

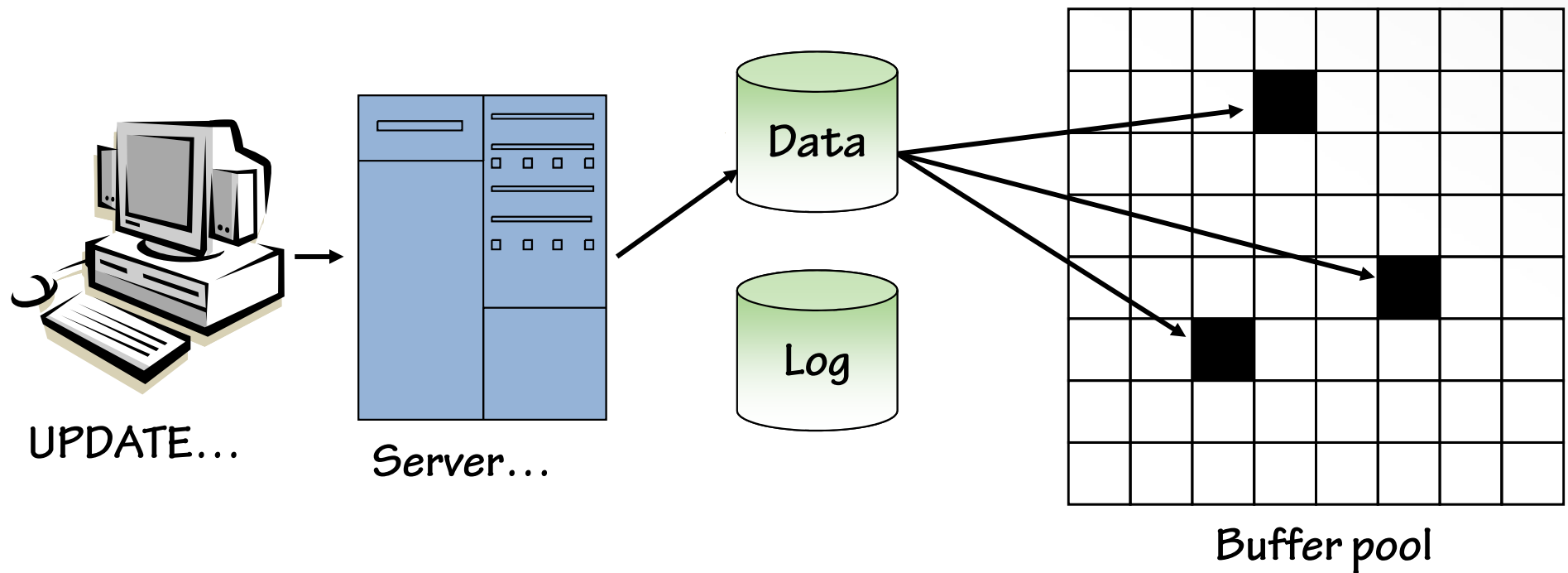
Anatomy of a Data Modification

- **User/application sends an UPDATE query**
 - The update is highly selective (only 5 rows)
- **Indexes exist to aid in finding these rows efficiently**
- **The update is a SINGLE statement batch therefore this is an IMPLICIT transaction**
 - Transactions can be 'explicit' or 'implicit'
 - Explicit transactions are controlled by the user
 - Started with BEGIN TRAN
 - Ended with COMMIT TRAN or ROLLBACK TRAN
 - Implicit transactions are created internally by SQL Server and committed automatically when the operations complete
 - And obviously rolled-back if something goes wrong

Anatomy of a Data Modification

- **Server receives the request and locates the data in cache OR reads the data from disk into cache**
 - Since this is highly selective only the necessary pages are read into cache (maybe a few extra but that's not important here)
 - Let's use an example where the 5 rows being modified are located on 3 different data pages

