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

Transactions are logical work units that perform activities in a database. Transactions may consist of reading, writing, deleting, updating, or a combination of these. Transactions follow the ACID rule representing atomicity, consistency, isolation, and durability. Atomicity means that the whole transaction will run in a whole; if a part fails, the rest of that transaction will not modify the database; instead, it rolls back to the beginning of the transaction. Consistency means all data must be a consistent data type after the transaction according to the defined rules in the database. Isolation means all transactions are isolated from other transactions; when one transaction is running, other transactions will not be affected. Duration means the committed transactions are changed permanently.

To control a transaction and make sure every transaction follows the ACID test, sql server uses locking and row versioning mechanisms to ensure the integrity of transactions and maintain the consistency of databases when multiple users are accessing data simultaneously or short for concurrency effect.

Two kinds of concurrency control are Pessimistic concurrency control and Optimistic concurrency control.

Pessimistic concurrency control uses locks to prevent users from modifying data that affects other users and use this when there is high contention for data, where using locks saves more resources than rolling back when there’s a concurrency conflict—usually taking more time on waiting and less failure than optimistic concurrency control.

There are three basic locks on row level: exclusive lock, shared lock, and update lock. Exclusive locks will ensure that a page or row will be reserved exclusively for the transaction that imposed the exclusive lock, as long as the transaction hold the lock, only one exclusive lock can be imposed on the transaction. A shared lock will reserve a page or a row to be available only for reading, which means other transactions will be prevented from modifying the locked record as long as the lock is active, it allows write operations, but no ddl changes will be allowed. Update lock is similar to exclusive lock but is designed to be more flexible, which means it can be imposed on a record that already has a shared lock and once the transaction is ready to change the data, the update lock will be transformed into an exclusive lock.

Optimistic concurrency control uses a row versioning mechanism to control concurrency control which means when a suer updates data, the system checks to see if another user changed the data after it was read. If so, the transaction will receive an error and roll back to the beginning. This strategy is mainly used in environments where there is low contention for data.

Isolation levels are sued to define the degree to which one transaction must be isolated from resource or data modifications made by other concurrent transactions. There are five isolation levels from lower to higher: Read uncommitted, read committed, repeatable read, serializable, snapshot. The system default one is read committed. Read uncommitted is the lowest isolation level that does not issue shared locks to prevent other transactions from modifying data read by the current transaction. With Read Committed, transactions issue exclusive locks at the time of data modification, thus not allowing other transactions to read the modified data that is not yet committed. The Read Committed isolation level prevents the Dirty Read issue. However, data can be changed by other transactions between individual statements within the current transaction, resulting in a Non-repeatable Read or a Phantom Row. In Repeatable Read, statements cannot read data that has been modified but not yet committed by other transactions. No other transaction can modify data that the current transaction has read until the current transaction completes. In the serializable isolation level, statements cannot read data that has been modified but not yet committed by other transactions. No other transactions can modify data that the current transaction has read until the current transaction completes. In snapshot isolation, data read by any statement in a transaction will be the transactional consistent version of the data that existed at the start of the transaction. Data modifications made by other transactions after the start of the current transaction are not visible to statements executing in the current transaction. SNAPSHOT transactions do not request locks when reading data