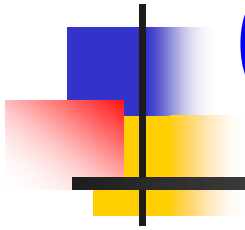


Chapter 13

회복 시스템
(Recovery System)

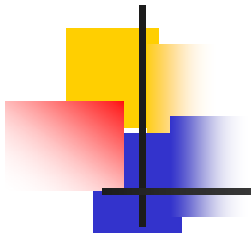
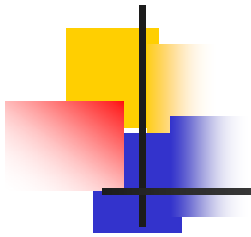


Table of Contents



1. Failure Classification

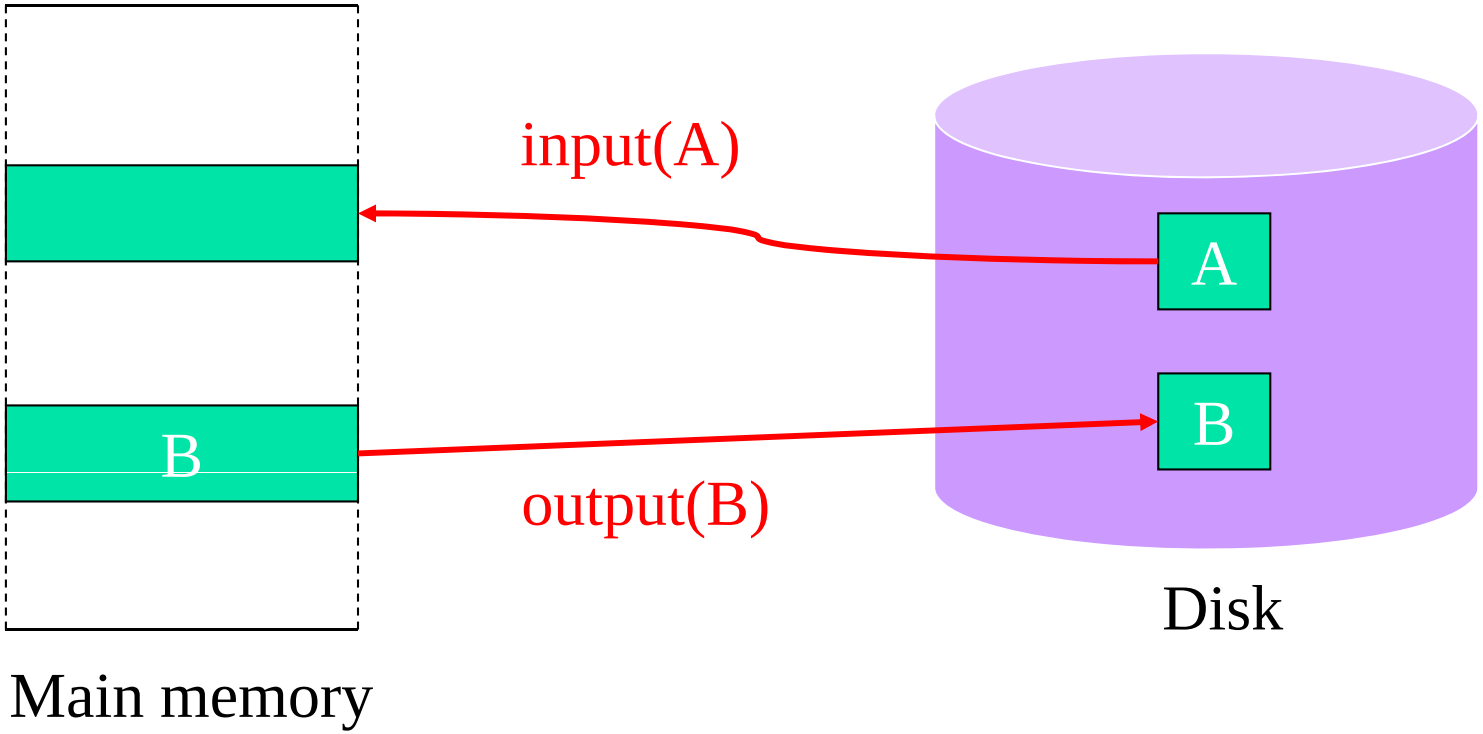
- Transaction Failure
  - Logical Error: bad input, overflow, data not found ...
  - System Error: deadlock
- System Crash
  - DBMS나 OS의 실행 중지 (예: 정전)
  - Volatile memory의 내용 파손
- Media Failure
  - Nonvolatile memory의 내용 파손

