HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Lesson 11
Transaction Management

Recovery

Learning objectives

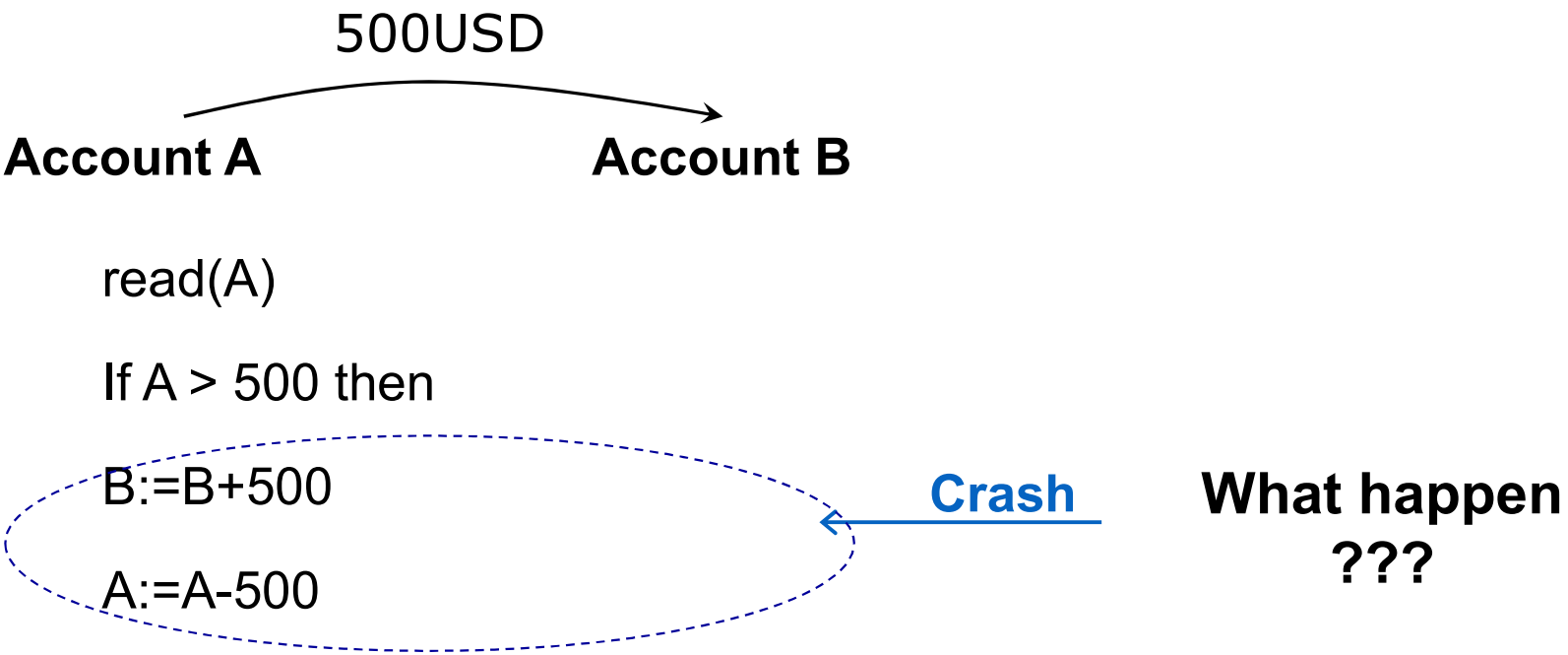
- ***Upon completion of this lesson, students will be able to:***

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

Outline

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

Example


500USD
Account A
Account B
read(A)
If A > 500 then
B:=B+500
A:=A-500
Crash
What happen
???

