Database System Concepts
Chapter 17: Recovery System

Departamento de Engenharia Informática
Instituto Superior Técnico

1st Semester
2007/2008

Slides (fortemente) baseados nos slides oficiais do livro
“Database System Concepts”
©Silberschatz, Korth and Sudarshan.
Outline

1. Concepts

2. Log-Based Recovery
   - Deferred Modification
   - Immediate Modification
   - Checkpoints

3. Concurrent Transactions

4. Buffering
Outline

1. Concepts
2. Log-Based Recovery
3. Concurrent Transactions
4. Buffering
Failure Classification

- **Transaction failure:**
  - **Logical errors:** transaction cannot complete due to some internal error condition
  - **System errors:** the database system must terminate an active transaction due to an error condition (e.g., deadlock)

- **System crash:** a power failure or other hardware or software failure causes the system to crash
  - **Fail-stop assumption:** non-volatile storage contents are assumed to not be corrupted by system crash
    - Database systems have numerous integrity checks to prevent corruption of disk data

- **Disk failure:** a head crash or similar disk failure destroys all or part of disk storage
  - Destruction is assumed to be detectable: disk drives use checksums to detect failures
Recovery algorithms are techniques to ensure database consistency and transaction atomicity and durability despite failures.

Recovery algorithms have two parts:

1. Actions taken during normal transaction processing to ensure enough information exists to recover from failures.
2. Actions taken after a failure to recover the database contents to a state that ensures atomicity, consistency and durability.
Storage Structure

- **Volatile storage:**
  - does not survive system crashes
  - examples: main memory, cache memory

- **Nonvolatile storage:**
  - survives system crashes
  - examples: disk, tape, flash memory, non-volatile (battery backed up) RAM

- **Stable storage:**
  - a mythical form of storage that survives all failures
  - approximated by maintaining multiple copies on distinct nonvolatile media
Data Access

- **Physical blocks**: blocks residing on the disk
- **Buffer blocks**: blocks residing temporarily in main memory
- Block movements between disk and main memory are initiated through the following two operations:
  - `input(B)` transfers the physical block `B` to main memory
  - `output(B)` transfers the buffer block `B` to the disk, and replaces the appropriate physical block there
- Each transaction $T_i$ has its private work-area in which local copies of all data items accessed and updated by it are kept
  - $T_i$’s local copy of a data item $X$ is called $x_i$
  - We assume, for simplicity, that each data item fits in, and is stored inside, a single block
Transaction transfers data items between system buffer blocks and its private work-area using the following operations:

- **read**($X$) assigns the value of data item $X$ to the local variable $x_i$
- **write**($X$) assigns the value of local variable $x_i$ to data item $X$ in the buffer block
- both these commands may necessitate the issue of an **input**($B_X$) instruction before the assignment, if the block $B_X$ in which $X$ resides is not already in memory

- **output**($B_X$) need not immediately follow **write**($X$)
- System can perform the output operation when it deems fit
Example of Data Access

Buffer Block A

Buffer Block B

Buffer

input(A)

output(B)

write(Y)

read(X)

x

Y

x_1

y_1

work area of T_1

work area of T_2

memory

A

B

disk
Recovery and Atomicity

- Modifying the database without ensuring that the transaction will commit may leave the database in an inconsistent state
  - Consider transaction $T_i$ that transfers 50 € from account $A$ to account $B$
  - Several output operations may be required for $T_i$—a failure may occur after one of these modifications have been made but before all of them are made

- To ensure atomicity despite failures, we first output information describing the modifications to stable storage without modifying the database itself

- We study two approaches:
  - log-based recovery
  - shadow-paging

- We assume (initially) that transactions run serially, that is, one after the other.
Outline

1. Concepts
2. Log-Based Recovery
   - Deferred Modification
   - Immediate Modification
   - Checkpoints
3. Concurrent Transactions
4. Buffering
A log is kept on stable storage

The log is a sequence of log records, and maintains a record of update activities on the database

We assume for now that log records are written directly to stable storage (that is, they are not buffered)

Two approaches using logs:

- Deferred database modification
- Immediate database modification
Log Records

- When transaction $T_i$ starts, it registers itself by writing a record
  
  $<T_i, \text{start}>$

- Before $T_i$ executes $\text{write}(X)$, a log record
  
  $<T_i, X, V_1, V_2>$

  is written, where $V_1$ is the value of $X$ before the write, and $V_2$ is the value *to be written* to $X$

- When $T_i$ finishes its last statement, the log record
  
  $<T_i, \text{commit}>$

  is written
The **deferred database modification** scheme records all modifications to the log, but defers all the writes to after partial commit.

