Lecture #09: Index Concurrency Control

15-445/645 Database Systems (Fall 2018)
`https://15445.courses.cs.cmu.edu/fall2018/`
Carnegie Mellon University
Prof. Andy Pavlo

1 Index Concurrency Control

A *concurrency control* protocol is the method that the DBMS uses to ensure "correct" results for concurrent operations on a shared object.

A protocol's correctness criteria can vary:

- **Logical Correctness:** Can I see the data that I am supposed to see? This means that the thread is able to read values that it should be allowed to read.
- **Physical Correctness:** Is the internal representation of the object sound? This means that there are not pointers in our data structure that will cause a thread to read invalid memory locations.

The logical contents of the index is the only thing we care about in this lecture. They are not quite like other database elements so we can treat them differently.

2 Locks vs. Latches

Locks:

- Protects the indexs logical contents from other transactions.
- Held for (mostly) the entire duration of the transaction.
- The DBMS needs to be able to rollback changes.

Latches:

- Protects the critical sections of the indexs internal data structure from other threads.
- Held for operation duration.
- The DBMS does not need to be able to rollback changes.
- Two Modes:
  - **`READ`:** Multiple threads are allowed to read the same item at the same time. A thread can acquire the read latch if another thread has it in read mode.
  - **`WRITE`:** Only one thread is allowed to access the item. A thread cannot acquire a write latch if another thread holds the latch in any mode.

3 Latch Crabbing / Coupling

Lock crabbing / coupling is a protocol to allow multiple threads to access/modify B+Tree at the same time:

1. Get latch for parent.
2. Get latch for child.
3. Release latch for parent if it is deemed safe. A **safe node** is one that will not split or merge when updated (not full on insertion or more than half full on deletion).

Basic Latch Crabbing Protocol:

- **Search:** Start at root and go down, repeatedly acquire latch on child and then unlatch parent.
- **Insert/Delete:** Start at root and go down, obtaining `X` latches as needed. Once child is latched, check if it is safe. If the child is safe, release latches on all its ancestors.

Improved Lock Crabbing Protocol: The problem with the basic latch crabbing algorithm is that transactions always acquire an exclusive latch on the root for every insert/delete operation. This limits parallelism. Instead, we can assume that having to resize (i.e., split/merge nodes) is rare, and thus transactions can acquire shared latches down to the leaf nodes. Each transaction will assume that the path to the target leaf node is safe, and use `READ` latches and crabbing to reach it, and verify. If any node in the path is not safe, then do previous algorithm (i.e., acquire `WRITE` latches).

- **Search:** Same algorithm as before.
- **Insert/Delete:** Set `READ` latches as if for search, go to leaf, and set `WRITE` latch on leaf. If leaf is not safe, release all previous latches, and restart transaction using previous Insert/Delete protocol.

4 Leaf Node Scans

The threads in these protocols acquire latches in a “top-down” manner. This means that a thread can only acquire a latch from a node that is below its current node. If the desired latch is unavailable, the thread must wait until it becomes available. Given this, there can never be deadlocks.

Leaf node scans are susceptible to deadlocks because now we have threads trying to acquire locks in two different directions at the same time (i.e., left-to-right and right-to-left). Index latches do not support deadlock detection or avoidance.

Thus, the only way we can deal with this problem is through coding discipline. The leaf node sibling latch acquisition protocol must support a “no-wait” mode. That is, B+tree code must cope with failed latch acquisitions. This means that if a thread tries to acquire a latch on a leaf node but that latch is unavailable, then it will immediately abort its operation (releasing any latches that it holds) and then restart the operation.