4.5.2 Transaction

A transaction T is a finite sequence of actions on DB exhibiting the following effects:

- Atomic action: Nothing or All.
- Consistency preservation: consistency state of DB → another consistency state of DB.
- Isolation: concurrent transactions should run as if they are independent each other.
- Durability:The effects of a successfully completed transaction are permanently reflected in DB and recoverable even failure occurs later.

Example: transfer money s from account A to account B

Begin transaction

read A

A:=A-s

if A<0 then Display "insufficient fund"

Rollback /*undo and terminate */

else B:=B+s

Display "transfer complete"

Commit /*commit the update and terminate */

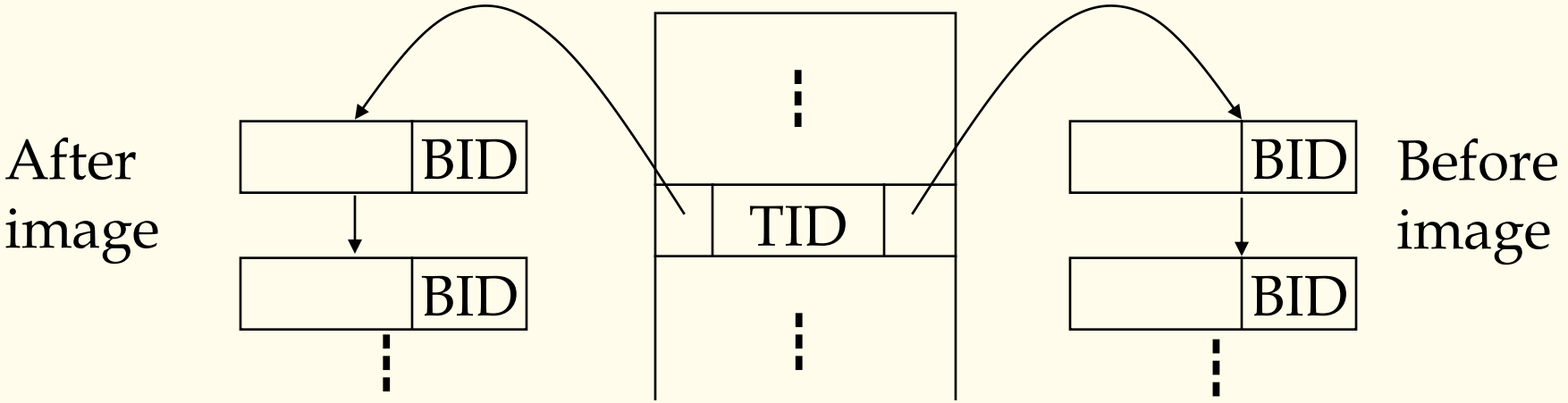
Rollback --- abnormal termination. (Nothing)

Commit --- normal termination. (All)

4.5.3 Some Structures to Support Recovery

Recovery information (such as Log) should be stored in nonvolatile storage. The following information need to be stored in order to support recovery:

1) Commit list : list of TID which have been committed.
2) Active list : list of TID which is in progress.
3) Log :



4.5.4 Commit Rule and Log Ahead Rule

1) Commit Rule

A.I must be written to nonvolatile storage before commit of the transaction.

2) Log Ahead Rule

If A.I is written to DB before commit then B.I must first written to log.

3) Recovery strategies

(1) The features of undo and redo (are idempotent) :

undo(undo(undo --- undo(x) ---)) = undo(x)

redo(redo(redo --- redo(x) ---)) = redo(x)