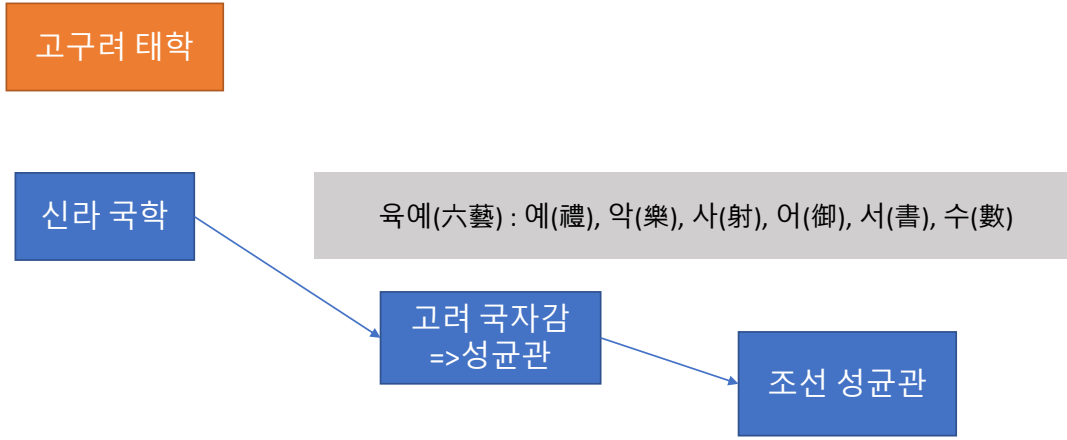
제19장 복구 시스템

Chapter 19 Recovery System

황용사 CE528-1238

목차

- ❖실패의 분류
- ❖저장장치
- ❖복구와 원자성
- ❖복구 알고리즘
- ❖버퍼 관리



실패 분류

- **트랜잭션 실패 Transaction failure:**
  - 논리적 오류: 트랜잭션은 몇몇 내부 오류 조건 때문에 완성할 수 없다.
  - 시스템 오류: 데이터베이스 시스템은 오류 조건 때문에 활동 트랜잭션을 끝내야만 한다. (예, 교착상태 deadlock)
- **시스템 손상 System crash:**
  - 정전 고장 또는 다른 기계적 고장 또는 소프트웨어 고장은 결국 시스템 고장을 유발한다.
  - 실패-정지 가정: 비휘발성 저장소 내용은 시스템 고장으로 파괴되지 않는다고 가정한다
    - ✓ 데이터베이스 시스템은 디스크 자료의 파괴를 방지하기 위하여 상당한 수의 무결성 검사 항목들을 가지고 있다.
- **디스크 고장 Disk failure:**
  - 헤드(읽기/쓰기 장치) 고장 또는 비슷한 디스크 고장은 디스크 저장장치의 모두 또는 부분을 파괴한다.
  - 파괴는 탐지 가능한 것으로 가정한다: 디스크 구동장치는 고장을 탐지하기 위하여 검사 합(checksums)을 사용한다.

알고리즘의 어원

알콰리즈미

- Abū ʿAbdallāh Muḥammad ibn Mūsā al-Khwārizmī[note 1]
  - 아부 압달라 무함마드 이븐 무사 알콰리즈미
  - 일찍이 문자로 표기: Algoritmi 또는 Algaurizin,
  - (CE 780, Khwārizm[2][4][5] – CE 850)은 이라크 수도 박흐닫(Baghdad), 지혜의 집에 있던 페르시안 수학자, 천문학자, 지리학자, 그리고 학자
  - 인도, 그리스 등의 많은 지식 집대성, 그 중에서 인도의 숫자와 숫자 계산 방법 즉, 산수를 아라비아에 가서 가르침.
  - 그것이 유럽으로 전파되어 '아라비아 수' 가 됨.
  - '콰라즘에서 온 사람이 가르쳐 준 수(알 콰라즘, al khowarazm)' 라 하게 됨

