CS522 Advanced Database Systems
Failure Recovery

Chengyu Sun
California State University, Los Angeles

Failure Model

- System crash
  - CPU halts
  - Memory lost
  - *Data on disk is OK*
- Everything else

Recover from System Crash

- Data recovery
- Recovery incomplete transactions
  - Removed changes made by un-committed transaction – Undo
  - Reapply changes made by committed transactions – Redo

Logging

- Log
  - A sequence of log records
  - Append only
  - *Undo*, *Redo*, *Undo/Redo*
- Common log records
  - <START T>
  - <COMMIT T>
  - <ABORT T>

Undo Logging

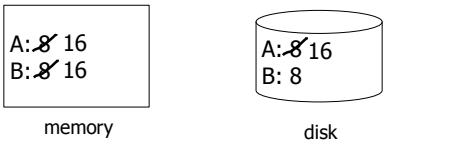
- Log records
  - <T,X,v>
- Rules
  - U_1: write <T,X,v> before writing new value of X *to disk*
  - U_2: write <COMMIT T> after writing all new values *to disk*

Undo Logging Example

- Write log record
  - <START T>
  - <COMMIT T>
  - <ABORT T>
  - <T,X,v>
- Flush log

(1) →
READ(A,t)
(2) →
t = t*2
(3) →
WRITE(A, t)
(4) →
READ(B,t)
(5) →
OUTPUT(A)
(6) →
OUTPUT(B)
(7) →

T_1: Read (A,t); t ← t×2
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B); failure!



HGM Notes

Undo Recovery

- Scan the complete log, and undo changes made by transactions that do not have a <COMMIT T> record
- Implementation issues
  - Interleaving records
  - Scan order
    - latest to earliest??
    - earliest to latest??

More about Undo Recovery

- System failure during recovery
  - Undo recovery is idempotent
- Checkpointing
  - Simple checkpointing
    - <CKPT>

Nonquiescent Checkpointing

- Write <START CKPT(T_1,...,T_k)> and *flush log*
  - T_1,...,T_k are *active* transactions, e.g. transactions that have not completed
- Wait until T1,...,Tk commit or abort
- Write <END CKPT> and *flush log*

Redo Logging

- Log records
  - <T,X,v>
- Rules
  - R1: write <T,X,v> and <COMMIT T> before writing new value of X to disk

Redo Logging Example

- Write log record
  - <START T>
  - <COMMIT T>
  - <ABORT T>
  - <T,X,v>
- Flush log

(1) →
READ(A,t)
(2) →
t = t*2
(3) →
WRITE(A, t)
(4) →
READ(B,t)
(5) →
OUTPUT(A)
(6) →
OUTPUT(B)
(7) →

Redo Recovery

- Scan the complete log, and re-apply changes made by transactions that have a <COMMIT T> record
- Implementation issues
  - Redo order
    - latest to earliest??
    - earliest to latest??

Checkpointing a Redo Log

- Write <START CKPT(T_1,...,T_k)> and flush log
- ??
- Write <END CKPT> and flush log

Redo Checkpointing Example

<START T1>
...
<START T2>
...
<START T3>
...
<COMMIT T2>
...
<START CKPT(??)>
??
<END CKPT>
...

there is a <END CKPT> in log??
there is no <END CKPT> in log??

Undo vs. Redo

- Undo: I/O may be too *frequent*
- Redo: I/O may be too *infrequent*
- Both: may lead to contradictory requirements on buffer management

Undo/Redo Logging

- Log records
  - <T,X,v,w>
- Rules
  - UR_1: write <T,X,v,w> before writing new value of X to disk

Undo/Redo Recovery

- Redo all the committed transactions
- Undo all the incomplete transactions

Checkpointing an Undo/Redo Log

- Write <START CKPT(T_1,...,T_k)>
- Flush *all* dirty pages
- Write <END CKPT>

Undo/Redo Checkpointing Example

<START T1>
...
<START T2>
...
<START T3>
...
<COMMIT T2>
...
<START CKPT(??)>
??
<END CKPT>
...

there is a <END CKPT> in log??
there is no <END CKPT> in log??