4. Database Management Systems （3/4）

4.5 Recovery

4.5.1 Introduction

The main roles of recovery mechanism in DBMS are:

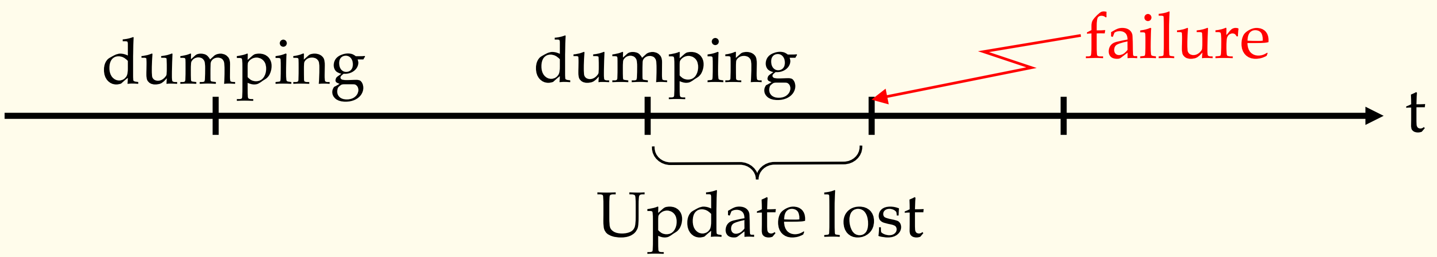
(1) Reducing the likelihood of failures (prevention)

(2) Recover from failures (solving)

Restore DB to a consistent state after some failures.

- Redundancy is necessary.
- Should inspect all possible failures.
- General method:

1) Periodical dumping

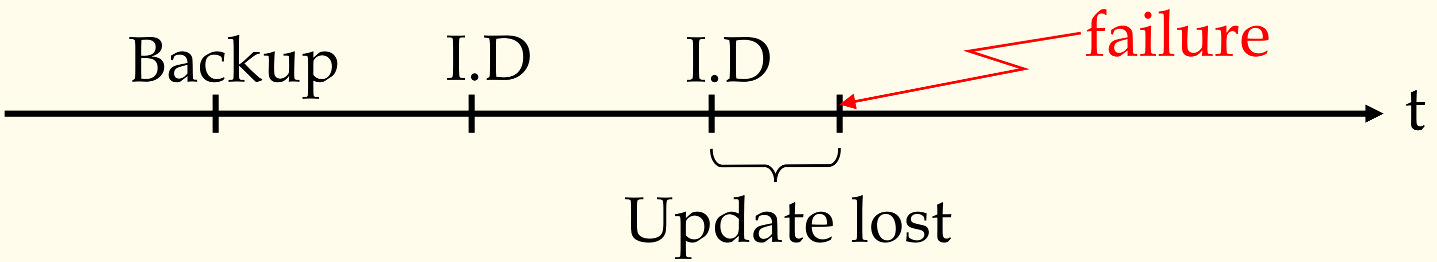


dumping dumping failure t

Update lost

- Variation : Backup + Incremental dumping

  I.D --- updated parts of DB



Backup I.D I.D failure t

Update lost

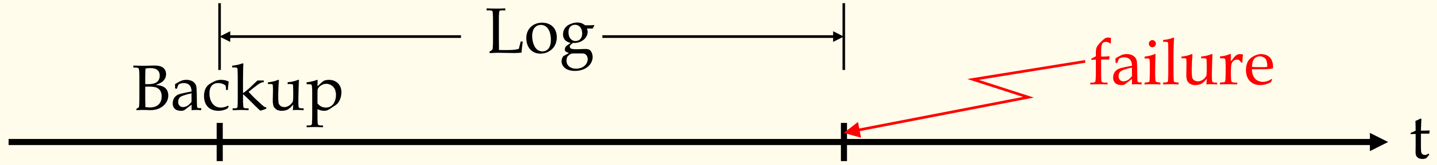
This method is easy to be implemented and the overhead is low, but the update maybe lost after failure occurring. So it is often used in file system or small DBMS.