Assume that transactions execute serially:

- Transaction starts by writing \(< T_i, \text{start} >\) record to log
- A **write**\((X)\) operation results in a log record \(< T_i, X, V >\) being written, where \(V\) is the new value for \(X\)
  - Note: old value is not needed for this scheme
- The write is not performed on \(X\) at this time, but is deferred
- When \(T_i\) partially commits, \(< T_i, \text{commit} >\) is written to the log
- Finally, the log records are read and used to actually execute the previously deferred writes
Transaction Recovery

- During recovery, a transaction needs to be redone if and only if both $< T_i, \text{start} >$ and $< T_i, \text{commit} >$ are in the log.
- Redoing a transaction $T_i$ sets the value of all data items updated by the transaction to the new values.
- Crashes can occur while:
  - the transaction is executing the original updates
  - while recovery action is being taken.
An Example

Log

\[ \langle T_0, \text{start} \rangle \]

\[ T_0 : \]
read(A)
A := A - 50
write(A)
read(B)
B := B + 50
write(B)

\[ T_1 : \]
read(C)
C := C - 100
write(C)
An Example

Database System Concepts

Log-Based Recovery

Deferred Modification

Immediate Modification

Checkpoints

Concurrent Transactions

Buffering

Log

\(< T_0, \text{start} >
\)< \(T_0, A, 950 \)>

\(T_0 : \)
\begin{align*}
\text{read}(A) \\
A & := A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B & := B + 50 \\
\text{write}(B)
\end{align*}

\(T_1 : \)
\begin{align*}
\text{read}(C) \\
C & := C - 100 \\
\text{write}(C)
\end{align*}
An Example

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Log Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$</td>
<td>$&lt; T_0, \text{start} &gt;$</td>
</tr>
<tr>
<td></td>
<td>$&lt; T_0, A, 950 &gt;$</td>
</tr>
<tr>
<td></td>
<td>$&lt; T_0, B, 2050 &gt;$</td>
</tr>
</tbody>
</table>

$T_0$:
- read($A$)
- $A := A - 50$
- write($A$)
- read($B$)
- $B := B + 50$
- write($B$)

$T_1$:
- read($C$)
- $C := C - 100$
- write($C$)
An Example

$T_0:$
- read($A$)
- $A := A - 50$
- write($A$)
- read($B$)
- $B := B + 50$
- write($B$)

$T_1:$
- read($C$)
- $C := C - 100$
- write($C$)

Log:
- $< T_0, \text{start} >$
- $< T_0, A, 950 >$
- $< T_0, B, 2050 >$

No redo actions need to be taken
An Example

$T_0:$
- read($A$)
- $A := A - 50$
- write($A$)
- read($B$)
- $B := B + 50$
- write($B$)

$T_1:$
- read($C$)
- $C := C - 100$
- write($C$)

Log

$< T_0, \text{start} >$
$< T_0, A, 950 >$
$< T_0, B, 2050 >$
$< T_0, \text{commit} >$
An Example

**Log**

$< T_0, \text{start} >$

$< T_0, A, 950 >$

$< T_0, B, 2050 >$

$< T_0, \text{commit} >$

$< T_1, \text{start} >$

---

$T_0 :$

**read**($A$)

$A := A - 50$

**write**($A$)

**read**($B$)

$B := B + 50$

**write**($B$)

---

$T_1 :$

**read**($C$)

$C := C - 100$

**write**($C$)
An Example

$T_0 :$

$\text{read}(A)$
$A := A - 50$
$\text{write}(A)$
$\text{read}(B)$
$B := B + 50$
$\text{write}(B)$

$T_1 :$

$\text{read}(C)$
$C := C - 100$
$\text{write}(C)$

Log

$< T_0, \text{start} >$
$< T_0, A, 950 >$
$< T_0, B, 2050 >$
$< T_0, \text{commit} >$
$< T_1, \text{start} >$
$< T_1, C, 600 >$
An Example

Log

\(< T_0, \text{start} >\)
\(< T_0, A, 950 >\)
\(< T_0, B, 2050 >\)
\(< T_0, \text{commit} >\)
\(< T_1, \text{start} >\)
\(< T_1, C, 600 >\)

\textbf{system failure!}

\[ T_0 : \]
\begin{align*}
\text{read}(A) \\
A &:= A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B &:= B + 50 \\
\text{write}(B)
\end{align*}

\[ T_1 : \]
\begin{align*}
\text{read}(C) \\
C &:= C - 100 \\
\text{write}(C)
\end{align*}

\textbf{redo}(T_0) \text{ must be performed since }\]
\(< T_0, \text{commit} >\) \text{ is present}
An Example

**Log**

\[ < T_0, \text{start} > \]
\[ < T_0, A, 950 > \]
\[ < T_0, B, 2050 > \]
\[ < T_0, \text{commit} > \]
\[ < T_1, \text{start} > \]
\[ < T_1, C, 600 > \]
\[ < T_1, \text{commit} > \]

**T_0**:
- read(A)
- \( A := A - 50 \)
- write(A)
- read(B)
- \( B := B + 50 \)
- write(B)

**T_1**:
- read(C)
- \( C := C - 100 \)
- write(C)
An Example

Concepts
Log-Based Recovery
Deferred Modification
Immediate Modification
Checkpoints
Concurrent Transactions
Buffering

Database System Concepts

$T_0$:
- read($A$)
  - $A := A - 50$
- write($A$)
- read($B$)
  - $B := B + 50$
- write($B$)

$T_1$:
- read($C$)
  - $C := C - 100$
- write($C$)

Log

$< T_0, \text{start} >$
$< T_0, A, 950 >$
$< T_0, B, 2050 >$
$< T_0, \text{commit} >$
$< T_1, \text{start} >$
$< T_1, C, 600 >$
$< T_1, \text{commit} >$

system failure!

redo($T_0$) must be performed, followed by redo($T_1$) since $< T_0, \text{commit} >$ and $< T_1, \text{commit} >$ are present.
Immediate Database Modification