아부 압달라 무함마드 이븐 무사

- ❖콰라즘에서 온 사람이 "인도의 수"를 아라비아에서 가르침
- ❖오늘날 "아라비아 수"가 되고 알고리즘이라 이름을 남김

복구 알고리즘 Recovery Algorithms

- 계좌 A 에서 B로 $50를 송금하는 트랜잭션 Ti를 고려하라.
  - 두 가지 갱신: 1) read(A); A=A-50; write(A) 2) read(B); B=B+50; write(B);
- 트랜잭션 T_i : 데이터베이스로 출력될 A와 B 포함 블록의 갱신을 요구한다.
  - 어떤 고장이 이들 수정 중 하나는 완료된 뒤에 그러나 이들 둘 다 만들어지기 전에 일어날 지도 모른다.
  - 트랜잭션은 완료를 보장함이 없이, 데이터베이스를 수정하는 것은 비일관성 상태를 데이터베이스에 남길 수 있다.
  - 데이터베이스를 수정하지 못하는 것은 트랜잭션이 완료한 뒤에 곧 바로 고장이 일어난다면 손실 갱신으로 결론이 날 수 있다.
  - 복구 알고리즘은 두 개의 부분을 가지고 있다:
    1. 충분한 정보를 보장하기 위하여 정상적인 정보처리 동안 취했던 행동들은 실패로부터 복구하기 위하여 존재한다.
    2. 원자성, 일관성, 지속성을 보장하는 어떤 상태로 DB 내용을 복구하기 위하여 실패 뒤 취했던 행동들은 실패로부터 복구하려고 존재한다.

저장 장치

❖휘발성 저장 장치 Volatile storage :
- 시스템 깨어짐에서 자료를 구하지 못한다.
- 예: 주기억 장치, cache memory

❖비휘발성 저장장치 Nonvolatile storage :
- 시스템 깨어짐에서 자료를 구한다.
- 예: disk, tape, flash memory,
  비휘발성 (배터리 여벌) RAM
- 그러나 아직도 자료를 잃어버릴 수 있음

❖안전한 저장장치 Stable storage :
- 모든 실패를 구하는 저장의 신비한 형태
- 별도의 비휘발성 매체상에서 복수의 사본을 유지함으로써 개략화된다.
- 안전한 저장장치를 어떻게 구현할 것인가에 관한 좀더 자세한 것에 대하여 책을 보라.

저장 장치: 안전한 저장 장치 구현

- 별도의 디스크 상에 각 블록의 복수 사본을 유지하라.
  - 사본들은 화재, 홍수와 같은 재해로부터 보호하기 위하여 먼 지역에 있을 수 있다.
- 자료 전송 중 고장은 아직 비일관성 상태로 결론 날 수 있다. 블록 전송은 다음 결과를 낼 수 있다.
  - 성공적 완료 commit: 전송된 블록은 목적지에 안전하게 도착하였다.
  - 부분 실패 partial failure: 전송 중 실패로 목적지 블록은 부정확 정보를 가진다.
  - 완전 실패 abort: 전송 초기 실패로 목적지 블록은 결코 갱신되지 않은 체 있다.
- 자료 전송 중 고장이 날 때 저장소 매체를 보호하는 것으로 일관성 상태로 되돌려져야 한다.(두 거울(mirror) 디스크의 경우 각 블록 마다 두개 사본 가정):
  1. 첫 물리적 블록상에 정보를 쓰라.
  2. 첫 쓰기가 성공적으로 완료될 때 두번째 블록에 같은 정보를 쓰라.
  3. 두번째 쓰기가 성공적으로 완료된 경우에만 완료된다.