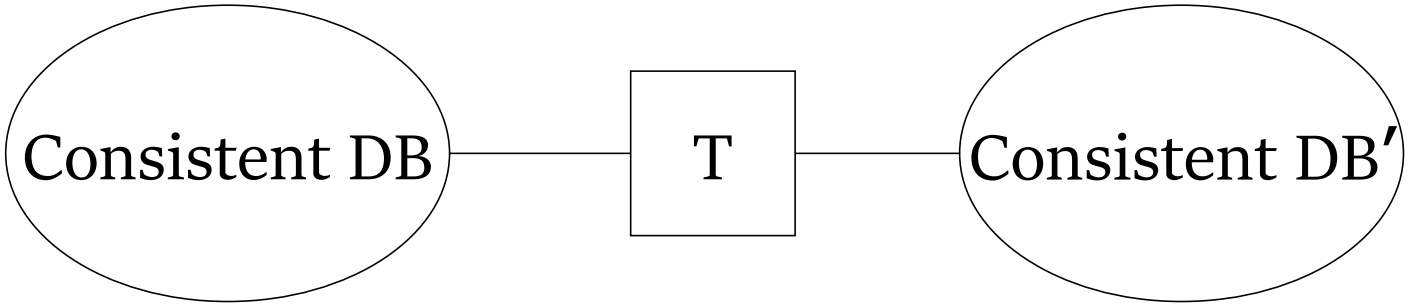
1. Transaction & Recovery

1.1. Objective

1.2. Problems

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

1.2. Problems

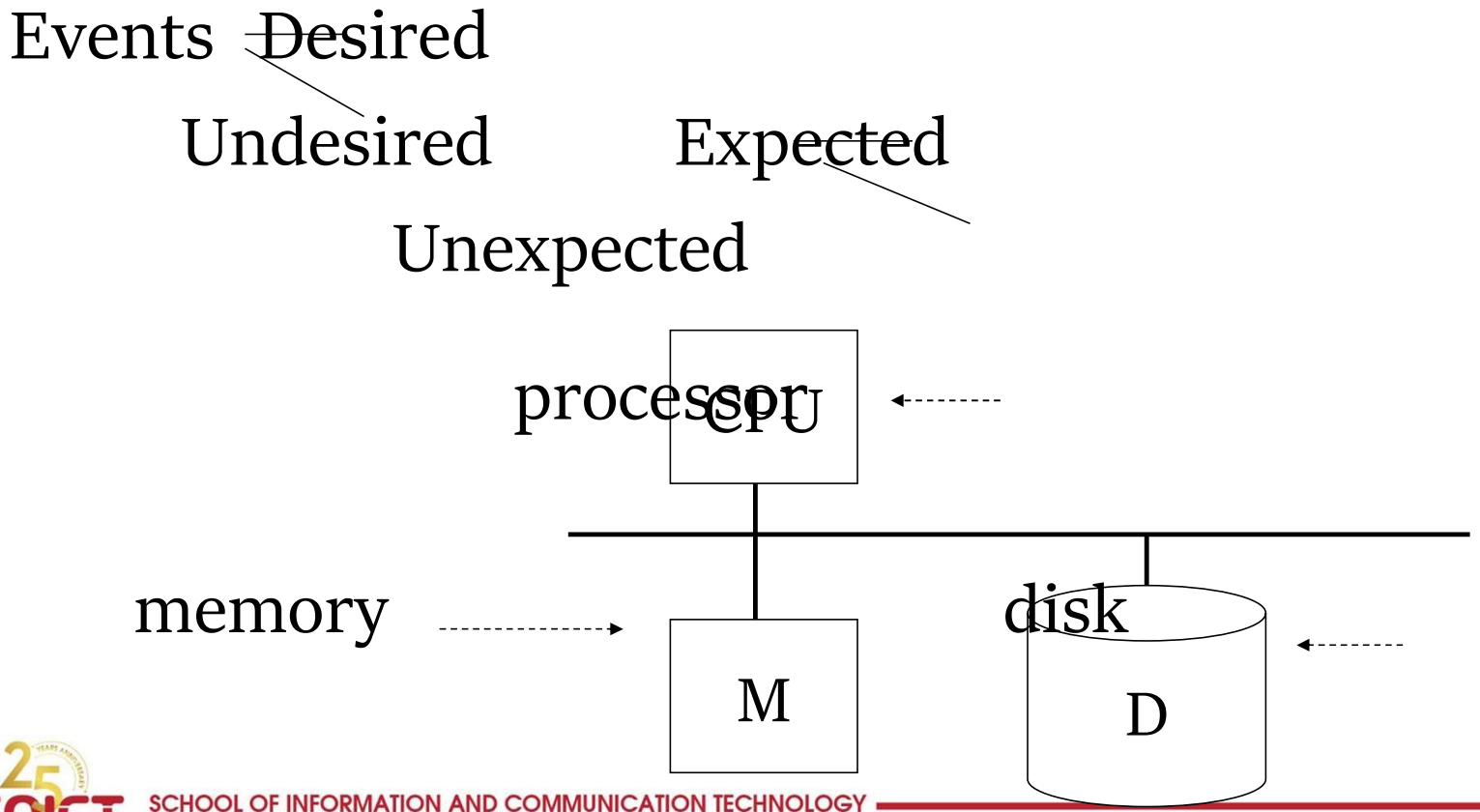
- Constraint violation?
  - Transaction bug
  - DBMS bug
  - Hardware failure
    - e.g., disk crash
  - Data sharing
    - e.g., T1 and T2 in parallel