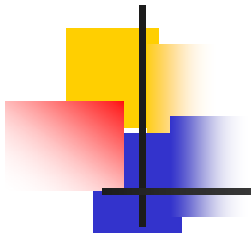
2. Storage Structure

- Storage Types
  - Volatile storage
    - 정전과 같은 system crash 발생시 내용 파손
    - 예: Main memory, Cache memory
  - Nonvolatile storage
    - System crash가 발생해도 내용 유지
    - 예: Hard disk, Magnetic tape
  - Stable storage
    - 어떤 형태의 failure가 발생해도 내용 유지
    - Replication에 의해 S/W적으로 구현

Data Access

- Input(B)와 Output(B)





Data Access (Cont'd)

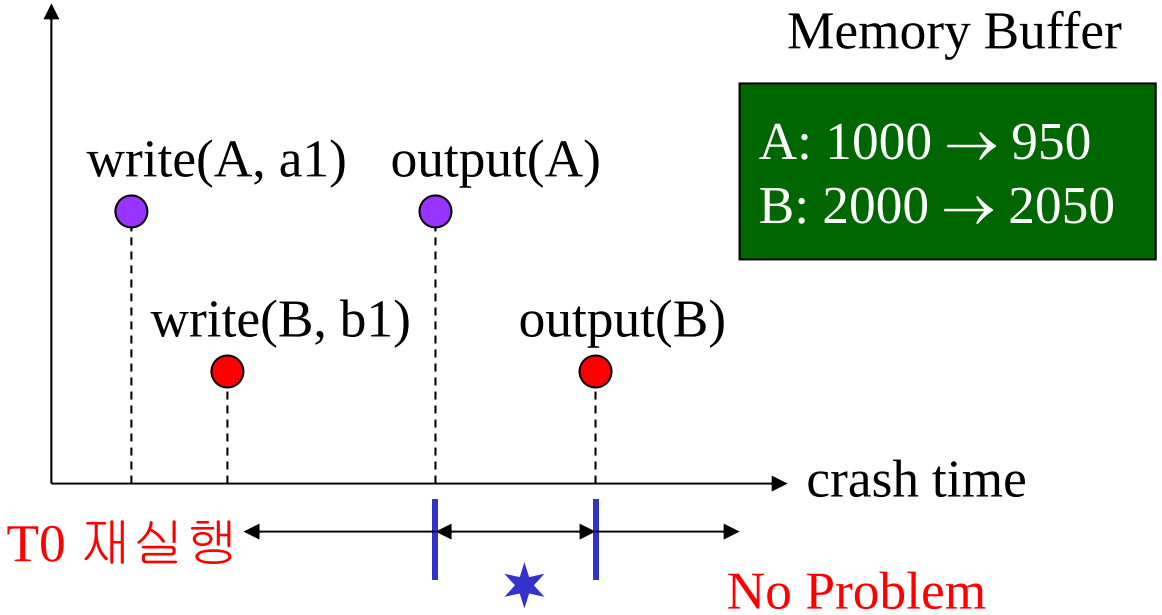
- Read(X)와 Write(X)
  - Read(X, x_i): X를 읽어서 지역 변수 x_i에 할당
    - X를 저장한 버퍼 블록 B_X가 없을 경우, input(B_X)
    - B_X 블록에서 X를 읽어 지역 변수 x_i에 할당
  - Write(X , x_i): 지역 변수 x_i의 값을 X에 할당
    - X를 저장한 버퍼 블록 B_X가 없을 경우, input(B_X)
    - x_i의 값을 B_X 블록의 X에 할당
  - 주의: Output ≠ Write
    - System crash 발생시 recovery 필요
    - Synchronous IO의 문제점?