- The **immediate database modification** scheme allows database updates of an uncommitted transaction to be made as the writes are issued.
- Update log record must be written before database item is written.
  - We assume that the log record is output directly to stable storage.
  - Can be extended to postpone log record output, so long as prior to execution of an `output(B)` operation for a data block $B$, all log records corresponding to items $B$ must be flushed to stable storage.
- Output of updated blocks can take place at any time before or after transaction commit.
- Order in which blocks are output can be different from the order in which they are written.
Transaction Recovery

- Recovery procedure has two operations instead of one:
  - **undo**($T_i$) restores the value of all data items updated by $T_i$ to their old values, going backwards from the last log record for $T_i$
  - **redo**($T_i$) sets the value of all data items updated by $T_i$ to the new values, going forward from the first log record for $T_i$

  Both operations must be *idempotent* (even if the operation is executed multiple times the effect is the same as if it is executed once)

- When recovering after failure:
  - Transaction $T_i$ needs to be **undone** if the log contains the record $< T_i, \text{start} >$, but does not contain the record $< T_i, \text{commit} >$
  - Transaction $T_i$ needs to be **redone** if the log contains both the record $< T_i, \text{start} >$ and the record $< T_i, \text{commit} >$

- Undo operations are performed first, then redo operations
An Example

Log-Based Recovery
Deferred Modification
Immediate Modification
Checkpoints
Concurrent Transactions
Buffering

---

$T_0:$

- read($A$)
- $A := A - 50$
- write($A$)
- read($B$)
- $B := B + 50$
- write($B$)

$T_1:$

- read($C$)
- $C := C - 100$
- write($C$)

Log:

$< T_0, \text{start} >$
An Example

Log

\[ T_0 : \]
\[
\text{read}(A) \\
A := A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B := B + 50 \\
\text{write}(B)
\]

\[ T_1 : \]
\[
\text{read}(C) \\
C := C - 100 \\
\text{write}(C)
\]

Log
\[ < T_0, \text{start} > \\
< T_0, A, 1000, 950 > \]
An Example

**Database System Concepts**

**Log-Based Recovery**

**Deferred Modification**

**Immediate Modification**

**Checkpoints**

**Concurrent Transactions**

**Buffering**

---

**An Example**

### Log

\[ \langle T_0, \text{start} \rangle, \langle T_0, A, 1000, 950 \rangle, \langle T_0, B, 2000, 2050 \rangle \]

---

**Transaction T0**

- read(A)
- \( A := A - 50 \)
- write(A)
- read(B)
- \( B := B + 50 \)
- write(B)

---

**Transaction T1**

- read(C)
- \( C := C - 100 \)
- write(C)
An Example

Log

\[ T_0 : \]
- read(A)
- A := A - 50
- write(A)
- read(B)
- B := B + 50
- write(B)

\[ T_1 : \]
- read(C)
- C := C - 100
- write(C)

\[ \log < T_0, \text{start} > \]
\[ < T_0, A, 1000, 950 > \]
\[ < T_0, B, 2000, 2050 > \]

system failure!

undo(\(T_0\)): B is restored to 2000 and A to 1000
An Example

Log

\[
T_0 : \quad \begin{array}{l}
\text{read}(A) \\
A := A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B := B + 50 \\
\text{write}(B)
\end{array}
\]

\[
T_1 : \quad \begin{array}{l}
\text{read}(C) \\
C := C - 100 \\
\text{write}(C)
\end{array}
\]

\[
\text{Log} \quad < T_0, \text{start} > \\
< T_0, A, 1000, 950 > \\
< T_0, B, 2000, 2050 > \\
< T_0, \text{commit} >
\]
An Example

Log

\[
\text{\textless} T_0, \text{start} > \\
\text{\textless} T_0, A, 1000, 950 > \\
\text{\textless} T_0, B, 2000, 2050 > \\
\text{\textless} T_0, \text{commit} > \\
\text{\textless} T_1, \text{start} > \\
\]

\[
T_0 : \\
\underline{\text{read}(A)} \\
A := A - 50 \\
\underline{\text{write}(A)} \\
\underline{\text{read}(B)} \\
B := B + 50 \\
\underline{\text{write}(B)} \\
\]

\[
T_1 : \\
\underline{\text{read}(C)} \\
C := C - 100 \\
\underline{\text{write}(C)} \\
\]

An Example

Log

\[
\text{\textless} T_0, \text{start} > \\
\text{\textless} T_0, A, 1000, 950 > \\
\text{\textless} T_0, B, 2000, 2050 > \\
\text{\textless} T_0, \text{commit} > \\
\text{\textless} T_1, \text{start} > \\
\]

\[
T_0 : \\
\underline{\text{read}(A)} \\
A := A - 50 \\
\underline{\text{write}(A)} \\
\underline{\text{read}(B)} \\
B := B + 50 \\
\underline{\text{write}(B)} \\
\]

\[
T_1 : \\
\underline{\text{read}(C)} \\
C := C - 100 \\
\underline{\text{write}(C)} \\
\]
An Example

Database System Concepts

Log-Based Recovery
Deferred Modification
Immediate Modification
Checkpoints
Concurrent Transactions
Buffering

$T_0 :$
- read($A$)
  $A := A - 50$
- write($A$)
- read($B$)
  $B := B + 50$
- write($B$)

$T_1 :$
- read($C$)
  $C := C - 100$
- write($C$)

