

Transaction Management Recovery

Vũ Tuyết Trinh

1

1

Learning objectives

•Upon completion of this lesson, students will be able to:

1. Understand recovery process
2. Be able to select a suitable recovery strategy

2

2

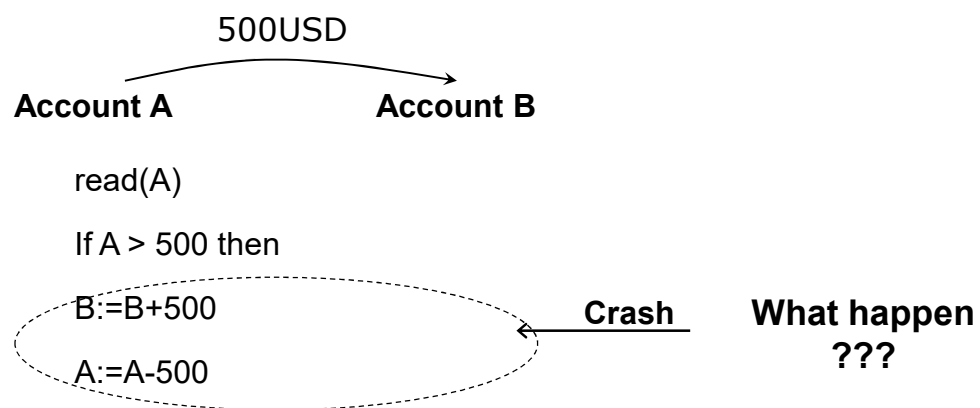
Outline

1. Transaction and Recovery
2. Failure
3. Transaction Log
4. Checkpoint

3

3

Example



4

4

1. Transaction & Recovery

1.1. Objective

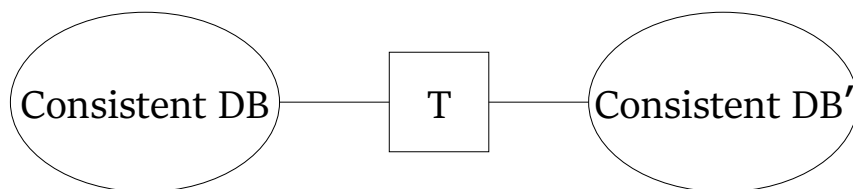
1.2. Problems

5

5

1.1. Objective

- Collection of action that preserve consistency



with assumption

IF T starts with consistent state +
 T executes in isolation
THEN T leaves consistent state

6

6

1.2. Problems

- Constraint violation?
 - Transaction bug
 - DBMS bug
 - Hardware failure
 - e.g., disk crash
 - Data sharing
 - e.g., T1 and T2 in parallel

7

7

2. Failures

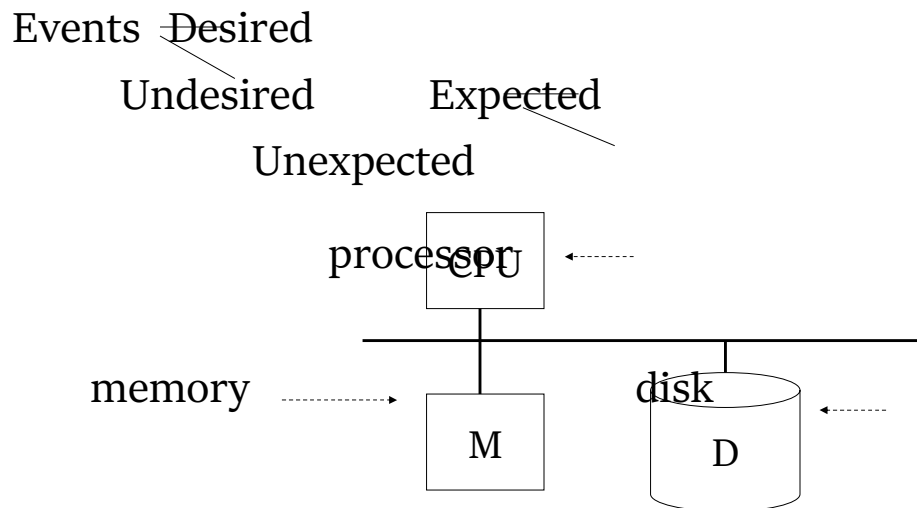
2.1. Classification

2.2. How to do

8

8

2.1. Classification



9

2.2. How to do

Failure → recovery

- Maintaining the consistency of DB by ROLLBACK to the last consistency state.
 - Ensuring 2 properties
 - Atomic
 - Durability
- > Using LOG

