4.2. Concurrency protocols

In this lecture we look at...
[Section notes PDF 37Kb]

4.2.01. Introduction

- Concurrency control protocols
- Concurrency techniques
  - Locks, Protocols, Timestamps
  - Multimode locking with conversion
- Guarenteeing serializability
- Associated cost
- Timestamps and ordering

4.2.02. Guarenteeing serializability

- Two phase locking protocol (2PL)
  - Growing/expanding
    - Acquisition of all locks
    - Or upgrading of existing locks
  - Shrinking
    - Release of locks
    - Or downgrading
  - Guarentees serializability
    - equivalency without checking schedules

4.2.03. A typical transaction pair

T_1	T_2
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
unlock(Y);	unlock(X);
write_lock(X);	write_lock(Y);
read_item(X);	read_item(Y);
X=X+Y;	Y=X+Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

- Violates rules of two phase locking
- unlock occurs during locking/expanding phase

4.2.04. 2PL: Guaranteed serializable

T_1	T_2
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
write_lock(X);	write_lock(Y);
unlock(Y);	unlock(X);
read_item(X);	read_item(Y);
X=X+Y;	Y=X+Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

- Less efficient (cost), but serializable

4.2.05. Guarantee cost

- T_2 ends up waiting for read access to X
- Either after T_1 finished
  - T_1 cannot release X even though it has finished using it
  - Incorrect phase (still expanding)
- Or before T_1 has used it
  - T_1 has to claim X during expansion, even if it doesn't use it until later
- Cost: limits the amount of concurrency

4.2.06. Alternatives

- Concurrency control
  - Locks limit concurrency
    - Busy waiting
  - Timestamp ordering (TO)
  - Order transaction execution
    - for a particular equivalent serial schedule
    - of transactions ordered by timestamp value
      - Note: difference to lock serial equivalent
  - No locks, no deadlock

4.2.07. Timestamps

- Unique identifier for transaction (T)
- Assigned in order of submission
  - Time
    - linear time, current date/sys clock - one per cycle
  - Counter
    - counter, finite bitspace, wrap-around issues

  - Timestamp aka. Transaction start time
  - TS(T)

4.2.08. Timestamping

- DBMS associates two TS with each item

- Read_TS(X): gets read timestamp of item X
  - timestamp of most recent successful read on X
  - = TS(T) where T is youngest read transaction

- Write_TS(X): gets write timestamp of item X
  - as for read timestamp

4.2.09. Timestamping

- Transaction T issues read_item(X)
  - TO algorithm compares TS(T) with Write_TS(X)
  - Ensures transaction order execution not violated
- If successful,Write_TS(X) <= TS(T)
  - Read_TS(X) = $\text{MAX}_{\text{TS(T), current Read\_TS(X)}}$
- If fail,Write_TS(X) > TS(T)
  - T aborted, rolled-back and resubmitted with new TS
  - Cascading rollback

4.2.10. Timestamping

- Transaction T issues write_item(X)
  - TO algorithm compares TS(T) with Read_TS(X) and compares TS(T) with Write_TS (X)
- If successful, op_TS(X) <= TS(T)
  - Write_TS(X) = TS(T)
- If fail, op_TS(X) > TS(T)
  - T aborted, cascade etc.
- All operations focus on not violating the execution order defined by the timestamp ordering

4.2.11. Updates

- Insertion
  - 2PL: DBMS secures exclusive write-lock
  - TOA: op_TS(X) set to TS(creating transaction)
- Deletion
  - 2PL: as insert
  - TOA: waits to ensure later transactions don’t access

- Phantom problem
  - Record being inserted matches inclusion conditions
  - of another transaction
    (e.g. selection by dno=5)
  - Locking doesn't guarantee inclusion

    (need index locking)