Log
- $T_0, \text{start} >$
- $T_0, A, 1000, 950 >$
- $T_0, B, 2000, 2050 >$
- $T_0, \text{commit} >$
- $T_1, \text{start} >$
- $T_1, C, 700, 600 >$
**An Example**

**Log**

\[< T_0, \text{start} >\]
\[< T_0, A, 1000, 950 >\]
\[< T_0, B, 2000, 2050 >\]
\[< T_0, \text{commit} >\]
\[< T_1, \text{start} >\]
\[< T_1, C, 700, 600 >\]

**system failure!**

\[\text{undo}(T_1) \text{ and redo}(T_0): \] C is restored to 700, and then A and B are set to 950 and 2050 respectively.

\[T_0: \]
\[\begin{align*}
\text{read}(A) \\
A &:= A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B &:= B + 50 \\
\text{write}(B)
\end{align*}\]

\[T_1: \]
\[\begin{align*}
\text{read}(C) \\
C &:= C - 100 \\
\text{write}(C)
\end{align*}\]
An Example

$T_0$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>read $A$</td>
</tr>
<tr>
<td></td>
<td>$A := A - 50$</td>
</tr>
<tr>
<td>write</td>
<td>write $A$</td>
</tr>
<tr>
<td>read</td>
<td>read $B$</td>
</tr>
<tr>
<td></td>
<td>$B := B + 50$</td>
</tr>
<tr>
<td>write</td>
<td>write $B$</td>
</tr>
</tbody>
</table>

$T_1$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>read $C$</td>
</tr>
<tr>
<td></td>
<td>$C := C - 100$</td>
</tr>
<tr>
<td>write</td>
<td>write $C$</td>
</tr>
</tbody>
</table>

Log

- $< T_0, \text{start} >$
- $< T_0, A, 1000, 950 >$
- $< T_0, B, 2000, 2050 >$
- $< T_0, \text{commit} >$
- $< T_1, \text{start} >$
- $< T_1, C, 700, 600 >$
- $< T_1, \text{commit} >$
An Example

Log

\(< T_0, \text{start} > \>
\,< T_0, A, 1000, 950 > \>
\,< T_0, B, 2000, 2050 > \>
\,< T_0, \text{commit} > \>
\,< T_1, \text{start} > \>
\,< T_1, C, 700, 600 > \>
\,< T_1, \text{commit} > \>
\,system failure!

\( T_0 : \)
\[
\begin{align*}
\text{read}(A) \\
A & := A - 50 \\
\text{write}(A) \\
\text{read}(B) \\
B & := B + 50 \\
\text{write}(B)
\end{align*}
\]

\( T_1 : \)
\[
\begin{align*}
\text{read}(C) \\
C & := C - 100 \\
\text{write}(C)
\end{align*}
\]

\textbf{redo}(T_0) \text{ and redo}(T_1): \ A \text{ and } B \text{ are set to 950 and 2050, respectively, then } C \text{ is set to 600}
Checkpoints

- Problems in recovery procedure:
  - searching the entire log is time-consuming
  - we might unnecessarily redo transactions which have already output their updates to the database.

- Streamline recovery procedure by periodically performing **checkpointing**
  - Output all log records currently residing in main memory onto stable storage
  - Output all modified buffer blocks to the disk
  - Write a log record `<checkpoint>` onto stable storage
Recovery using Checkpoints

- During recovery we need to consider only the most recent transaction $T_i$ that started before the checkpoint, and transactions that started after $T_i$
  - 1. Scan backwards from end of log to find the most recent $<\text{checkpoint}>$ record
  - 2. Continue scanning backwards till a record $<T_i, \text{start}>$ is found
    - Need only consider the part of log following this record; earlier part of log can be ignored, and can be erased whenever desired
  - 3. For all transactions starting from $T_i$ or later with no $<T_i, \text{commit}>$, execute $\text{undo}(T_i)$
    - Done only in case of immediate modification
  - 4. Scanning forward in the log, for all transactions starting from $T_i$ or later with a $<T_i, \text{commit}>$, execute $\text{redo}(T_i)$
Checkpointing Example

- $T_1$ can be ignored (updates already output to disk)
- $T_4$ undone
- $T_2$ and $T_3$ redone
1 Concepts
2 Log-Based Recovery
3 Concurrent Transactions
4 Buffering
Recovery With Concurrent Transactions

- We modify the log-based recovery schemes to allow multiple transactions to execute concurrently
  - All transactions share a single disk buffer and a single log
  - A buffer block can have data items updated by one or more transactions
- We assume concurrency control using strict two-phase locking;
  - I.e. the updates of uncommitted transactions should not be visible to other transactions
    - Otherwise how to perform undo if $T_1$ updates $A$, then $T_2$ updates $A$ and finally $T_1$ has to abort?
- Logging is done as described earlier
  - Log records of different transactions may be interspersed in the log
The checkpointing technique and actions taken on recovery have to be changed since several transactions may be active when a checkpoint is performed.

Checkpoints are performed as before, except that the checkpoint log record is now of the form

\[
\langle \text{checkpoint } L \rangle
\]

where \( L \) is the list of transactions active at the time of the checkpoint.

