회복 시스템

컴퓨터과학과 정재화

한국방송통신대학교
Korea National Open University

학습목차

01

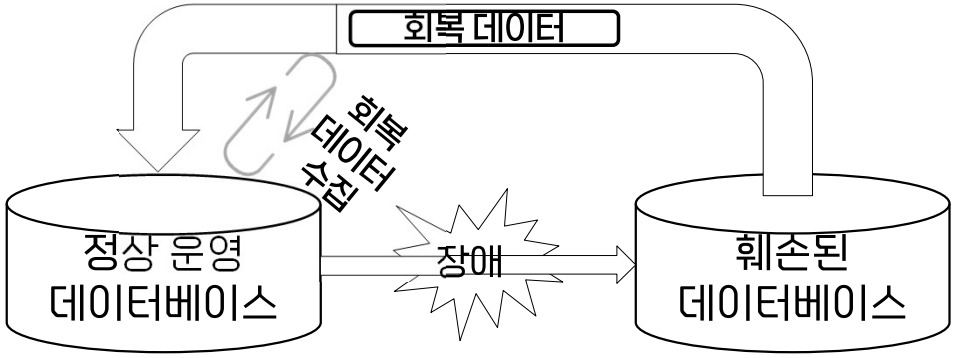
회복 시스템의 개념

- 회복의 역할
- 회복 데이터의 구성
- 데이터베이스 연산 처리 과정

한국방송통신대학교
Korea National Open University

회복의 역할

- 예상치 못한 HW 고장 및 SW 오류가 발생
  - 사용자의 작업에 대한 안정적 디스크 반영 여부 보장이 불가능
- 오류 발생 이전의 일관된 상태로 데이터베이스를 복원시키는 기법이 요구
  - 고장 원인 검출, DBMS의 안전성 및 신뢰성을 보장
- 데이터베이스는 데이터 복원 절차 내재화



