Background

A transaction is a unit of work, including query/update data and data definition. When users try to access the same data at same time, SQL Server tries to use locks to isolate inconsistent data and use the isolation level to query data to control consistency.

1 Transaction

A transaction is a single unit of work. If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database. If a transaction encounters errors and must be canceled or rolled back, then all of the data modifications are erased.

Transaction has four characteristics, Atomicity, Consistency, Isolation, Durability

* Atomicity: The atomicity acid property in SQL. It means either all the operations (insert, update, delete) inside a transaction take place or none. Or you can say, all the statements (insert, update, delete) inside a transaction are either completed or rolled back.
* Consistency: This SQL ACID property ensures database consistency. It means, whatever happens in the middle of the transaction, this acid property will never leave your database in a half-completed state. If the transaction completed successfully, then it will apply all the changes to the database. If there is an error in a transaction, then all the changes that already made will be rolled back automatically. It means the database will restore to its state that it had before the transaction started. If there is a system failure in the middle of the transaction, then also, all the changes made already will automatically rollback.
* Isolation: Every transaction is individual, and One transaction can’t access the result of other transactions until the transaction completed. Or, you can’t perform the same operation using multiple transactions at the same time. We will explain this SQL acid property in a separate article.
* Durability: Once the transaction completed, then the changes it has made to the database will be permanent. Even if there is a system failure, or any abnormal changes also, this SQL acid property will safeguard the committed data.

2 Lock

2.1 Lock type

2.1.1 Shared Lock: Used for operations that do not change or update data (read-only operations), such as SELECT statements. Shared lock can be imposed by several transactions at the same time over the same page or row and in that way several transactions can share the ability for data reading since the reading process itself will not affect anyhow the actual page or row data. A shared lock will allow write operations, but no DDL changes will be allowed.

2.1.2 Exclusive lock: This lock type, when imposed, will ensure that a page or row will be reserved exclusively for the transaction that imposed the exclusive lock, as long as the transaction holds the lock. The exclusive lock will be imposed by the transaction when it wants to modify the page or row data, which is in the case of DML statements DELETE, INSERT and UPDATE. An exclusive lock can be imposed to a page or row only if there is no other shared or exclusive lock imposed already on the target. only one exclusive lock can be imposed to a page or row, and once imposed no other lock can be imposed on locked resources.

2.1.3 UPDLOCK: Used in updatable resources. Prevent common forms of deadlock when multiple sessions are reading, locking, and possibly subsequent resource updates. Allows to read data (without blocking other transactions) and update the data later, while ensuring that the data has not been changed since the last time the data was read. When use UPDLOCK to read the record, also can add an update lock to the fetched record, so that the locked record cannot be changed in other threads and can only be changed after the end of the transaction of this thread.

There also some specific lock like Intent lock, Schema lock, Bulk update lock.

3 Isolation Levels

SQL Server isolation levels are used to define the degree to which one transaction must be isolated from resource or data modifications made by other concurrent transactions. The different Isolation Levels are:

1: Read Uncommitted 2: Read Committed 3: Repeatable Read 4: Serializable 5: Snapshot

3.1Read Committed

Read Committed is the default isolation level.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.

The behavior of read commit depends on the setting of the read\_committed\_snapshot database option. If read\_committed\_snapshot is set to OFF, shared locks prevent other transactions from modifying rows while the current transaction is running a read operation. The shared locks also block the statement from reading rows modified by other transactions until the other transaction is completed. If read\_committed\_snapshot is set to ON (the default on SQL Azure Database), row versioning is used to present each statement with a transactional consistent snapshot of the data as it existed at the start of the statement. No locks are issued on the data.

3.2 Read Uncommitted

Transactions running at this level do not issue shared locks to prevent other transactions from modifying data read by the current transaction. Also, transactions are not blocked by exclusive locks at the time of data modification, thus allowing other transactions to read the modified data which is not yet committed.

3.3 Repeatable Read

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 has been read by the current transaction until the current transaction completes.Shared locks are placed on all data read by each statement in the transaction and are held until the transaction completes. This prevents other transactions from modifying any rows that have been read by the current transaction. This isolation level prevents the Non-Repeatable Read issue. However, other transactions can insert new rows that match the search conditions of statements issued by the current transaction. If the current transaction then retries the statement it will retrieve the new rows, which results in phantom reads.

3.4 Serializable

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 has been read by the current transaction until the current transaction completes. Other transactions cannot insert new rows with key values that would fall in the range of keys read by any statements in the current transaction until the current transaction completes.

3.5 Snapshot Isolation

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. SNAPSHOT transactions reading data do not block other transactions from writing data. Transactions writing data do not block SNAPSHOT transactions from reading data.