Timestamp Based protocols

Isaac Chang

Timestamps

1. Every transaction is associated with a unique timestamp that is assigned right before a transaction is executed
2. If another transaction enters the system, the timestamp it is assigned must be larger than the timestamp of any pending transactions
3. Two ways of assigning timestamps
    a. Use the value of the system’s clock
    b. Keep a counter that is incremented whenever a new transaction enters the system

Timestamp conflicts

1. Transactions can encounter ordering conflicts when the invariant is not held when multiple transactions access or write to the database before or after other transactions are meant to
2. Transaction needs to read data that was recently overwritten
   a. Accessing stale data
3. Transaction needs to write data that was recently read
   a. Writing data that was previously required by another transaction (system should wait for any writes before reading)
4. Transaction needs to write data that was recently written to
   a. Writing obsolete data
5. In any of these situations, the transactions are rolled back

Drawbacks to timestamp conflict resolution

1. No transactions are force to wait for other transactions, but a large transaction could potentially be rolled back over and over again by other transactions
2. A large transaction can be blocked from finishing if it keeps encountering a conflict from with other transactions
3. The protocol as it is can generate transaction schedules that are not recoverable

Preventing infinite rollbacks

1. Writes are postponed and executed together at the end of transactions
    a. No other transaction is able to access the data currently being written to
2. Reads of data that will be written to are postponed until transactions that are writing to the data finishes
3. Transactions that read data are paused until other transactions that are writing to that data are completed
    a. Read write dependency
4. A mix of these 3 extensions to the original protocol helps prevent transactions from rollback jams

Thomas’ write rule

1. Modification to conflict from writing to data that has just been written to
2. Instead of rolling back when the write conflict is detected, ignore the write all together and keep going
3. Increases the amount of allowed transaction schedules
4. Helps lessen transaction overlaps (maintain view serializability)
5. Does not guarantee preventing conflicts with concurrent transactions
    a. Other conflicts may still occur after transaction ignores write

References

1. Database System Concepts by Silberschatz, Korth and Sudarshan, 18.5
2.