3. Recovery and Atomicity

- ✶ 에서의 Recovery Action
  - T0 재실행: A is 900, not 950
  - T0 재실행 않음: A = 950, B = 2000

Example
Transaction: T0

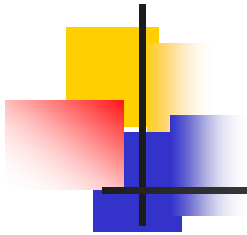
read(A, a1)
a1 = a1 – 50
write(A, a1)
read(B, b1)
b1 = b1 + 50
write(B, b1)



Buffer Policy vs. Recovery Operations

- Buffer Policy
  - 실행 중인 트랜잭션의 결과가 디스크에 기록 가능?
    - STEAL, ~STEAL
  - 완료한 트랜잭션의 결과를 반드시 디스크에 기록?
    - FORCE, ~FORCE
- Recovery Operation (고장 발생 후)
  - UNDO: 실행 중인 트랜잭션의 결과를 디스크에서 삭제
  - REDO: 이전에 완료한 트랜잭션을 재 실행

	STEAL	~STEAL
FORCE	UNDO	No Recovery
~FORCE	UNDO, REDO	REDO



4. Log-Based Recovery

- Update Log Record
  - Transaction이 update 연산 실행할 때마다 생성
  - 구조
    - Transaction 식별자, Data Item 식별자
    - Old value, New Value (로그 성격에 따라 optional)

- Log Record의 종류
  - $<T_i$ start$>$
  - $<T_i, X_j, V_1, V_2>$
  - $<T_i$ commit$>$
  - $<T_i$ abort$>$

Deferred DB Modification

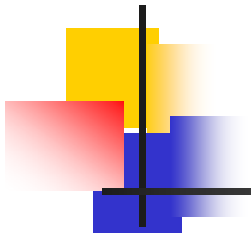
- 특성
  - 트랜잭션 완료 전에 갱신 내용을 DB 반영 않음
    - ~STEAL
  - UNDO 불필요 - Log에 old value 기록할 필요 없음

<T0 start> <T0, A, 950> <T0, B, 2050>	<T0 start> <T0, A, 950> <T0, B, 2050> <T0 commit> <T1 start> <T1, C, 600>	<T0 start> <T0, A, 950> <T0, B, 2050> <T0 commit> <T1 start> <T1, C, 600> <T1, commit>
No recovery	REDO(T0)	REDO(T0) REDO(T1)

Immediate DB Modification

- 특성
  - 트랜잭션 완료 전에 갱신 내용을 DB 반영할 수 있음
    - STEAL
  - UNDO와 REDO 모두 필요 - Log에 old value 기록