10

3. Transaction Log

- 3.1. Log record
- 3.2. Undo logging
- 3.3. Redo logging
- 3.4. Discussion

11

11

3.1. Log record

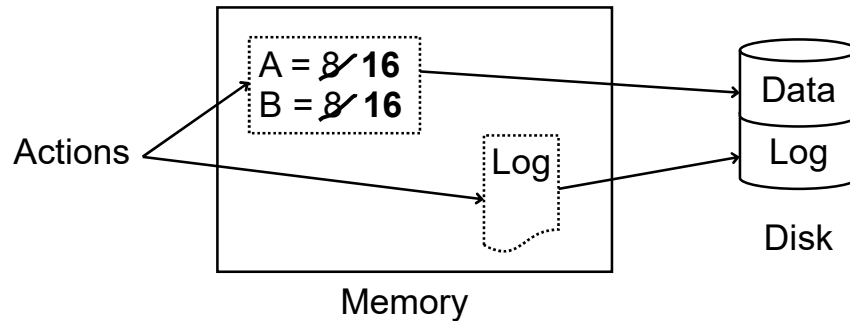
- A sequence of log record keeping trace of actions executed by DBMS
 - <start T>
 - Log the beginning of the transaction execution
 - <commit T>
 - Transaction is already finished
 - <abort T>
 - Transaction is canceled
 - <T, X, v, w>
 - Transaction makes an update action, before update $X=v$, after update $x = w$

12

12

3.1. Log record

- Handled in main memory and put to external memory (disk) when possible



13

13

3.2. Undo logging

Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
1							<start T>
2	Read(A,t)	8	8		8	8	
3	t:=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8>
5	Read(B,t)	8	16	8	8	8	
6	t:=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8>
8	Flush log						
9	Output(A)	16	16	16	16	8	
10	Output(B)	16	16	16	16	16	
11							<commit T>
12	Flush log						

14

14

3.2. Undo logging

- Undo-Logging Rules

- For every action generate undo log record (containing old value)
- Before X is modified on disk, log records pertaining to X must be on disk (write ahead logging: WAL)
- Before commit is flushed to log, all writes of transaction must be reflected on disk

15

15

3.2. Undo logging: Example

Read(A)

If A > 50 then display("so du hop le")

Else {

 A:=A+50

 =====→CRASH

 display ("ghi no tai khoan A")

}

16

16

3.2. Undo logging: Recovery Rules

- Let S is set of unfinished transactions
 - $\langle \text{start } T_i \rangle$ in log
 - $\langle \text{commit } T_i \rangle$ or $\langle \text{abort } T_i \rangle$ is not in log
- For each $\langle T_i, X, v \rangle$ in log
 - If $T_i \in S$ then - Write(X, v)
- Output(X)
- For each $T_i \in S$
 - Write $\langle \text{abort } T_i \rangle$ to log

17

17

3.3. Redo logging

Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
1							$\langle \text{start } T \rangle$
2	Read(A,t)	8	8		8	8	
3	$t := t * 2$	16	8		8	8	
4	Write(A,t)	16	16		8	8	$\langle T, A, 16 \rangle$
5	Read(B,t)	8	16	8	8	8	
6	$t := t * 2$	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	$\langle T, B, 16 \rangle$
8							$\langle \text{commit } T \rangle$
9	Flush log						
10	Output(A)	16	16	16	16	8	
11	Output(B)	16	16	16	16	16	$\langle T, \text{end} \rangle$

18

18

3.3. Redo logging: Rules

1. For every action, generate redo log record (containing new value)
2. Before X is modified on disk (DB), all log records for transaction that modified X (including commit) must be on disk
3. Flush log at commit
4. Write END record after DB updates flushed to disk

19

19

3.3. Redo logging: Recovery Rules

- Let S = set of transactions with
 - $\langle T_i, \text{commit} \rangle$ in log
 - no $\langle T_i, \text{end} \rangle$ in log
- For each $\langle T_i, X, v \rangle$ in log, in forward order (earliest \rightarrow latest)
 - If $T_i \in S$ then write(X, v)
output(X)
- For each $T_i \in S$
 - write $\langle T_i, \text{end} \rangle$

