row versioning to view data as it existed at the start of a transaction.
* Pessimistic Concurrency Control:
* It uses locks, if a resource is locked the other transaction must wait for the other resource or transaction to release lock.
* Transaction can wait for other transaction to release the lock.
* Smaller chance of rolling back but higher chance to wait because data is already changed.

We can categorize locks by the size of the data they lock: row lock, page lock (minimum physical unit where data stored in hard drive = 8 bytes), range lock, table lock and database lock.

Categorizing locks by the usage: Shared lock(read), Update lock and Exclusive lock.

* Shared Lock: for read operations that do not change or update data, such as SELECT statement.

Once reading finished shared lock would disappear.

* Update Lock: For resources that can be updated. Prevents from deadlock that occurs when multiple sessions are reading.
* Exclusive lock: for data modification operations, ensures multiple updates can’t be made to the same resource at the same time.

Isolation/Concurrency Level:

1. Read Uncommitted: for reading data shared lock is not being applied and ignores all the locks. Reading is not protected and not waiting for other locks. This is fastest way of retrieving data, but the quality of data is not being considered and there is no deadlock.
2. Read Committed (system default): Shared lock is applied when we are trying to read the data once reading finished shared lock will disappear. In order to update/delete something transaction is going to apply update lock before any changes. After that update lock will transfer to exclusive lock which means this data is exclusive to this particular transaction and no other transaction can see anything about this data. It will be there until rollback the transaction or commit the transaction.
3. Serializable: is based on repeatable read with additional protection. Data locked by shared lock, and nobody can’t insert new row to table or set until transaction ends. More secure but more waiting time.

Deadlock = when multiple transactions are waiting for each other to release their corresponding locks. Once deadlock happens, SQL Server can detect it automatically and will kill the simple but simple one will give error message.