저장 장치: 안전한 저장 장치 구현 2

- 블록 기록 중 시스템 실패로 두 블록의 내용이 다를 수 있다. 고장을 복구하기 위하여 두 개의 사본을 모두 검사해야 한다:
  1. 만약 둘이 같고 발견된 오류가 없으면 더 이상의 행위는 필요 없다.
  2. 만약 시스템이 어떤 한 블록에 오류가 있음을 발견한 경우 다른 블록의 내용을 대체시킨다.
  3. 만약 오류는 없으나 두 블록의 내용이 다를 경우 첫 블록으로 두 번째 블록의 값을 변경시킨다. 복구 절차는 안전 저장소에 성공적 기록이 이루어지거나 그렇지 않으면 아무런 변화가 없도록 해준다.
  4. 복구 중 모든 대응 블록 쌍의 비교는 고비용 요구, 따라서 소수의 SRAM을 사용하여 어느 블록에 기록 중이었는지 저장해 둠으로써 크게 비용을 감소시킬 수 있다. 즉 복구 때는 오직 기록 작업이 진행되었던 블록만 비교하면 된다.

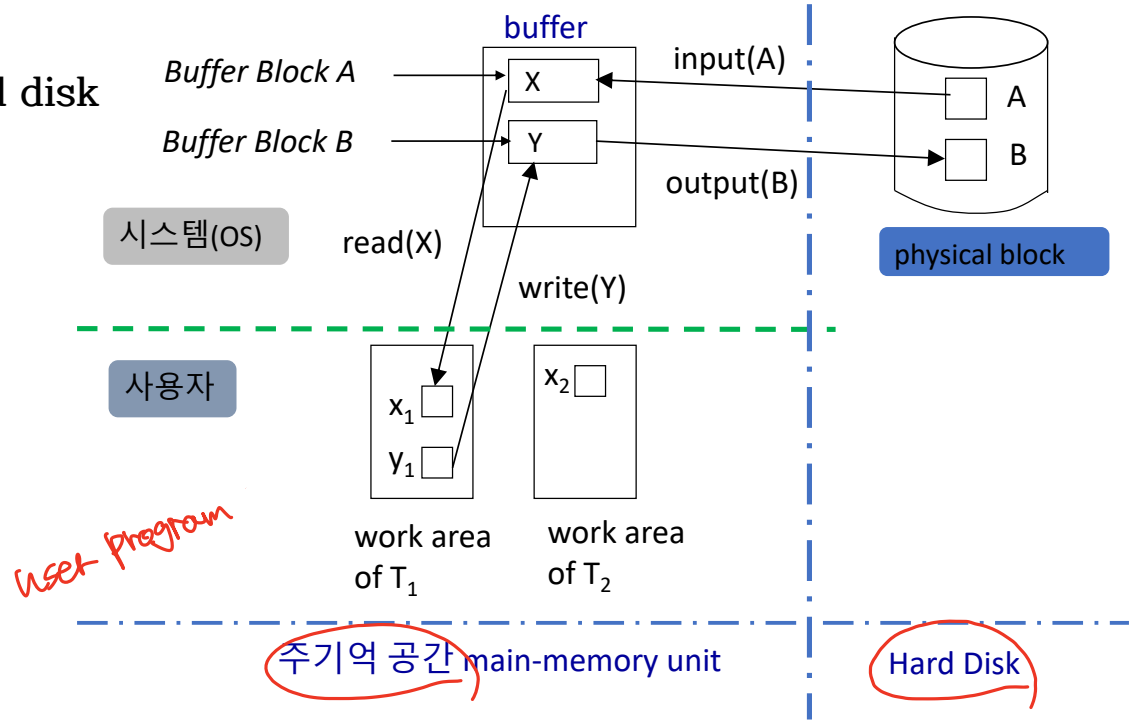
- 복구 중에 모든 대응 블록 쌍 비교는 고비용 요구.
  - 비휘발성 RAM으로 어느 블록에 기록 중이었는지를 저장해 둠으로써 크게 감소시킬 수 있다. 복구 때는 오직 기록 작업이 진행되었던 블록만 비교함.