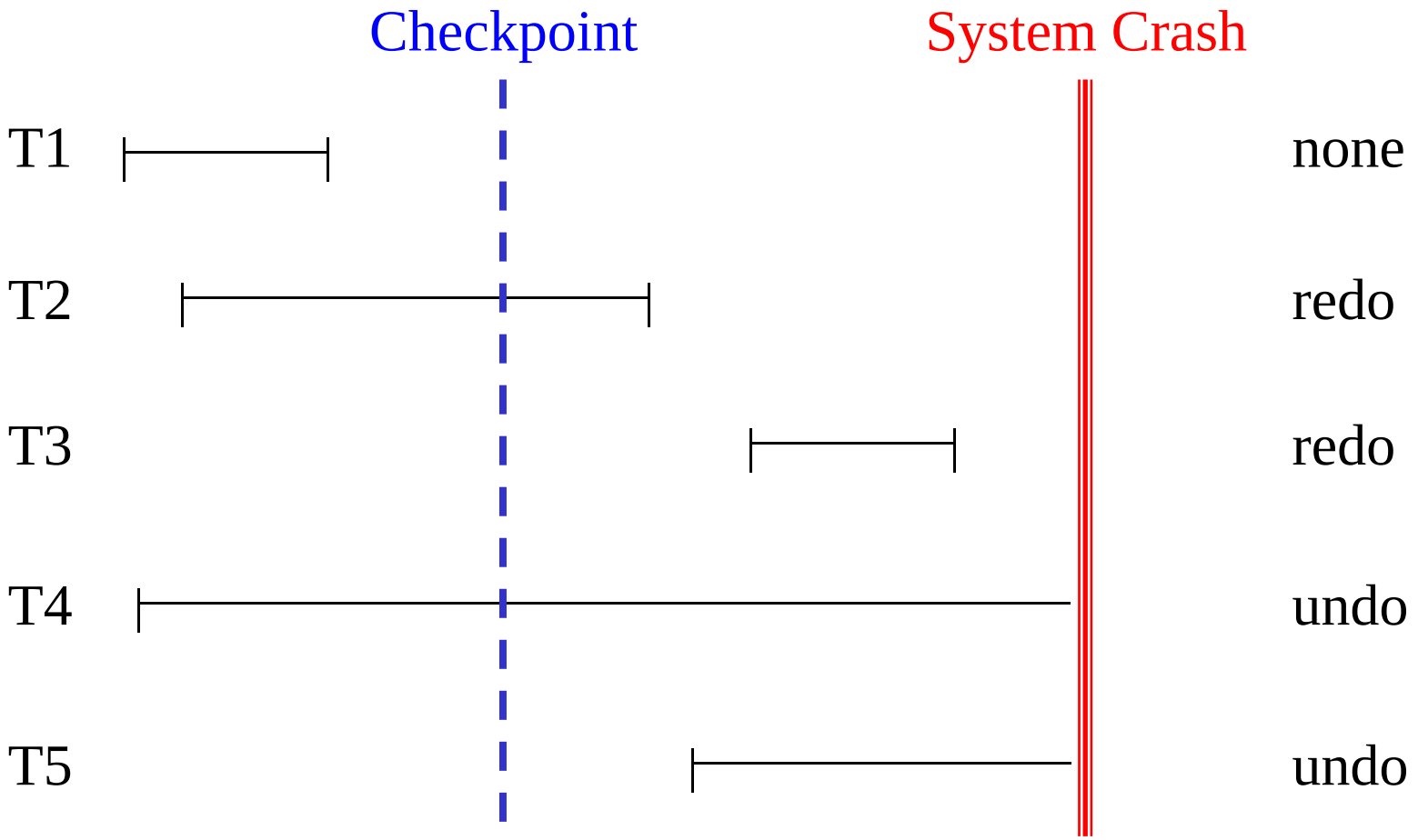
<T0 start> <T0, A, 1000, 950> <T0, B, 2000, 2050>	<T0 start> <T0, A, 1000, 950> <T0, B, 2000, 2050> <T0 commit> <T1 start> <T1, C, 700, 600>	<T0 start> <T0, A, 1000, 950> <T0, B, 2000, 2050> <T0 commit> <T1 start> <T1, C, 700, 600> <T1, commit>
UNDO(T0)	REDO(T0), UNDO(T1)	REDO(T0), REDO(T1)