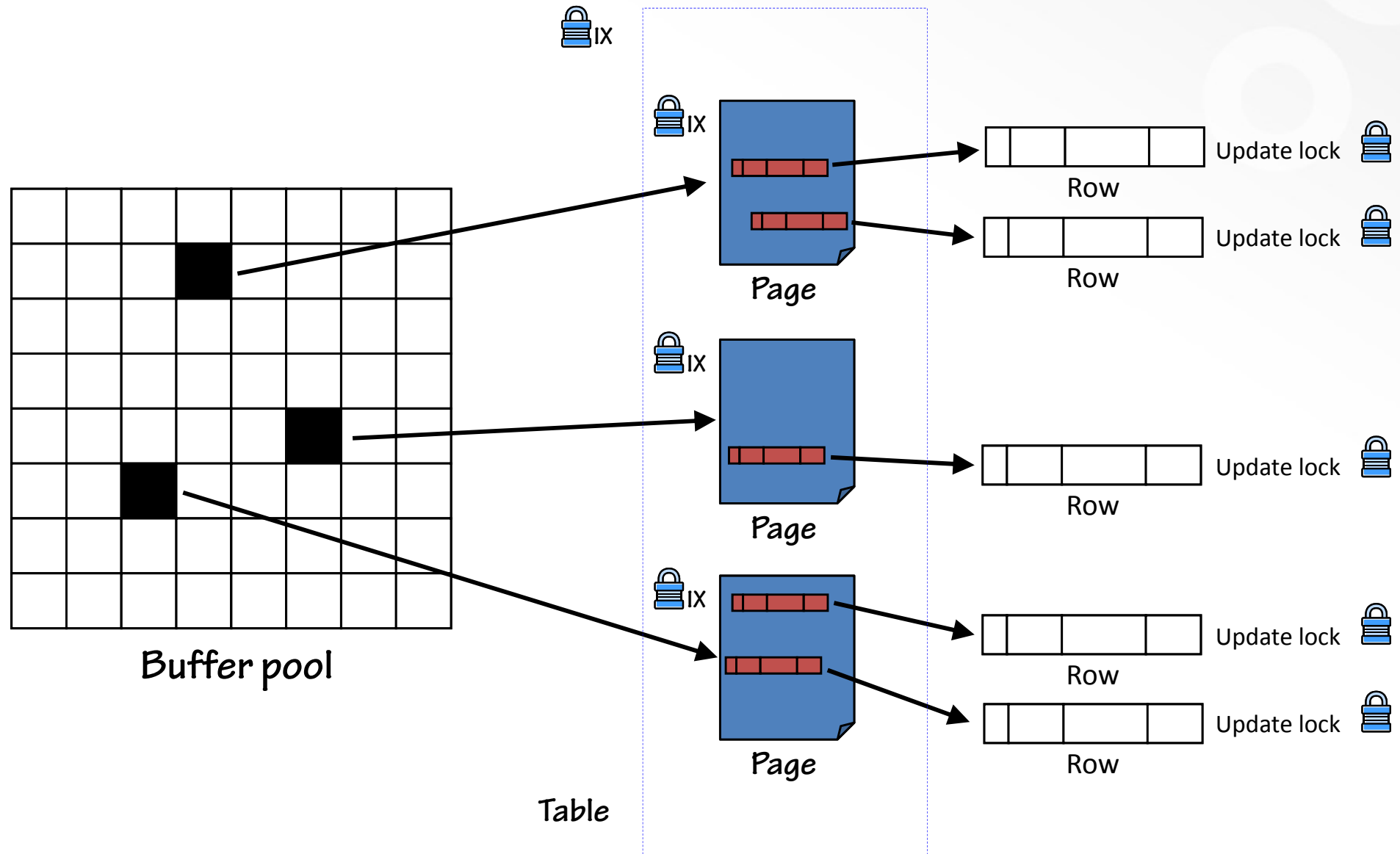
What it looks like: Data Reading From Disk



Anatomy of a Data Modification

- **SQL Server proceeds to lock the necessary data**
 - Locks are necessary to give a consistent point FOR ALL rows from which to start
 - If any other transaction(s) have ANY of these rows locked we will wait until ALL locks have been acquired before we can proceed
 - Locks are initially taken to stabilize the rows and then upgraded to exclusive locks
 - In the case of this update (because it's highly selective and because indexes exist to make this possible) SQL Server will use row level locking