자료 접근 Data Access

- 물리적 블록 Physical blocks
  - 디스크 상에 머물고 있는 블록들이다.
  - 트랙/섹터, 한 블록의 크기는 512B(일반)
- 버퍼 블록 Buffer blocks
  - 주기억장치에 일시적으로 머무르는 블록들이다.
  - 운영체제가 관리한다.
- 디스크와 주기억 사이 블록 이동들은 다음 두 연산을 통해서 초기화된다:
  - input(*B*) 물리 블록 B를 주기억으로 전송한다. 이를 버퍼블록이라 한다.
  - output(*B*) 버퍼블록 B를 디스크로 전송한다. 그리고 적합한 물리 블록을 대체한다.
- 단순히 가정하기를 각 자료항목은 단일 디스크 안에 저장된다.

자료 접근의 예

❖MMU와 hard disk



자료 접근 2

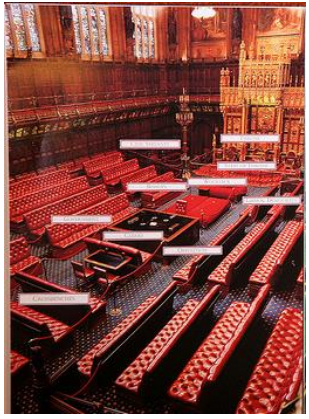
- 각 트랜잭션 T_i : 자신의 개별 작업영역을 가지고 있고, 거기서 접근되거나 갱신된 모든 자료항목 X의 지역 사본 xi는 유지된다.
- 시스템 버퍼 블록과 개별 작업 영역간에 자료 항목 전송:
  - read(X) 자료 항목 X의 값을 지역 변수 x_i에 할당한다.
    - ✓ X가 위치하는 블록 Bx가 메인 메모리에 없을 경우 input(Bx)를 실행한다.
    - ✓ 버퍼 블록으로부터 X의 값을 xi에 할당한다.
  - Write(X) 지역변수 x_i *의 값을 버퍼블록에 있는 자료항목* X에 할당한다.
    - ✓ X가 위치하는 블록 Bx가 메인 메모리에 없을 경우 input(Bx)를 실행한다.
    - ✓ 버퍼 Bx 내의 X에 xi 값을 할당한다.
- 디스크와 MMU 사이 블록 전송
  - 두 연산은 디스크 => MMU로 블록 전송 요구, 반면에 MMU로부터 디스크로 요구하지 않음
  - Output(Bx)는 write(X) 실행 즉시 실행될 필요 없다. 실패이면 기록 전에 자료 상실.
  - 강제기록(force-output) – DB 시스템이 버퍼 B에 대하여 output(B)를 실행하는 것

복구와 원자성 Recovery and Atomicity

- 고장임에도 불구하고 원자성을 보장하기 위하여, DB를 수정함이 없이 안전한 저장소에 수정된 정보들을 먼저 출력하라.
- 상세히 로그(등록,log)-기반 복구 알고리즘을 공부한다.
  - 먼저 핵심 개념을 나타내라.
  - 그리고 나서 실재 알고리즘을 나타낸다.
- 더 유용한 편법: 그림자 쪽짓기 shadow-paging
  - Shadow cabinet 그림자 내각