2. Failures

2.1. Classification

2.2. How to do

2.1. Classification



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

3. Transaction Log

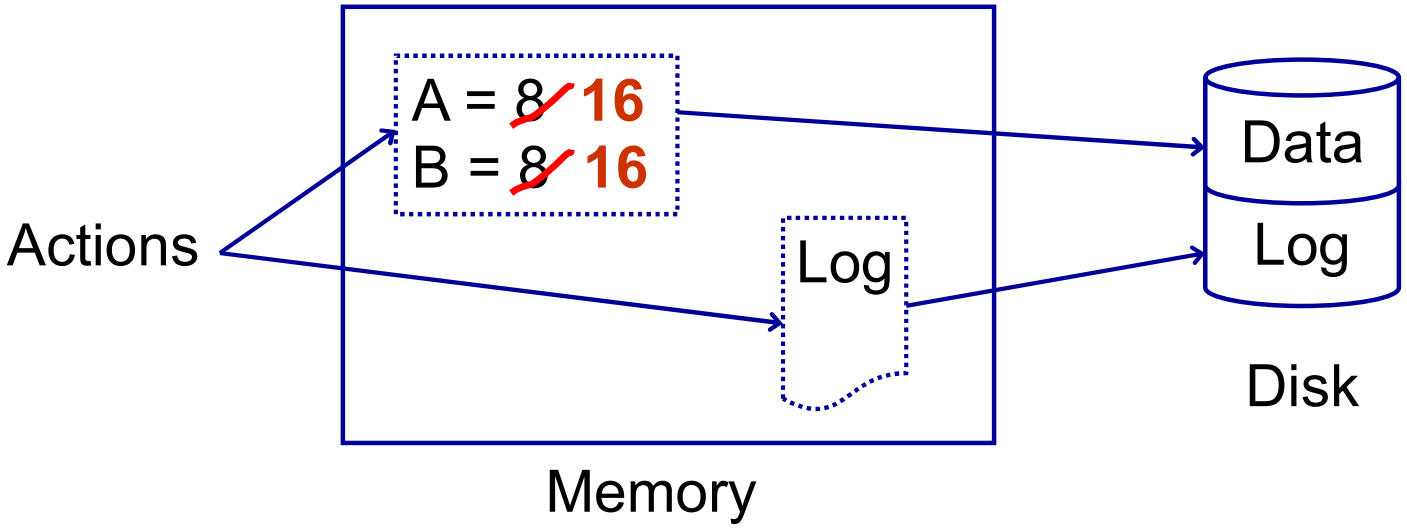
3.1. Log record
3.2. Undo logging
3.3. Redo logging
3.4. Discussion

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

3.1. Log record

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



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**						

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

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”)

}

3.2. Undo logging: Recovery Rules

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

3.3. Redo 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, 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, 16>
8							<commit T>
9	**Flush log**						
10	Output(A)	16	16	16	16	8	
11	Output(B)	16	16	16	16	16	
							<T, end>