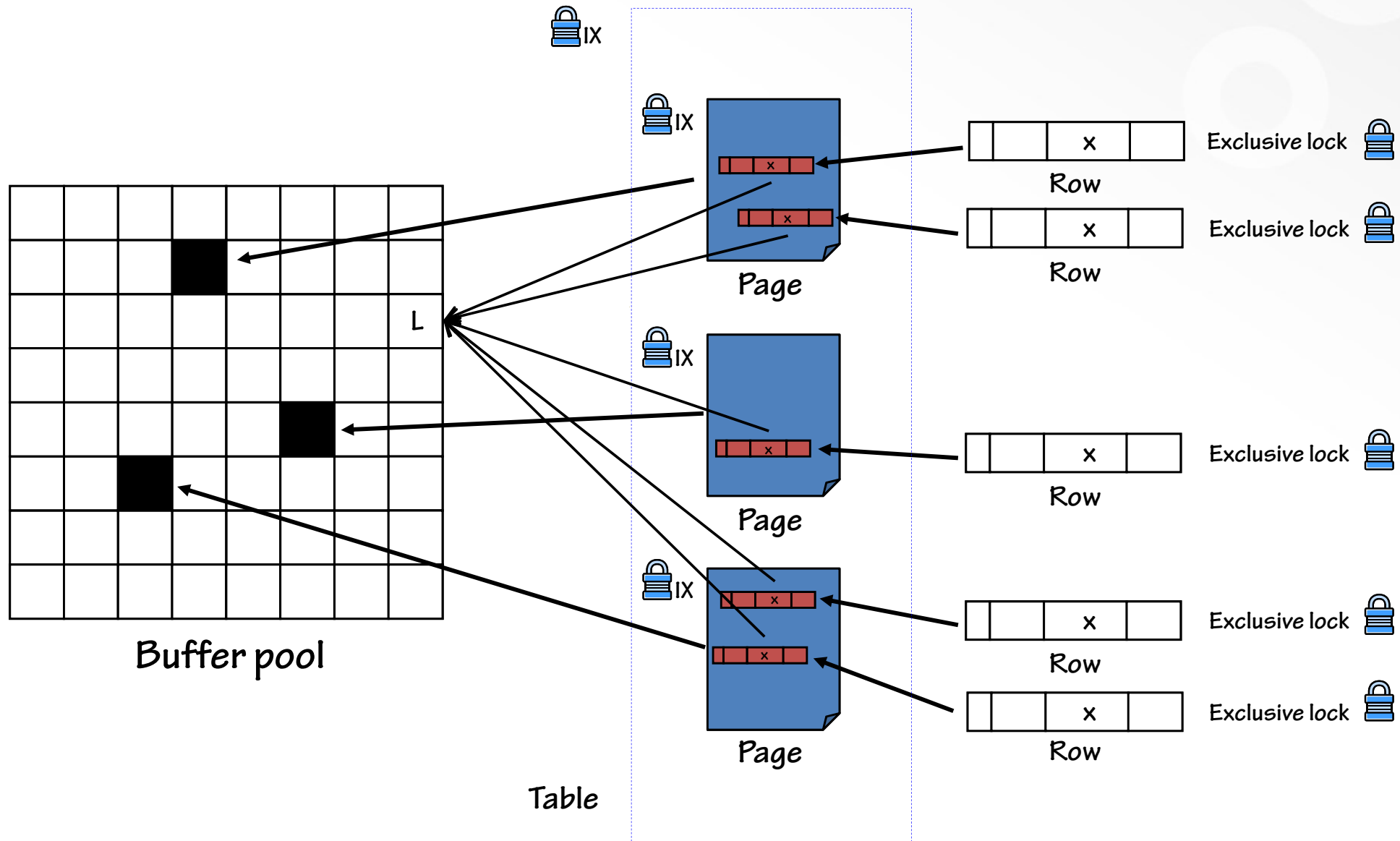
What It Looks Like: Acquiring Locks



Anatomy of a Data Modification

- The rows are locked but there are also 'intent' locks at higher levels to make sure other larger locks (like other potentially conflicting page or table level locks) are not attempted and then fail
 - This transaction holds the following locks:
 - 5 update row-level locks
 - 3 intent-exclusive page-level locks
 - 1 intent-exclusive table-level lock
 - The connection also holds a shared database-level lock
- And if indexes are accessed/used then there might be additional locks required – to read the data – they are not significant here