2) Backup + Log

Log : record of all changes on DB since the last backup copy was made.

Change: { Old value (before image --- B.I)
New value (after image --- A.I) } Recorded into Log

For	update op. :	B.I	A.I
	insert op. :	----	A.I
	delete op. :	B.I	----



While recovering:

- Some transactions maybe half done, should undo them with B.I recorded in Log.
- Some transactions have finished but the results have not been written into DB in time, should redo them with A.I recorded in Log. (finish writing into DB)

It is possible to recover DB to the most recent consistent state with Log.

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 */
```

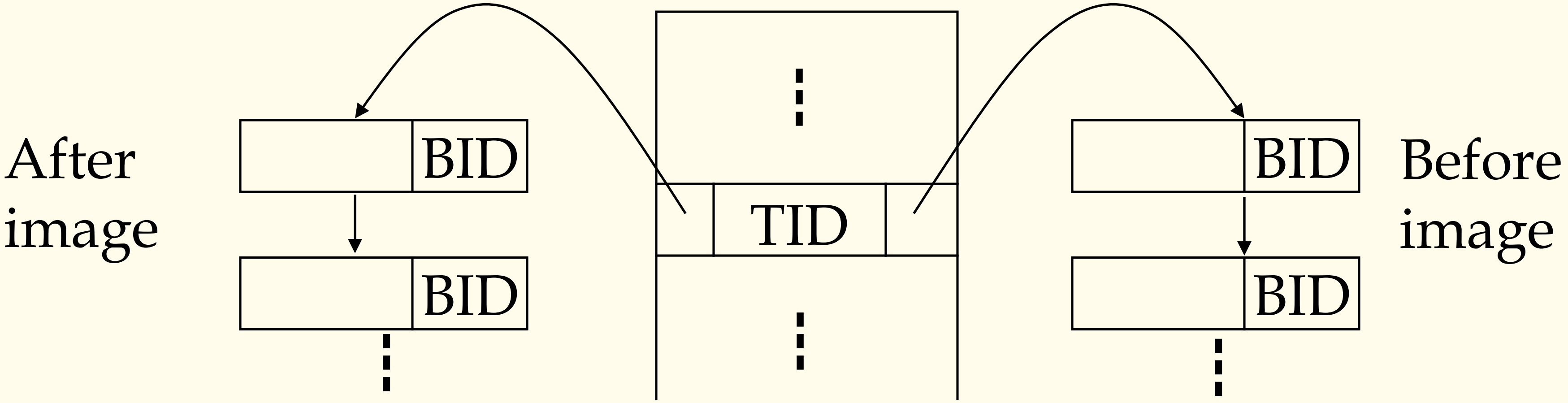
Rollback --- abnormal termination. (Nothing)

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) :

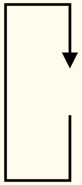
undo(undo(undo --- undo(x) ---)) = undo(x)

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

(2) Three kinds of update strategy

a) A.I→DB before commit

TID →active list



B.I →Log (Log Ahead Rule)

A.I →DB

⋮

commit

TID →commit list

delete TID from active list

The recovery after failure in this situation

Check two lists for every TID while restarting after failure:

Commit list	Active list	
	✓	Undo, delete TID from active list
✓	✓	delete TID from active list
✓		nothing to do

b) A.I→DB after commit

TID →active list

A.I →Log (Commit Rule)

⋮

TID →commit list

commit

A.I →DB

delete TID from active list

The recovery after failure in this situation

Check two lists for every TID while restarting after failure:

Commit list	Active list	
	✓	delete TID from active list
✓	✓	redo, delete TID from active list
✓		nothing to do

c) A.I→DB concurrently with commit

TID →active list

A.I, B.I →Log (Two Rules)

A.I →DB (partially done)

⋮

TID →commit list

commit

A.I →DB (completed)

delete TID from active list

The recovery after failure in this situation

Check two lists for every TID while restarting after failure:

Commit list	Active list	
	✓	Undo, delete TID from active list
✓	✓	redo, delete TID from active list
✓		nothing to do

Conclusion :

	redo	undo
a)	✘	✓
b)	✓	✘
c)	✓	✓
d)	✘	✘

?