ECE356: Database Systems Winter 2018

Lecture 25 — Snapshot Isolation

Jeff Zarnett

Snapshot Isolation for Concurrency Control

We already got a look at the idea of transaction isolation but we are now going to examine more carefully how it works behind the scenes. In general, the idea is that every transaction has its own "world" it can operate in and then we need to merge the result when the transaction is ready to commit. And that needs to be done atomically.

We can use validation to decide whether a transaction is allowed to commit. This applies really only for transactions that do at least one write. Reads don't interfere with one another, really, and if the two writes are on disjoint items, there are no problems there either. It only gets interesting if there are two transactions that have items in common that run at the same time. Remember that here, when we talk about concurrency, we mean two transactions that are "active" at the same time.

Usually, we do not like to just allow both transactions to go forward because the first write is then overwritten by the second; a "lost update" [**?**]. This is sometimes acceptable as a choice, even if some textbooks say that this is really undesirable and horrible.

The first snapshot isolation strategy is called *first committer wins*: whichever transaction T is ready to commit first has to pass a simple test. If any transaction concurrent with T has already written an update to any data item that T wants to write, T is rolled back; otherwise T commits and its updates are written to the database. If the