For now we assume no updates are in progress while the checkpoint is carried out.
When the system recovers from a crash, it first does the following:

1. Initialize *undo-list* and *redo-list* to empty.
2. Scan the log backwards from the end, stopping when the first `<checkpoint L>` record is found.
3. For each record found during the backward scan:
   - if the record is `<T_i, commit>`, add $T_i$ to *redo-list*.
   - if the record is `<T_i, start>`, then if $T_i$ is not in *redo-list*, add $T_i$ to *undo-list*.
4. For every $T_i$ in $L$, if $T_i$ is not in *redo-list*, add $T_i$ to *undo-list*. 
At this point *undo-list* consists of incomplete transactions which must be undone, and *redo-list* consists of finished transactions that must be redone.

Recovery now continues as follows:

1. Scan log backwards from most recent record, stopping when \( < T_i, \text{start} > \) records have been encountered for every \( T_i \) in *undo-list*.
   - During the scan, perform *undo* for each log record that belongs to a transaction in *undo-list*.

2. Locate the most recent \( < \text{checkpoint } L > \) record.

3. Scan log forwards from the \( < \text{checkpoint } L > \) record till the end of the log.
   - During the scan, perform *redo* for each log record that belongs to a transaction on *redo-list*.
Example of Recovery

Log

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>initial values</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
</tr>
</tbody>
</table>

- A
- B
- C
- D
- E

Scan log backwards
Perform undo
Perform redo

Database System Concepts

Log-Based Recovery

Concurrent Transactions

Buffering
Example of Recovery

Scan log backwards

1. Perform undo
2. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_3 )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

Log

- \( T_0, \text{start} \)
- \( T_0, A, 0, 10 \)
- \( T_0, \text{commit} \)
- \( T_1, \text{start} \)
- \( T_1, B, 0, 10 \)
- \( T_2, \text{start} \)
- \( T_2, C, 0, 10 \)
- \( T_2, C, 10, 20 \)
- \( \text{checkpoint} \{ T_1, T_2 \} \)
- \( T_3, \text{start} \)
- \( T_3, A, 10, 20 \)
- \( T_4, \text{start} \)
- \( T_3, D, 0, 10 \)
- \( T_4, E, 0, 10 \)
- \( T_3, \text{commit} \)

1. Scan log backwards
2. Perform undo
3. Perform redo

\[
\begin{array}{c|c}
\text{redo-list} & \text{undo-list} \\
\hline
T_3 \\
\end{array}
\]

\[
\begin{array}{cccccc}
A & B & C & D & E \\
\hline
\text{initial values} & 0 & 0 & 0 & 0 & 0 \\
\text{current value} & 20 & 10 & 20 & 10 & 10 \\
\end{array}
\]
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log:

\[
\begin{align*}
&< T_0, \text{start} > \\
&< T_0, A, 0, 10 > \\
&< T_0, \text{commit} > \\
&< T_1, \text{start} > \\
&< T_1, B, 0, 10 > \\
&< T_2, \text{start} > \\
&< T_2, C, 0, 10 > \\
&< T_2, C, 10, 20 > \\
&< \text{checkpoint} \{ T_1, T_2 \} > \\
&< T_3, \text{start} > \\
&< T_3, A, 10, 20 > \\
&< T_4, \text{start} > \\
&< T_3, D, 0, 10 > \\
&< T_4, E, 0, 10 > \\
&< T_3, \text{commit} >
\end{align*}
\]

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_3)</td>
<td>(T_3)</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccc}
A & B & C & D & E \\
\hline
\text{initial values} & 0 & 0 & 0 & 0 & 0 \\
\text{current value} & 20 & 10 & 20 & 10 & 10 \\
\end{array}
\]
Example of Recovery

Log

< T₀, start >
< T₀, A, 0, 10 >
< T₀, commit >
< T₁, start >
< T₁, B, 0, 10 >
< T₂, start >
< T₂, C, 0, 10 >
< T₂, C, 10, 20 >
< checkpoint { T₁, T₂ } >
< T₃, start >
< T₃, A, 10, 20 >
< T₄, start >
< T₃, D, 0, 10 >
< T₄, E, 0, 10 >
< T₃, commit >

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₃</td>
<td>T₄</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

Log

\[
\begin{align*}
<T_0, \text{start}> \\
<T_0, A, 0, 10> \\
<T_0, \text{commit}> \\
<T_1, \text{start}> \\
<T_1, B, 0, 10> \\
<T_2, \text{start}> \\
<T_2, C, 0, 10> \\
<T_2, C, 10, 20> \\
\text{checkpoint} \{T_1, T_2\} \\
<T_3, \text{start}> \\
<T_3, A, 10, 20> \\
<T_4, \text{start}> \\
<T_3, D, 0, 10> \\
<T_4, E, 0, 10> \\
<T_3, \text{commit}> \\
\end{align*}
\]

1. Scan log backwards
2. Perform undo
3. Perform redo

\[
\begin{array}{c|c}
\text{redo-list} & \text{undo-list} \\
\hline
T_3 & T_4 \\
\end{array}
\]

\[
\begin{array}{ccccc}
A & B & C & D & E \\
\hline
\text{initial values} & 0 & 0 & 0 & 0 & 0 \\
\text{current value} & 20 & 10 & 20 & 10 & 10 \\
\end{array}
\]
Example of Recovery