Checkpoints

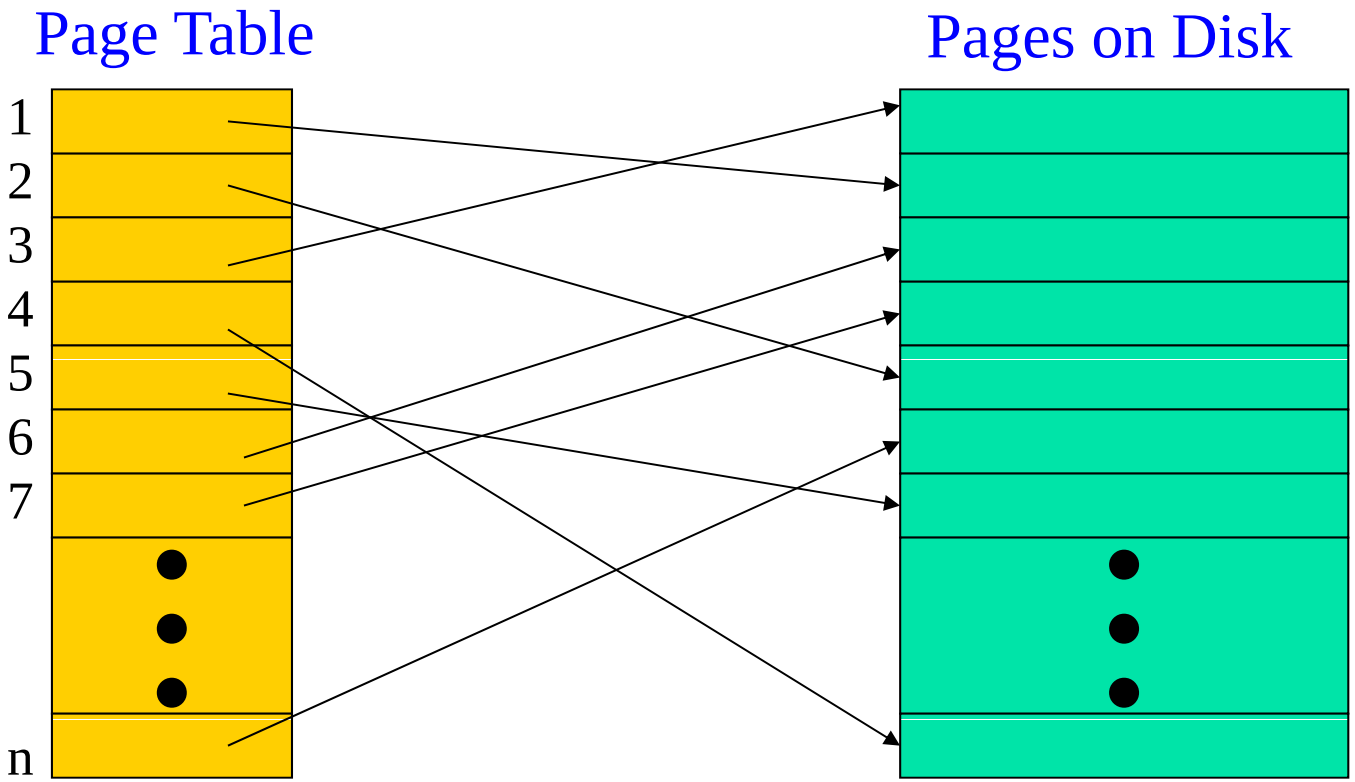
- Problems at System Restart
  - 어떤 로그 레코드부터 조사할 것인가?
  - 대부분의 트랜잭션 실행 결과는 DB에 이미 반영
- Checkpoint Steps
  - 로그 버퍼의 모든 로그 레코드들을 로그 디스크에 기록
  - 갱신된 데이터 버퍼 페이지들을 디스크에 기록
  - <checkpoint> 로그 레코드를 로그 디스크에 기록
- 주의
  - Checkpoint동안에 일반 트랜잭션의 실행 중단?