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

3.3. Redo logging: Recovery Rules

- Let S = set of transactions with
  - <Ti, commit> in log
  - no <Ti, end> in log
- For each <Ti, X, v> in log, in forward order (earliest $\rightarrow$ latest)
  - If Ti $\in$ S then write(X, v)
    output(X)
- For each Ti $\in$ S
  - write <Ti, end>

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

3. Transaction Log

Undo/ Redo 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, 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	

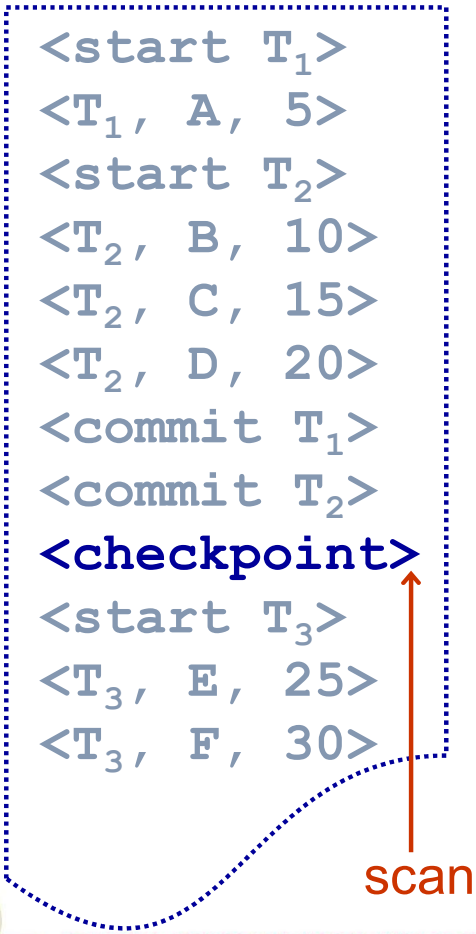
4. Checkpoint

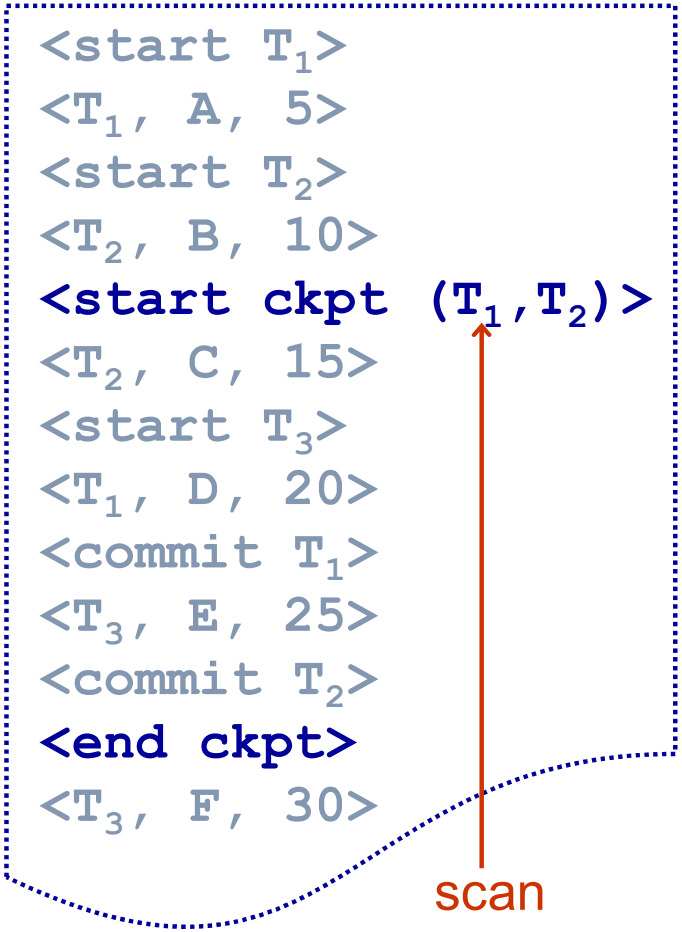
4.1. Purpose
4.2. Checkpoint for Undo logging
4.3. Checkpoint for Redo logging
4.4 Checkpoint for Undo/Redo logging