20

20

3.4. Discussion

- Undo Logging
 - need to write to disk as soon transaction finishes
 - -> Access disk
- Redo Logging
 - need to keep all modified blocks in memory until commit
 - -> Use memory

21

21

3. Transaction Log

	Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
Undo/	1							<start T>
Redo	2	Read(A,t)	8	8		8	8	
logging	3	t:=t*2	16	8		8	8	
	4	Write(A,t)	16	16		8	8	<T, A, 8, 16>
	5	Read(B,t)	8	16	8	8	8	
	6	t:=t*2	16	16	8	8	8	
	7	Write(B,t)	16	16	16	8	8	<T, B, 8, 16>
	8	Flush log						
	9	Output(A)	16	16	16	16	8	
	10							<commit T>
	11	Output(B)	16	16	16	16	16	

22

22

4. Checkpoint

- 4.1. Purpose
- 4.2. Checkpoint for Undo logging
- 4.3. Checkpoint for Redo logging
- 4.4 Checkpoint for Undo/Redo logging

23

23

4.1. Purpose

- Decreases the amount of time required for data store recovery
- Makes a portion of the transaction log unneeded for any future data store recovery operation

24

24

4.2. Checkpoint for Undo Logging

```

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<T2, C, 15>
<T2, D, 20>
<commit T1>
<commit T2>
<checkpoint>
<start T3>
<T3, E, 25>
<T3, F, 30>

```

scan

```

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<start ckpt (T1, T2)>
<T2, C, 15>
<start T3>
<T1, D, 20>
<commit T1>
<T3, E, 25>
<commit T2>
<end ckpt>
<T3, F, 30>

```

scan

25

25

4.3. Checkpoint for Redo Logging

```

<start T1> <T1, A, 5> <start T2>
<commit T1> <T2, B, 10>
<T2, C, 10>
<start ckpt (T2)>
<T2, C, 15>
<start T3> <T3, D, 20>

```

scan

```

<start T1> <T1, A, 5> <start T2>
<commit T1> <T2, B, 10> <start ckpt (T2)>
<T2, C, 15> <start T3>
<T3, D, 20>
<end ckpt>
<commit T2>
<commit T3>

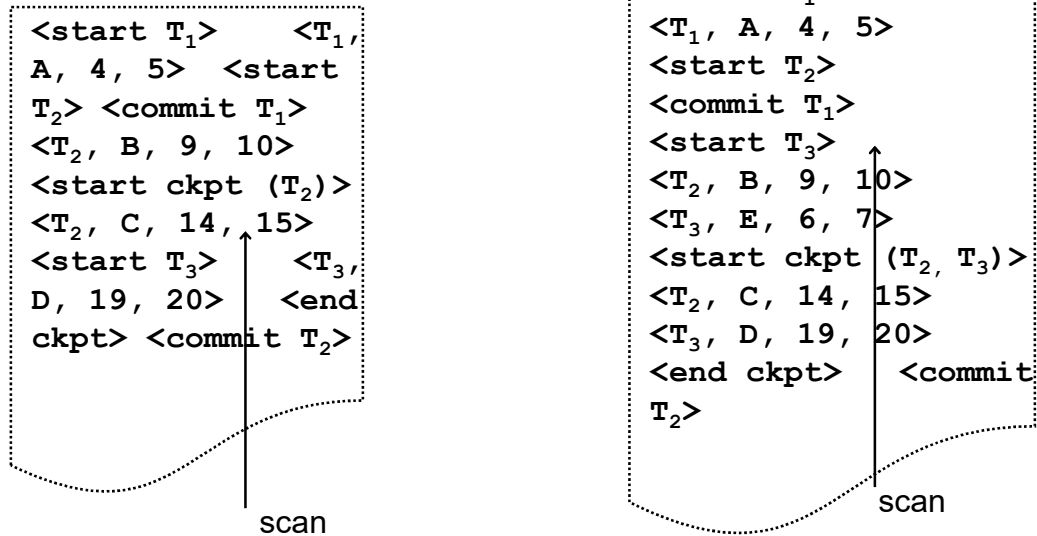
```

scan

26

26

4.4. Checkpoint for Undo/Redo Logging



27

27

Summary

- Transaction
 - Sequence of actions
- Recovery
 - Maintaining the consistency of DB by ROLLBACK to the last consistency state.
- Logging
 - Sequence of record keeping trace of actions executed by DBMS
- Checkpoint
 - Provides a more up-to-date data store image on which recovery can begin

28

28