Checkpoint and System Restart



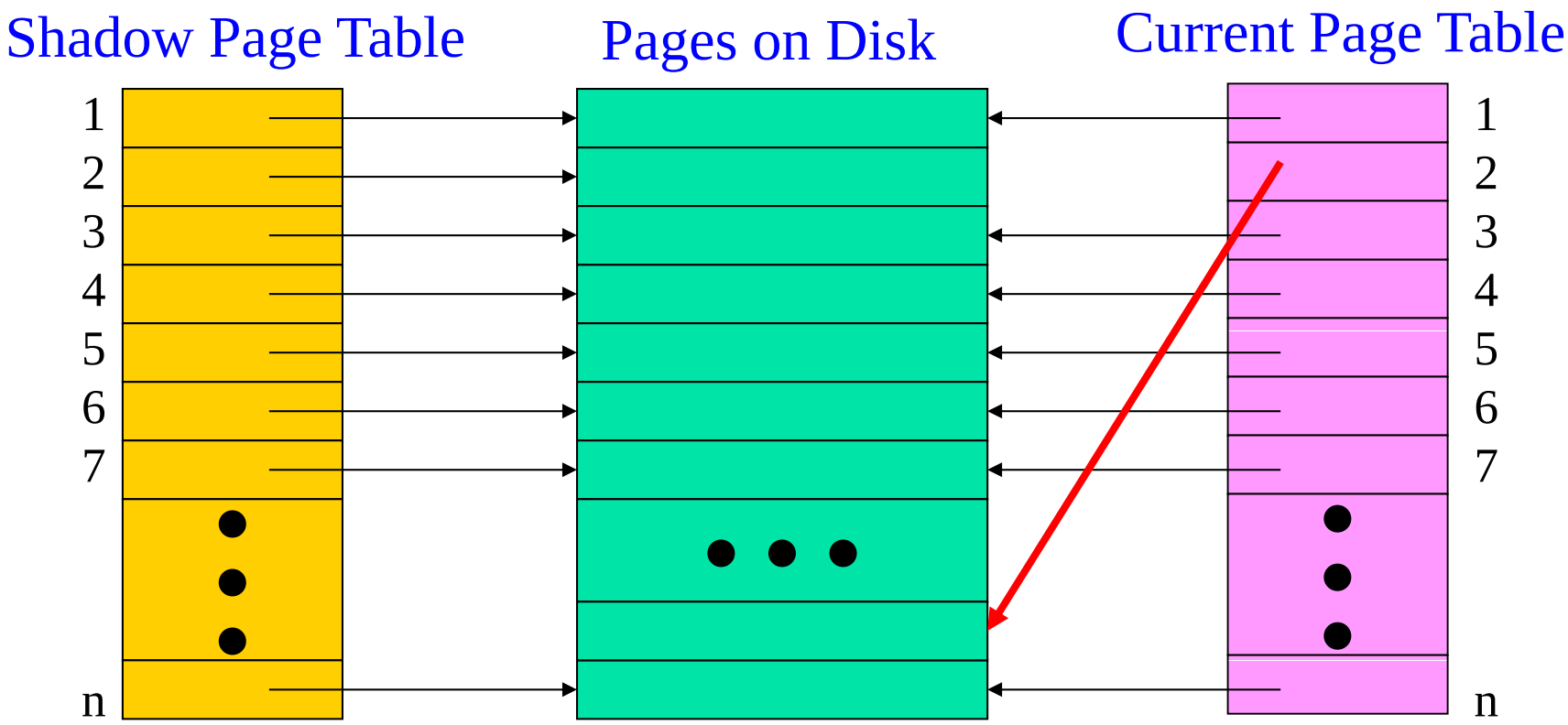
5. Shadow Paging

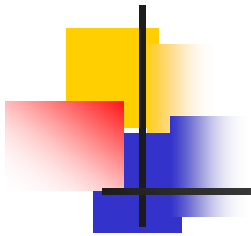
- Paging
  - A database is stored in a set of pages.
  - Page table maintains the pointer of each page.



Shadow Paging의 개념

- Shadow Page Table vs. Current Page Table
  - 트랜잭션의 실행 결과는 Current Page Table에 반영
  - 트랜잭션 완료: Current Page Table이 디스크에 저장



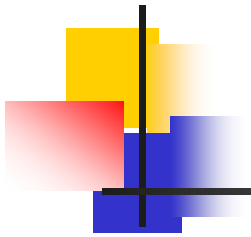


Shadow Paging의 개념 (Cont'd)

- Commit Procedure of a Transaction
  - 갱신된 모든 데이터 페이지들을 디스크에 기록
  - Current Page Table을 디스크에 기록
  - Current Page Table을 Shadow Page Table로 등록
    - Page Table을 가리키는 포인터 변경
    - Atomic Operation