4.1. Purpose

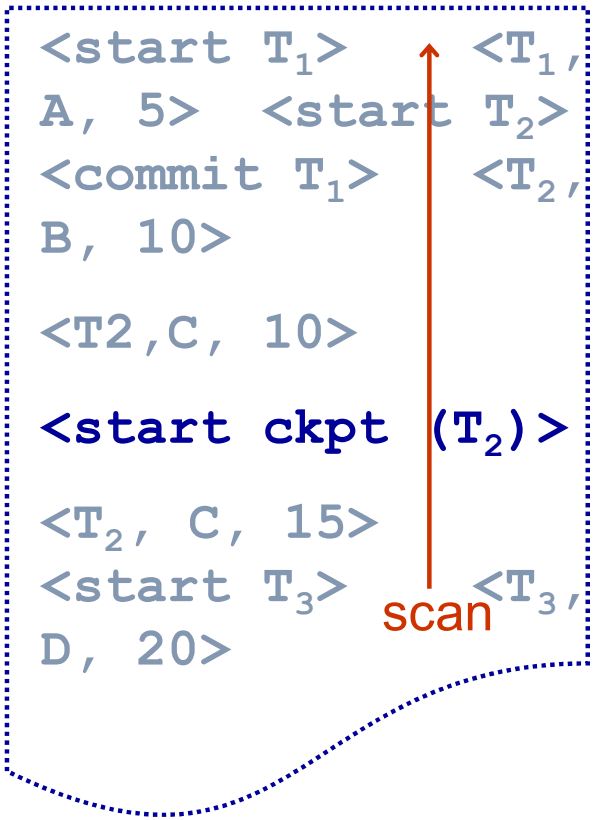
- Decreases the amount of time required for data store recovery
- Makes a portion of the transaction log unneeded for any future data store recovery operation