Log

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
<td>$T_4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

|< $T_0$, start >|
|< $T_0$, A, 0, 10 >|
|< $T_0$, commit >|
|< $T_1$, start >|
|< $T_1$, B, 0, 10 >|
|< $T_2$, start >|
|< $T_2$, C, 0, 10 >|
|< $T_2$, C, 10, 20 >|
|< checkpoint $\{T_1, T_2\}$ >|
|< $T_3$, start >|
|< $T_3$, A, 10, 20 >|
|< $T_4$, start >|
|< $T_3$, D, 0, 10 >|
|< $T_4$, E, 0, 10 >|
|< $T_3$, commit >|
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

**Log**

- `<T_0, start>`
- `<T_0, A, 0, 10>`
- `<T_0, commit>`
- `<T_1, start>`
- `<T_1, B, 0, 10>`
- `<T_2, start>`
- `<T_2, C, 0, 10>`
- `<T_2, C, 10, 20>`
- `<checkpoint {T_1, T_2}>`
- `<T_3, start>`
- `<T_3, A, 10, 20>`
- `<T_4, start>`
- `<T_3, D, 0, 10>`
- `<T_4, E, 0, 10>`
- `<T_3, commit>`

**redo-list**

<table>
<thead>
<tr>
<th>redo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_3</td>
</tr>
<tr>
<td>T_4</td>
</tr>
<tr>
<td>T_1</td>
</tr>
<tr>
<td>T_2</td>
</tr>
</tbody>
</table>

**undo-list**

<table>
<thead>
<tr>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_3</td>
</tr>
<tr>
<td>T_4</td>
</tr>
</tbody>
</table>

**A B C D E**

| initial values | 0 | 0 | 0 | 0 | 0 |
| current value  | 20| 10| 20| 10| 10|
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
<td>$T_4$</td>
</tr>
<tr>
<td></td>
<td>$T_1$</td>
</tr>
<tr>
<td></td>
<td>$T_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Initial values

Current value

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

Scan log backwards
Perform undo
Perform redo

Log

\[
\begin{align*}
\langle T_0, \text{start} \rangle \\
\langle T_0, A, 0, 10 \rangle \\
\langle T_0, \text{commit} \rangle \\
\langle T_1, \text{start} \rangle \\
\langle T_1, B, 0, 10 \rangle \\
\langle T_2, \text{start} \rangle \\
\langle T_2, C, 0, 10 \rangle \\
\langle T_2, C, 10, 20 \rangle \\
\langle \text{checkpoint} \{ T_1, T_2 \} \rangle \\
\langle T_3, \text{start} \rangle \\
\langle T_3, A, 10, 20 \rangle \\
\langle T_4, \text{start} \rangle \\
\langle T_3, D, 0, 10 \rangle \\
\langle T_4, E, 0, 10 \rangle \\
\langle T_3, \text{commit} \rangle \\
\end{align*}
\]

\[
\begin{array}{c|c}
\text{redo-list} & \text{undo-list} \\
\hline
T_3 & T_4 \\
T_1 & T_2 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
A & B & C & D & E \\
\hline
\text{initial values} & 0 & 0 & 0 & 0 & 0 \\
\text{current value} & 20 & 10 & 20 & 10 & 0 \\
\end{array}
\]
Example of Recovery

Log

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_3 )</td>
<td>( T_4 )</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>( T_2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log

\[ \begin{align*}
< T_0, \text{start} > \\
< T_0, A, 0, 10 > \\
< T_0, \text{commit} > \\
< T_1, \text{start} > \\
< T_1, B, 0, 10 > \\
< T_2, \text{start} > \\
< T_2, C, 0, 10 > \\
< T_2, C, 10, 20 > \\
< \text{checkpoint} \{ T_1, T_2 \} > \\
< T_3, \text{start} > \\
< T_3, A, 10, 20 > \\
< T_4, \text{start} > \\
< T_3, D, 0, 10 > \\
< T_4, E, 0, 10 > \\
< T_3, \text{commit} > \\
\end{align*} \]

\begin{align*}
\text{redo-list} & \quad \text{undo-list} \\
T_3 & \quad T_4 \\
T_1 & \\
T_2 & \\
\end{align*}

\begin{tabular}{c|cccccc}
 & A & B & C & D & E \\
\hline
initial values & 0 & 0 & 0 & 0 & 0 \\
current value & 20 & 10 & 20 & 10 & 0 \\
\end{tabular}
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log:

- \(< T_0, \text{start} >\)
- \(< T_0, A, 0, 10 >\)
- \(< T_0, \text{commit} >\)
- \(< T_1, \text{start} >\)
- \(< T_1, B, 0, 10 >\)
- \(< T_2, \text{start} >\)
- \(< T_2, C, 0, 10 >\)
- \(< T_2, C, 10, 20 >\)
- \(< \text{checkpoint} \{ T_1, T_2 \} >\)
- \(< T_3, \text{start} >\)
- \(< T_3, A, 10, 20 >\)
- \(< T_4, \text{start} >\)
- \(< T_3, D, 0, 10 >\)
- \(< T_4, E, 0, 10 >\)
- \(< T_3, \text{commit} >\)

\[
\begin{array}{c|c}
\text{redo-list} & \text{undo-list} \\
\hline
T_3 & T_4 \\
T_1 & T_2 \\
\end{array}
\]

A | B | C | D | E
---|---|---|---|---
initial values: 0 | 0 | 0 | 0 | 0 | 0
current value: 20 | 10 | 20 | 10 | 0
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