Shadow Paging vs. Logging

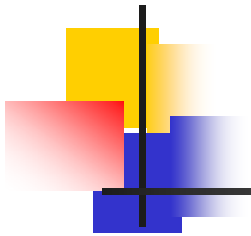
- Shadow Paging의 장점
  - Faster recovery - REDO와 UNDO 불필요
- Shadow Paging의 단점
  - Data Fragmentation
  - Garbage Collection 필요
  - Fine Granularity Locking 지원 곤란
  - Overhead of Normal Transaction Processing
- 대부분의 Commercial DBMS는 Logging 지원



6. ARIES

- A Transaction Recovery Method
  - Fine-Granularity Locking 지원
  - Partial Rollback 지원
  - Write Ahead Logging 이용

- ARIES의 기본 개념
  - Log Sequence Number (LSN) 이용
  - Index에 대한 Logical Undo 지원
  - Flexible Buffer Management: Steal & Not Force
  - Fuzzy Checkpoint 지원



Data Structures of ARIES

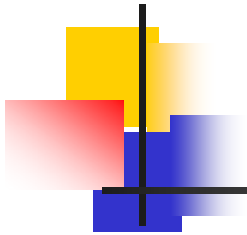
- Log Record
  - Log Sequence Number (LSN)
  - Type: Update, Commit, Compensation, ...
  - Transaction Identifier
  - Previous LSN (PrevLSN)
  - Record Identifier
  - Data: Value Logging / Operation Logging

- Data Page
  - Data + PageLSN

Data Structures (Cont'd)

- Transaction Table
  - Transaction Identifier
  - State: Prepared ('P') or Un-prepared ('U')
  - LastLSN
  - UndoNxtLSN

- Dirty Pages Table
  - Page Identifier
  - RecoveryLSN
    - Page가 dirty 상태로 변환될 때의 LSN
    - Page를 회복하기 위한 로그의 starting point



Normal Processing

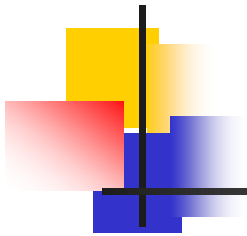
- Page의 record를 update하는 과정
  - Record에 대한 lock 요청 & 승인
  - Page를 buffer에 fix & ‘X’ mode로 latch
  - Record update
  - Log record를 생성하고, in-memory log list에 추가
  - Log의 LSN을 Page의 PageLSN 필드에 저장
  - Page latch 해제 & unfix (fix_count--)

- Transaction Commit
  - ‘commit’ log 생성, log force, lock release
  - Pending list의 개념

Normal Processing (Cont'd)

- Transaction Abort 과정
  - Writing 'rollback' log record
  - Rolling back the transaction to its beginning
  - Discarding the pending actions list
  - Releasing its lock

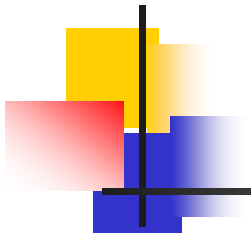
- Fuzzy Checkpoint
  - Begin_Checkpoint log record 기록
  - End_Checkpoint log record 생성
    - Transaction Table, DPT, ...
  - End_Checkpoint log record 기록
  - Master record에 Begin_Checkpoint log의 LSN 저장



Restart Processing

RESTART(Master-Addr)

1. Analysis(Master-Addr, Trans-Table, DPT, RedoLSN)
2. Restart-Redo(RedoLSN, Trans-Table, DPT)
3. Buffer pool DPT = DPT;
4. Restart-Undo(Trans-Table)
5. Require locks for prepared transactions;
6. Checkpoint()
7. RETURN



7. Media Recovery

- Recovery from a disk crash.
- Independent failure mode magnetic tapes 이용
  - Periodic dump of transaction-consistent state of database is much more lengthy than checkpoints.
- Media Recovery: Archival Dump + Redo LOGs