2021.11.15

여왕 폐하의 야당 그림자 내각
The Official Loyal Opposition Shadow Cabinet



귀족원(상원):792명

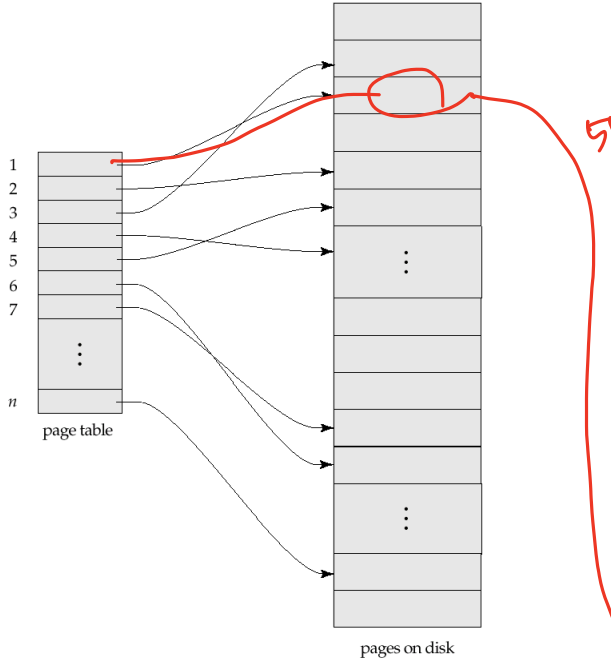
서민원(하원):650명

- The Official Loyal Opposition Shadow Cabinet (Shadow Cabinet) are, in British parliamentary practice, senior members of Her Majesty's Loyal Opposition who scrutinise their corresponding office holders in the Government, develop alternative policies, and hold the Government to account for its actions and responses.

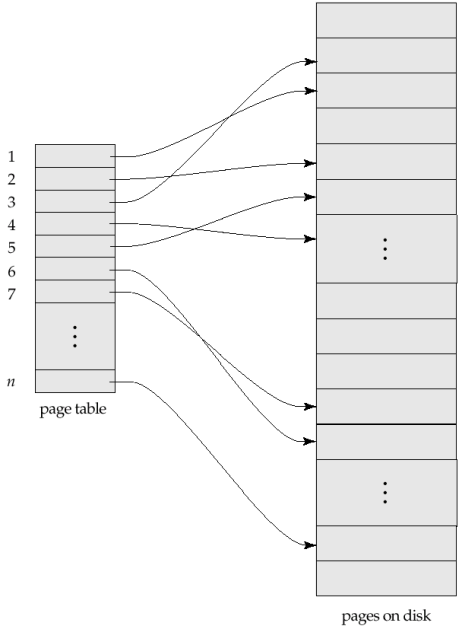
복구와 원자성: 그림자 쪽짓기 Shadow Paging

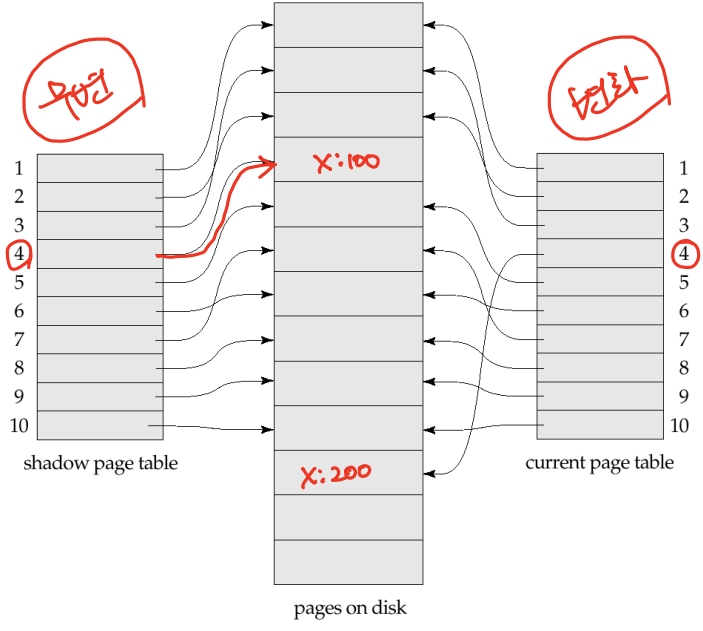
- 그림자 쪽 짓기는 등록(log) 기반 복구의 한 편법이다;
  - 이 전략은 유용한데 트랜잭션이 직렬적으로 수행하면 그렇다.
- 착상(idea): 트랜잭션의 일생동안 두 개의 쪽 표(page table)를 유지하라. 즉 현재 쪽 표 (current page table) 와 그림자 쪽 표 (shadow page table)
- 휘발성 저장소(volatile storage) RAM 내에 그림자 쪽 표를 저장하라. 즉 트랜잭션 수행에 앞서 DB의 상태는 복구될지도 모른다.
  - 그림자 쪽표는 수행동안 결코 수정되지 않는다.
- 시작할 때 두 쪽 표는 같다. 단지 현재 쪽 표는 트랜잭션 수행동안 자료항목 접근들을 위하여 사용된다. 즉 수정 사항이 반영된다.
- 어떤 쪽이 처음에 막 쓰여지려 할 때마다
  - 이 쪽의 한 사본이 사용 안된 쪽에 만들어진다..
  - 현재 쪽 표는 사본으로 지시하게 한다.
  - 갱신은 그 사본에서 이루어진다.

