What is a lock(ing), block(ing) or a deadlock?

In the next several videos, we will discuss and demonstrate what these objects are and how understanding them will help us in performance tuning the SQL Server. We begin with ‘what is a lock?’

What is locking in SQL SERVER?

SQL Server in an environment that has multiple users needing to access the same data simultaneously.
SQL Server must somehow control how these reads and writes to the database are carried out.

In a multi-user environment, locking is the process of **ensuring the integrity of the data** is maintained in the database

It does this by forcing every SQL Server transaction to pass the ACID test

The ACID test consists of 4 requirements that every transaction must pass
They are:

- **Atomicity**. In a transaction involving two or more discrete pieces of information, either all of the pieces are committed or none are
- **Consistency**. A transaction either creates a new and valid state of data, or, if any failure occurs, returns all data to its state before the transaction was started
- **Isolation**. A transaction in process and not yet committed must remain isolated from any other transaction
- **Durability**. Committed data is saved by the system such that, even in the event of a failure and system restart, the data is available in its correct state

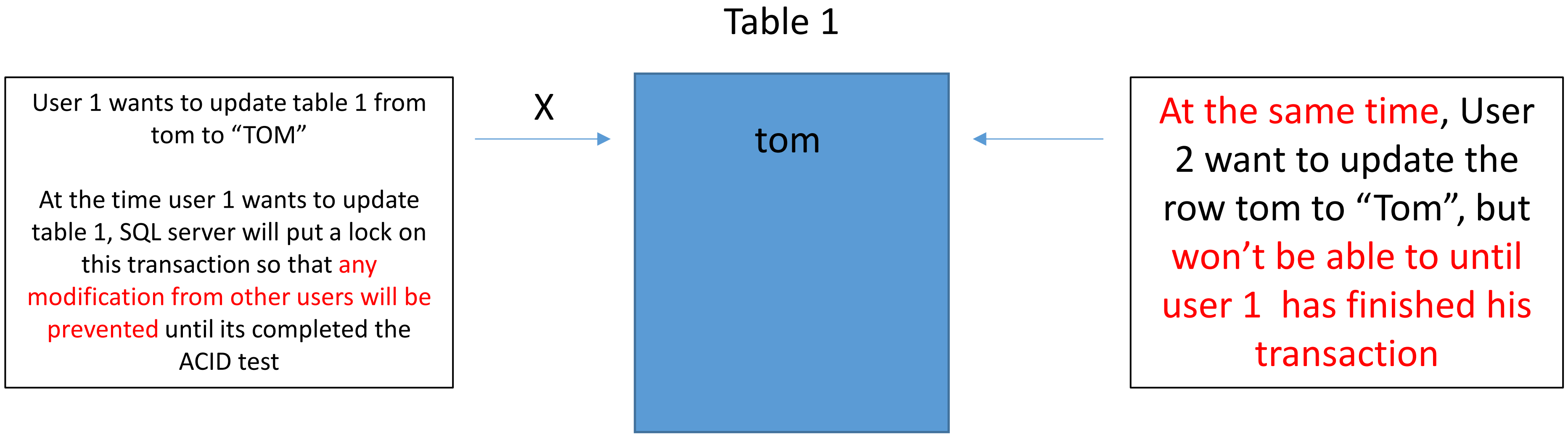
Locking in action

Example of locking

Locking can be applied to rows, tables, indexes, databases

During the transaction the row is held by the lock

Locks occur during reads, inserts, updates, deletes



User 1 wants to update table 1 from tom to "TOM"

At the time user 1 wants to update table 1, SQL server will put a lock on this transaction so that any modification from other users will be prevented until its completed the ACID test

At the same time, User 2 want to update the row tom to "Tom", but won't be able to until user 1 has finished his transaction

Types of Lock Modes

To better understand of how locking works, you need to be aware of different transaction levels and the different lock modes

- **Exclusive (X)**

  Exclusive locks are used to lock data being modified by one transaction thus preventing modifications by other concurrent transactions

- **Shared (S)**

  Shared locks are held on data being read. While a shared lock is being held other transactions can read but can't modify locked data

- **Update (U)**

  Update locks are a mix of shared and exclusive locks. The update lock itself can't modify the underlying data. It has to be converted to an exclusive lock before the modification takes place

- **Intent (I)**

  Intent locks are a means in which a transaction notifies other transaction that it is intending to lock the data

- **Schema (Sch)**

  Schema stability lock (Sch-S): Used while generating execution plans. These locks don't block access to the object data

  Schema modification lock (Sch-M): Used while executing a DDL statement. Blocks access to the object data since its structure is being changed

- **Bulk update (BU)**

  Bulk Update locks are used by bulk operations

Script to find lock info

```
SELECT resource_type, request_mode, resource_description
FROM sys.dm_tran_locks
WHERE resource_type <> ‘DATABASE’
```