Motivation

• *Undo logging*: cannot bring backup DB copies up to date
• *Redo logging*: need to keep all modified blocks in memory until commit
• Solution: undo/redo logging

Update ⇒ \(<T_i, X, v, w>\)
\(v = \text{old value, } w = \text{new value}\)

**Undo/redo logging**

- Suppose xact \(T\) changes DB element \(X\) from \(v\) (old value) to \(w\) (new value)
  - Write log record \(<T, X, v, w>\)
- Rule:
  - *Before* modifying any db element \(X\) \((v \rightarrow w)\) on disk changed by xact \(T\), write the update log record \(<T, X, v, w>\) on disk
  - *<T COMMIT>* can precede or follow any db-element change on disk

**Example**

\(T_1: \)
Read \((A, t); t \leftarrow t \times 2\)
\(A \leftarrow B\)
Write \((A, t);\)
Read \((B, t); t \leftarrow t \times 2\)
Write \((B, t);\)
Output \((A);\)
Output \((B);\)

Notice: *<T1, COMMIT>* log record can be before or after the two output operations
**Undo/redo Recovery**

- Redo all the committed xacts in the order earliest-first (forward)
- Undo all the incomplete xacts in the order latest-first (backward)

---

**Example 1**

T: Read (A,t); t ← t × 2; A = B
Write (A,t); Read (B,t); t ← t × 2; Write (B,t); Output (A); Output (B);

Recovery: T1 is committed. Redo: write 16 to both A and B on disk.

---

**Example 2**

T: Read (A,t); t ← t × 2; A = B
Write (A,t); Read (B,t); t ← t × 2; Write (B,t); Output (A); Output (B);

Recovery: same as before.

---

**Example 3**

T: Read (A,t); t ← t × 2; A = B
Write (A,t); Read (B,t); t ← t × 2; Write (B,t); Output (A); Output (B);

Recovery: T1 incomplete. Undo: old values ("8") of A and B are written to disk. Write <T1, abort> to the log.
Example 4

T1: Read (A,t); t ← t × 2 = B
Write (A,t);
Read (B,t); t ← t × 2
Write (B,t);
Output (A);
Output (B);

A:8
B:8

Example 5

T1: Read (A,t); t ← t × 2 = B
Write (A,t);
Read (B,t); t ← t × 2
Write (B,t);
Output (A);
Output (B);

A:8
B:8

Nonquiescent checkpoint: flexible

- Write (and flush) <START CKPT(T1,…,Tk)> log record, where T1,…,Tk are active xacts
- Flush to disk all modified data elements.
  - Recall that in redo logging, we only flush those data elements by committed xacts.
- Write and flush <END CKPT> log record

Example

Redo/Undo Log:
<T1, START>
<T1, A, 4, 5>
<T2, START>
<T1, COMMIT>
<T2, 8, 9, 10>
<START CKPT(T2)>
<T2, C, 14, 15>
<T3, START>
<T3, D, 19, 20>
<END CKPT>
<T2, COMMIT>
<T3, COMMIT>

+ Start checkpoint, T2 active
+ continue, accept new xacts,
make sure 8 = 10 by T2 is on disk,
and A = 5 by T1 is on disk.
+ end checkpoint
+ continue
Recovery: example 1

Redo/Undo Log:
• <T1, START>
• <T1, A, 4, 5>
• <T2, START>
• <T2, B, 9, 10>
• <START CKPT(T2)>
• <T2, C, 14, 15>
• <T3, START>
• <T3, D, 19, 20>
• <END CKPT>
• <T2, COMMIT>
• <T3, COMMIT>

- T2 and T3 already committed.
- Since we see <END CKPT> first (backward), T1's changes must be on disk. So T1 can be ignored.
- Redo T2 and T3, forward.
  - Do NOT need to look at the log records before <START CKPT(T2)>
  - Reason: their changes are already on disk.

Recovery: example 2

Redo/Undo Log:
• <T1, START>
• <T1, A, 4, 5>
• <T2, START>
• <T1, COMMIT>
• <T2, B, 9, 10>
• <START CKPT(T2)>
• <T2, C, 14, 15>
• <T3, START>
• <T3, D, 19, 20>
• <END CKPT>
• <T2, COMMIT>
• <T3, COMMIT>

- T2 committed.
- T3 incomplete.
- Redo T2 (forward): C = 15 (on disk)
- Undo T3 (backward): D = 19 (on disk)

Recovery: example 3

Redo/Undo Log:
• <T1, START>
• <T1, A, 4, 5>
• <T2, START>
• <T1, COMMIT>
• <T2, B, 9, 10>
• <START CKPT(T2)>
• <T2, C, 14, 15>
• <T3, START>
• <T3, D, 19, 20>
• <END CKPT>

- T2 and T3 incomplete.
- Undo T3 (backward): D = 19 (on disk)
- Undo T2 (backward): C = 14 and B = 9 (on disk)
- Therefore, once we write the <END CKPT> log record, we still need to keep the earliest <T1, START> log record for those T1's that we're active (when we started the CKPT) and incomplete during the CKPT.
  - In this case: <T2, START>

Recovery: example 3

Redo/Undo Log:
• <T1, START>
• <T1, A, 4, 5>
• <T2, START>
• <T1, COMMIT>
• <T2, B, 9, 10>
• <START CKPT(T2)>
• <T2, C, 14, 15>
• <T3, START>
• <T3, D, 19, 20>

- See <START CKPT(T2)> first (backward)
- T2 active and incomplete.
  - undo (backward): C = 14 and B = 9 (on disk)
- T3 incomplete.
  - undo (backward): D = 19 (on disk)
- T1 complete.
  - redo (forward): A = 5 (on disk)
Summary
• Deal with system failures
• Undo logging
• Redo logging
• Undo/Redo
• Checkpoints