복구와 원자성:표본 쪽 표 Sample Page Table



복구와 원자성:그림자 쪽 짓기 예





복구와 원자성 : 로그 기반 복구 Log-Based Recovery

- 한 로그(등록)는 안전한 저장소에 유지된다.
  - 등록(log)은 등록 기록단위 (log records)의 배열이다. 그리고 DB 상에 갱신 활동의 기록단위를 유지한다.
- 트랜잭션 T_i 가 시작할 때 , <T_i start> 등록 기록단위를 씀으로써 그 자체를 기록한다.
- T_i 가 write(X)를 수행하기 앞서 등록 기록단위 <T_i, X, V_1, V_2> 를 쓰며, 여기서 V_1 은 원래 X 값 (즉 옛 값 old value)이며, V_2 는 쓰여진 뒤 X 값(새 값 new value)다.
- T_i 가 마지막 문장을 끝마칠 때 등록 기록단위 <T_i commit>가 쓰여진다.
- 등록을 사용하는 두 접근들
  - 지연 DB 수정 – 버퍼는 Ti 가 완료된 뒤에 디스크에 쓰여진다.
  - 즉시 DB 수정– 버퍼가 수정되면 즉시 디스크에 쓰여진다.

복구와 원자성: 즉시 DB수정 Immediate Database Modification

- 즉시 수정 계획은 미완료 상태에서 트랜잭션 갱신이 버퍼 혹은 디스크 그 자체에 이루어지기를 허용하는데, 트랜잭션이 완료(commit) 되기 전 상황에서 허용된다.
  - 갱신 등록 기록단위는 DB 항목이 쓰여지기 전에 쓰여져야 한다. 등록 기록단위가 직접적으로 안전 저장소에 출력된다고 가정하라 (등록 기록단위 출력을 어떤 한도까지 어떻게 연기 할 수 있는지를 나중에 볼 것임)
  - 트랜잭션 완료 전후로 안전 저장소에 갱신된 블록의 출력은 언제든지 일어날 수 있다.
  - 블록들이 출력되는 순서는 그들이 쓰여지는 순서와 달라질 수 있다.
- 지연된 수정 계획은 트랜잭션 완료commit의 시점에서 만 버퍼 또는 디스크에 갱신을 이행한다.
  - 복구의 일부 측면들을 단순화한다.
  - 그러나 지역 사본을 저장하는데 부담을 가진다.

복구와 원자성: 트랜잭션 완료 Transaction Commit

- 한 트랜잭션은 완료되었다고 하는데 그 완료 등록 단위가 안전 저장소에 출력되는 때 그렇다.
  - 트랜잭션의 모든 이전 등록단위는 이미 출력되었어야만 한다.

<Ti, commit>

- 한 트랜잭션에 의하여 이행된 쓰기 Writes는 트랜잭션 완료 때 아직 버퍼 안에 있을 지도 모른다. 그리고 나중에 출력될 지도 모른다.

WAL(Write Ahead Log): 쓰기전 로그
- Disk 또는 버퍼에 쓰기에 앞서서 로그에 기록한다.

복구와 원자성: 즉시 DB 수정 예

Log	Write	Output
<T_0 **start**>		
<T_0, A, 1000, 950>		
<T_0, B, 2000, 2050		
	A = 950	
	B = 2050	
<T_0 **commit**>		
<T_1 **start**>		
<T_1, C, 700, 600>		
	C = 600	
<T_1 **commit**>		

주의 : B_X 는 X를 포함하고 있는 블록 표시.

B_B , B_C — B_C output before T_1 commits

B_A — B_A output after T_0 commits

끝. 제19장 복구 -01

제19장 복구 -02로 연결