**Log**

- `<T₀, start>`
- `<T₀, A, 0, 10>`
- `<T₀, commit>`
- `<T₁, start>`
- `<T₁, B, 0, 10>`
- `<T₂, start>`
- `<T₂, C, 0, 10>`
- `<T₂, C, 10, 20>`
- `<checkpoint {T₁, T₂}>`
- `<T₃, start>`
- `<T₃, A, 10, 20>`
- `<T₄, start>`
- `<T₃, D, 0, 10>`
- `<T₄, E, 0, 10>`
- `<T₃, commit>`

**redo-list**

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₃</td>
<td>T₄</td>
</tr>
<tr>
<td></td>
<td>T₁</td>
</tr>
<tr>
<td></td>
<td>T₂</td>
</tr>
</tbody>
</table>

**Current values**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

Log

1. Scan log backwards
2. Perform undo
3. Perform redo

redo-list

undo-list

<table>
<thead>
<tr>
<th>redo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
</tr>
</tbody>
</table>

A | B | C | D | E
---|---|---|---|---
initial values | 0 | 0 | 0 | 0 | 0
current value  | 20| 10| 20| 10| 0

Scan log backwards
Perform undo
Perform redo

A | B | C | D | E
---|---|---|---|---
initial values | 0 | 0 | 0 | 0 | 0
current value  | 20| 10| 20| 10| 0
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log:

\[
\begin{align*}
< T_0, \text{start} > \\
< T_0, A, 0, 10 > \\
< T_0, \text{commit} > \\
< T_1, \text{start} > \\
< T_1, B, 0, 10 > \\
< T_2, \text{start} > \\
< T_2, C, 0, 10 > \\
< T_2, C, 10, 20 > \\
< \text{checkpoint} \{ T_1, T_2 \} > \\
< T_3, \text{start} > \\
< T_3, A, 10, 20 > \\
< T_4, \text{start} > \\
< T_3, D, 0, 10 > \\
< T_4, E, 0, 10 > \\
< T_3, \text{commit} >
\end{align*}
\]

redo-list | undo-list
---|---
\(T_3\) | \(T_4\)
\(T_1\) | \(T_2\)

The table shows the initial values and current values for columns A, B, C, D, and E:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

---

**Log**

< $T_0$, **start** >
< $T_0$, A, 0, 10 >
< $T_0$, **commit** >
< $T_1$, **start** >
< $T_1$, B, 0, 10 >
< $T_2$, **start** >
< $T_2$, C, 0, 10 >
< $T_2$, C, 10, 20 >
< **checkpoint** { $T_1$, $T_2$ } >
< $T_3$, **start** >
< $T_3$, A, 10, 20 >
< $T_4$, **start** >
< $T_3$, D, 0, 10 >
< $T_4$, E, 0, 10 >
< $T_3$, **commit** >

---

**redo-list** | **undo-list**
---|---
$T_3$ | $T_4$
$T_1$ | $T_2$

---

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log

\[
\begin{align*}
&T_0, \text{start} > \\
&T_0, A, 0, 10 > \\
&T_0, \text{commit} > \\
&T_1, \text{start} > \\
&T_1, B, 0, 10 > \\
&T_2, \text{start} > \\
&T_2, C, 0, 10 > \\
&T_2, C, 10, 20 > \\
&\text{checkpoint } \{ T_1, T_2 \} > \\
&T_3, \text{start} > \\
&T_3, A, 10, 20 > \\
&T_4, \text{start} > \\
&T_3, D, 0, 10 > \\
&T_4, E, 0, 10 > \\
&T_3, \text{commit} >
\end{align*}
\]

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_3 )</td>
<td>( T_4 )</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>( T_2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log

```
< T0, start >
< T0, A, 0, 10 >
< T0, commit >
< T1, start >
< T1, B, 0, 10 >
< T2, start >
< T2, C, 0, 10 >
< T2, C, 10, 20 >
< checkpoint { T1, T2 } >
< T3, start >
< T3, A, 10, 20 >
< T4, start >
< T3, D, 0, 10 >
< T4, E, 0, 10 >
< T3, commit >
```

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

Log:

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_3</td>
<td>T_4</td>
</tr>
<tr>
<td></td>
<td>T_1</td>
</tr>
<tr>
<td></td>
<td>T_2</td>
</tr>
</tbody>
</table>

A | B | C | D | E
---|---|---|---|---
initial values: 0 | 0 | 0 | 0 | 0 | 0
current value: 20 | 0 | 0 | 10 | 0 | 0
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>initial values</th>
<th>current value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

### Log

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$, <strong>start</strong></td>
<td>$T_0$, $A$, 0, 10</td>
<td>$T_0$, <strong>commit</strong></td>
<td>$T_1$, <strong>start</strong></td>
<td>$T_1$, $B$, 0, 10</td>
</tr>
<tr>
<td>$T_2$, <strong>start</strong></td>
<td>$T_2$, $C$, 0, 10</td>
<td>$T_2$, $C$, 10, 20</td>
<td><strong>checkpoint</strong> ${T_1$, $T_2}$</td>
<td>$T_2$, <strong>start</strong></td>
</tr>
<tr>
<td>$T_3$, <strong>start</strong></td>
<td>$T_3$, $A$, 10, 20</td>
<td>$T_4$, <strong>start</strong></td>
<td>$T_3$, $D$, 0, 10</td>
<td>$T_4$, $E$, 0, 10</td>
</tr>
<tr>
<td>$T_3$, <strong>commit</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### redo-list