What It Looks Like: Modifications



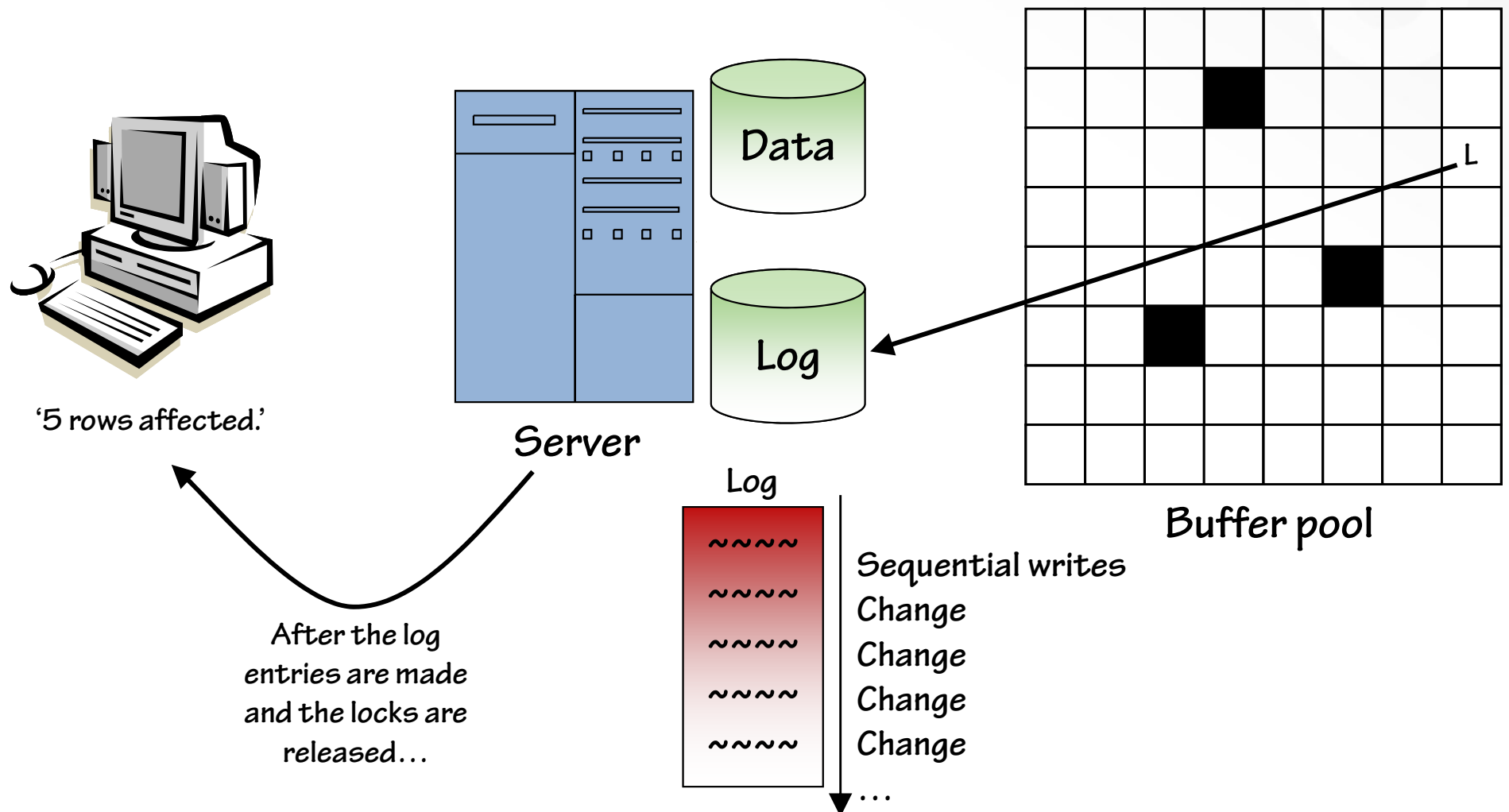
Anatomy of a Data Modification

- **SQL Server can now begin to make the modifications**
- **For EVERY row the process will include:**
 - Change to a stricter lock (eXclusive lock)
 - An update lock helps to allow better concurrency by being compatible with other shared locks (readers). Readers can read the pre-modified data as it is transactionally consistent
 - The eXclusive lock is required to make the change because once modified no other reads should be able to see this un-committed change
 - Make the modification (in cache)
 - Log the modification to the transaction log pages (also in cache)

Anatomy of a Data Modification

- **Finally, the transaction is complete**
- **This is the MOST critical part (the key to Durability)**
 - All rows have been modified
 - There are no other statements in this batch (because it's an implicit transaction)
- **Steps are:**
 - Write all log records for the transaction to the transaction log ON DISK (forced write-through to disk) = durable
 - This forces all of the transaction log up to the point of the COMMIT TRAN log record to be written to disk, regardless of which transaction it is for
 - Release all locks held by the transaction
 - Acknowledge the commit to the user/application:
 - (5 Rows Affected)

What It Looks Like: Committing



So Now What?

- The transaction log ON DISK contains a record of the changes made to the database by the transaction
- The data pages in the buffer pool reflect the changes made to the database by the transaction
- When do the up-to-date data pages get written from buffer pool into the data files on disk?

Checkpoint

Anatomy of a Data Modification: Where Are We At?

- **Optimization**
 - Data is very random
 - Log is sequential
- **Locks**
 - Granularity
 - Duration
 - Escalation
- **Transactions**
 - Can make a mess of things if you don't know what you're doing...
- **NOTE: Paul will be talking more about logging, recovery, log records and checkpoints in Module 5**