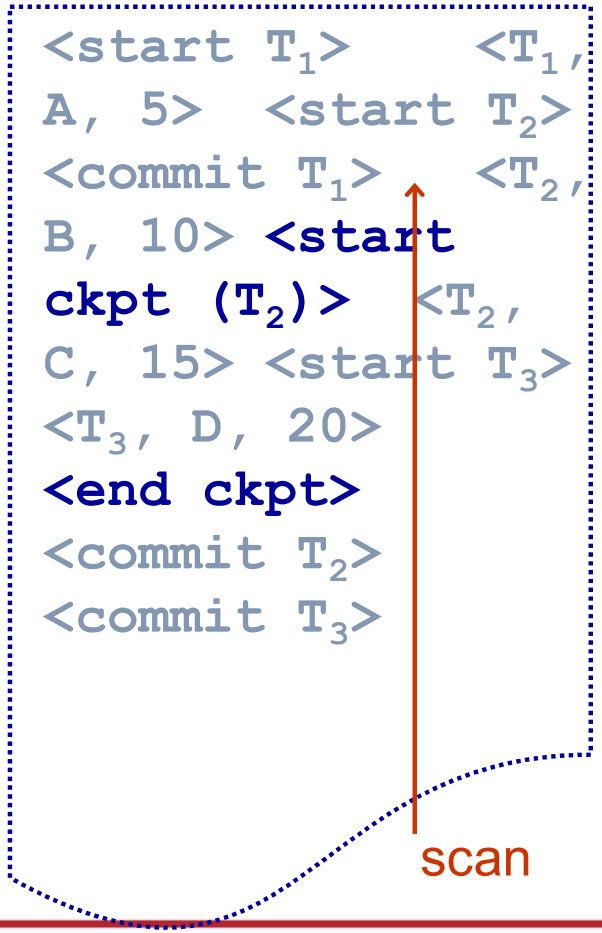
4.2. Checkpoint for Undo Logging



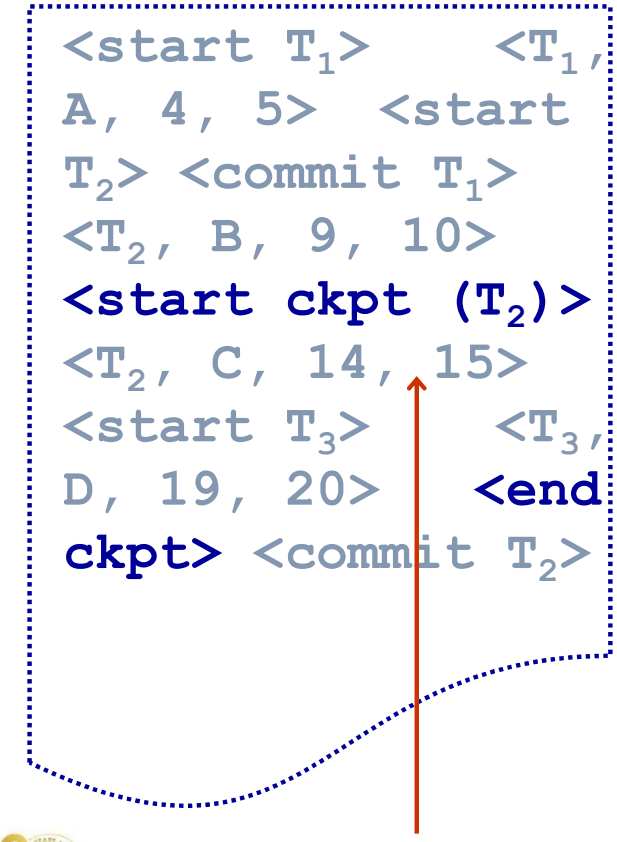


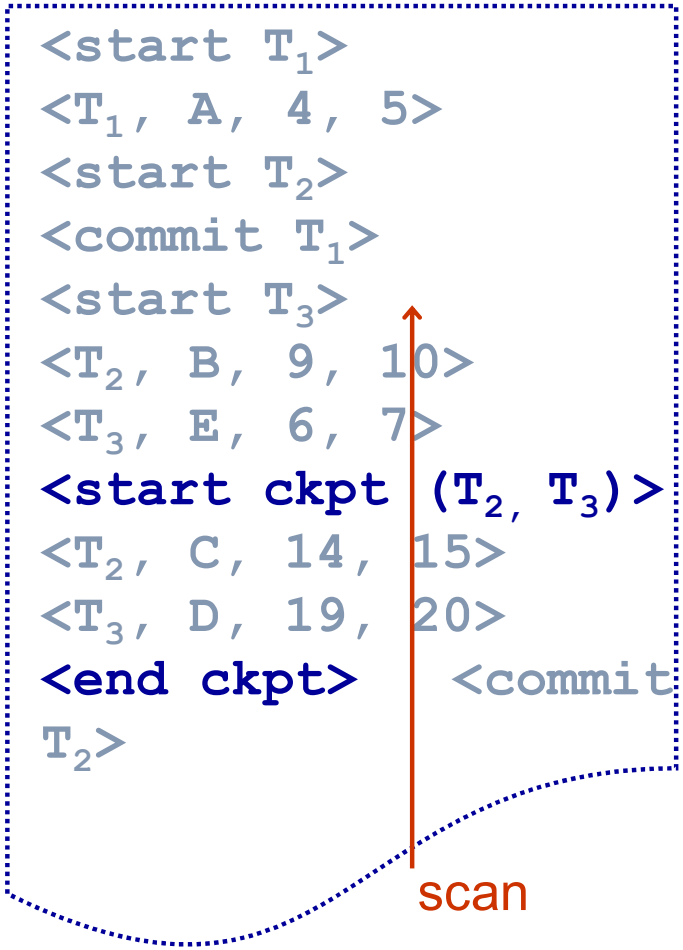
4.3. Checkpoint for Redo Logging





4.4. Checkpoint for Undo/Redo Logging





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

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict