Lecture 18-19: Concurrency Control

Wednesday, February 22, 2006
and
Friday, February 24, 2006

Announcements

• Homework 4 is posted

• Mon, 2/27: Guest Lecture *Indexes*
  Prof. Magda Balazinska (CSE, UW)

• Wed, 3/1: Guest Lecture *DB Administration*
  Shankar Pal, (Microsoft, SQL Server group)
Outline

• Concurrency control by timestamps 18.8
• Concurrency control by validation 18.9

Timestamps

Every transaction receives a unique timestamp
TS(T)

Could be:

• The system’s clock
• A unique counter, incremented by the scheduler
Timestaps

Main invariant:

The timestamp order defines the serialization order of the transaction

Timestaps

Associate to each element $X$:
- $RT(X) =$ the highest timestamp of any transaction that read $X$
- $WT(X) =$ the highest timestamp of any transaction that wrote $X$
- $C(X) =$ the commit bit: says if the transaction with highest timestamp that wrote $X$ commited

These are associated to each page $X$ in the buffer pool
Main Idea
For any two conflicting actions, ensure that their order is the serialized order:
In each of these cases
• $w_U(X) \ldots r_T(X)$
• $r_U(X) \ldots w_T(X)$
• $w_U(X) \ldots w_T(X)$
Check that $TS(U) < TS(T)$

When $T$ wants to read $X$, $r_T(X)$, how do we know $U$, and $TS(U)$?

Details
Read too late:
• $T$ wants to read $X$, and $TS(T) < WT(X)$

$$START(T) \ldots START(U) \ldots w_U(X) \ldots r_T(X)$$

Need to rollback $T$!
Details

Write too late:
• T wants to write X, and
WT(X) < TS(T) < RT(X)

\[
\text{START(T) \ldots START(U) \ldots r_{U}(X) \ldots w_{T}(X)}
\]

Need to rollback T!

Why do we check WT(X) < TS(T) ????

Details

Write too late, but we can still handle it:
• T wants to write X, and
TS(T) < RT(X) but WT(X) > TS(T)

\[
\text{START(T) \ldots START(V) \ldots w_{V}(X) \ldots w_{T}(X)}
\]

Don’t write X at all!
(but see later…)

More Problems

Read dirty data:
  • T wants to read X, and WT(X) < TS(T)
  • Seems OK, but…

\[
\text{START(U) \ldots START(T) \ldots w}_U(X) \ldots w_T(X) \ldots \text{ABORT(U)}
\]

If C(X)=1, then T needs to wait for it to become 0

More Problems

Write dirty data:
  • T wants to write X, and WT(X) > TS(T)
  • Seems OK not to write at all, but …

\[
\text{START(T) \ldots START(U) \ldots w}_U(X) \ldots w_T(X) \ldots \text{ABORT(U)}
\]

If C(X)=1, then T needs to wait for it to become 0
Timestamp-based Scheduling

When a transaction $T$ requests $r(X)$ or $w(X)$, the scheduler examines $RT(X)$, $WT(X)$, $C(X)$, and decides one of:

- To grant the request, or
- To rollback $T$ (and restart with later timestamp)
- To delay $T$ until $C(X) = 0$

RULES:

- There are 4 long rules in the textbook, on page 974
- You should be able to understand them, or even derive them yourself, based on the previous slides
- Make sure you understand them!

READING ASSIGNMENT: 18.8.4
Multiversion Timestamp

- When transaction T requests r(X) but \( WT(X) > TS(T) \), then T must rollback
- Idea: keep multiple versions of X: \( X_t, X_{t-1}, X_{t-2}, \ldots \) 

\[
TS(X_t) > TS(X_{t-1}) > TS(X_{t-2}) > \ldots
\]

- Let T read an older version, with appropriate timestamp

Details

- When \( w_T(X) \) occurs create a new version, denoted \( X_t \) where \( t = TS(T) \)
- When \( r_t(X) \) occurs, find a version \( X_t \) such that \( t < TS(T) \) and \( t \) is the largest such
- \( WT(X_t) = t \) and it never changes
- \( RD(X_t) \) must also be maintained, to reject certain writes (why?)
- When can we delete \( X_t \): if we have a later version \( X_{t1} \) and all active transactions T have \( TS(T) > t1 \)
Tradeoffs

• Locks:
  – Great when there are many conflicts
  – Poor when there are few conflicts
• Timestamps
  – Poor when there are many conflicts (rollbacks)
  – Great when there are few conflicts
• Compromise
  – READ ONLY transactions → timestamps
  – READ/WRITE transactions → locks

Concurrency Control by Validation

• Each transaction T defines a read set RS(T) and a write set WS(T)
• Each transaction proceeds in three phases:
  – Read all elements in RS(T). Time = START(T)
  – Validate (may need to rollback). Time = VAL(T)
  – Write all elements in WS(T). Time = FIN(T)

Main invariant: the serialization order is VAL(T)
Avoid $r_T(X) - w_U(X)$ Conflicts

START(U)  \[\begin{array}{c}
U: \\
\text{Read phase} & \text{Validate} & \text{Write phase}
\end{array}\]

VAL(U)  \[\text{conflicts}\]

FIN(U)

IF $RS(T) \cap WS(U)$ and $FIN(U) > START(T)$  
(U has validated and U has not finished before T begun)  
Then ROLLBACK(T)

Avoid $w_T(X) - w_U(X)$ Conflicts

START(U)  \[\begin{array}{c}
U: \\
\text{Read phase} & \text{Validate} & \text{Write phase}
\end{array}\]

VAL(U)  \[\text{conflicts}\]

FIN(U)

START(T)

IF $WS(T) \cap WS(U)$ and $FIN(U) > VAL(T)$  
(U has validated and U has not finished before T validates)  
Then ROLLBACK(T)
Final comments

- Locks and timestamps: SQL Server, DB2
- Validation: Oracle

(more or less)