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)	✘	✘

4.6 Concurrency Control

4.6.1 Introduction

In multi users DBMS, permit multi transaction access the database concurrently.

- Why concurrency?

1) Improving system utilization & response time.
2) Different transaction may access to different parts of database.

- Problems arise from concurrent executions

	T_1	T_2	T_1	T_2	T_1	T_2
	Read(x)			Read(t[x])	Read(x)	
		Read(x)	Write(t)			Write(x)
	x:=x+1					
	Write(x)	x:=2*x		Read(t[y])	Read(x)	
t		Write(x)	(rollback)			

Lost update **Dirty read** **Unrepeatable read**

So there maybe three kinds of conflict when transactions execute concurrently. They are write – write, write – read, and read – write conflicts. Write – write conflict must be avoided anytime. Write – read and read – write conflicts should be avoided generally, but they are endurable in some applications.