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
</tbody>
</table>

### undo-list

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Example of Recovery

Log

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₃</td>
<td>T₄</td>
</tr>
<tr>
<td>T₁</td>
<td>T₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

initial values:

- A: 0
- B: 0
- C: 0
- D: 0
- E: 0

current value:

- A: 20
- B: 0
- C: 0
- D: 10
- E: 0
Example of Recovery

Log

<table>
<thead>
<tr>
<th>Event</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$, start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_0$, A, 0, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_0$, commit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_1$, start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_1$, B, 0, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_2$, start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_2$, C, 0, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_2$, C, 10, 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>checkpoint {$T_1$, $T_2$}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_3$, start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_3$, A, 10, 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_4$, start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_3$, D, 0, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_4$, E, 0, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_3$, commit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Scan log backwards
2. Perform undo
3. Perform redo

redo-list

<table>
<thead>
<tr>
<th>redo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
</tr>
</tbody>
</table>

undo-list

<table>
<thead>
<tr>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
</tr>
</tbody>
</table>

A | B | C | D | E
---|---|---|---|---
initial values | 0 | 0 | 0 | 0 | 0

current value | 20 | 0 | 0 | 10 | 0
Example of Recovery

1. Scan log backwards
2. Perform undo
3. Perform redo

Log:
- `<T_0, start>`
- `<T_0, A, 0, 10>`
- `<T_0, commit>`
- `<T_1, start>`
- `<T_1, B, 0, 10>`
- `<T_2, start>`
- `<T_2, C, 0, 10>`
- `<T_2, C, 10, 20>`
- `<checkpoint \{T_1, T_2\}>`
- `<T_3, start>`
- `<T_3, A, 10, 20>`
- `<T_4, start>`
- `<T_3, D, 0, 10>`
- `<T_4, E, 0, 10>`
- `<T_3, commit>`

Redo-list: `T_3`

Undo-list: `T_4`, `T_1`, `T_2`

<table>
<thead>
<tr>
<th></th>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>T_4</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>T_1</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>T_2</code></td>
</tr>
</tbody>
</table>

Initial values:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Current value:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

**Log**

1. \(< T_0, \text{start} >\)
2. \(< T_0, A, 0, 10 >\)
3. \(< T_0, \text{commit} >\)
4. \(< T_1, \text{start} >\)
5. \(< T_1, B, 0, 10 >\)
6. \(< T_2, \text{start} >\)
7. \(< T_2, C, 0, 10 >\)
8. \(< T_2, C, 10, 20 >\)
9. \(< \text{checkpoint}\{ T_1, T_2 \} >\)
10. \(< T_3, \text{start} >\)
11. \(< T_3, A, 10, 20 >\)
12. \(< T_4, \text{start} >\)
13. \(< T_3, D, 0, 10 >\)
14. \(< T_4, E, 0, 10 >\)
15. \(< T_3, \text{commit} >\)

**redo-list**

- \(T_3\)
- \(T_4\)
- \(T_1\)
- \(T_2\)

**undo-list**

- \(T_3\)
- \(T_4\)
- \(T_1\)
- \(T_2\)

---

**Initial Values**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Current Value**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Recovery

Log

< T₀, start >
< T₀, A, 0, 10 >
< T₀, commit >
< T₁, start >
< T₁, B, 0, 10 >
< T₂, start >
< T₂, C, 0, 10 >
< T₂, C, 10, 20 >
< checkpoint { T₁, T₂ } >
< T₃, start >
< T₃, A, 10, 20 >
< T₄, start >
< T₃, D, 0, 10 >
< T₄, E, 0, 10 >
< T₃, commit >

1. Scan log backwards
2. Perform undo
3. Perform redo

<table>
<thead>
<tr>
<th>redo-list</th>
<th>undo-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₃</td>
<td>T₄</td>
</tr>
<tr>
<td>T₁</td>
<td>T₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial values</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>current value</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Outline

1. Concepts
2. Log-Based Recovery
3. Concurrent Transactions
4. Buffering
• With **log record buffering**, log records are buffered in main memory, instead of being output directly to stable storage.

• Log records are output to stable storage when a block of log records in the buffer is full, or a **log force** operation is executed.

• Log force is performed to commit a transaction by forcing all its log records (including the commit record) to stable storage.

• Several log records can thus be output using a single output operation, reducing the I/O cost.
The rules below must be followed if log records are buffered:

1. Log records are output to stable storage in the order in which they are created.
2. Transaction $T_i$ enters the *commit state* only when the log record $< T_i, \text{commit} >$ has been output to stable storage.
3. Before a block of data in main memory is output to the database, all log records pertaining to data in that block must have been output to stable storage.
   - This rule is called the *write-ahead logging* (or WAL) rule.
Database Buffering

- Database maintains an in-memory buffer of data blocks
  - When a new block is needed, if buffer is full an existing block needs to be removed from buffer
  - If the block chosen for removal has been updated, it must be output to disk

- As a result of the WAL rule, if a block with uncommitted updates is output to disk, log records for the updates are output to the log on stable storage first

- No updates should be in progress on a block when it is output to disk:
  - Before writing a data item, transaction acquires exclusive lock on block containing the data item
  - Lock can be released once the write is completed
    - Such locks held for short duration are called latches
  - Before a block is output to disk, the system acquires an exclusive latch on the block
End of Chapter 17