한국방송통신대학교 Korea National Open University

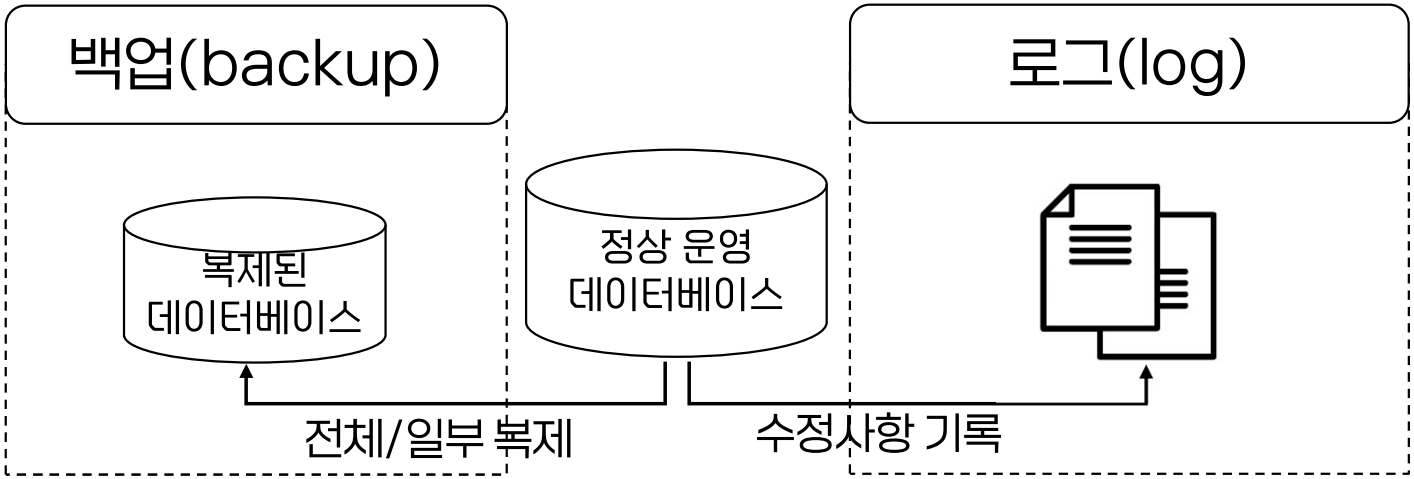
시스템 실패(system failure)의 유형

- 트랜잭션 실패
  - 논리적: 잘못된 데이터 입력, 부재, 버퍼 오버플로, 자원 초과 이용
  - 시스템적: 운용 시스템의 교착상태 발생
- 시스템 장애
  - 시스템의 하드웨어 고장, 소프트웨어의 오류
  - 주기억장치와 같은 휘발성 저장장치의 내용 손실
- 디스크 실패
  - 비휘발성 디스크 저장장치의 손상 및 고장으로 인한 데이터 손실

한국방송통신대학교
Korea National Open University

회복 데이터의 구성

- 백업: 데이터베이스의 일부 또는 전체를 주기적으로 별도의 저장장치에 복제하는 방식
- 로그: 데이터 변경 이전과 이후의 값을 별도의 파일에 누적 기록하는 방식



한국방송통신대학교
Korea National Open University

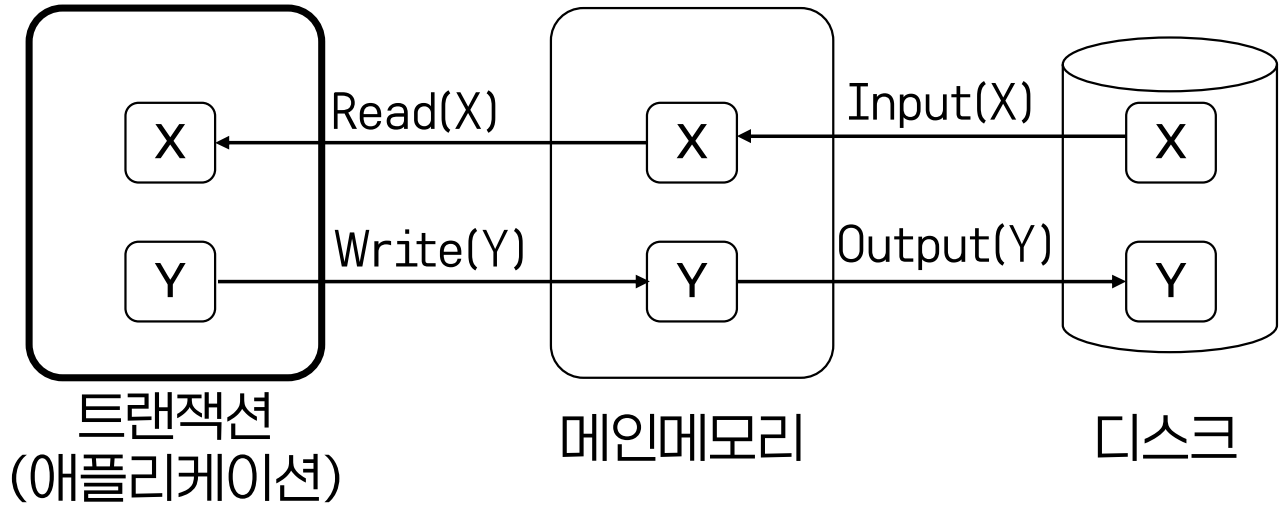
데이터 저장 구조

- 전체 데이터는 디스크와 같은 비휘발성 저장장치에 저장되며, 일부의 데이터만 주기억장치에 상주
- 데이터베이스는 데이터를 블럭(block) 단위로 전송하고 블럭 단위로 기억장소를 분할
- 트랜잭션은 디스크로부터 주기억장치로 데이터를 가져오며, 변경된 데이터는 다시 디스크에 반영

⊕ 가져오기, 내보내기 연산은 블럭 단위로 실행
- 물리적 블록: 디스크 상의 블럭
- 버퍼 블록: 주기억장치에 임시적으로 상주하는 블럭

데이터베이스 연산 처리 과정

- 메인 메모리와 디스크 사이의 연산
  - Input(X): 물리적 블록 X를 메인 메모리에 적재
  - Output(X): 버퍼 블록 X를 디스크에 저장



한국방송통신대학교
Korea National Open University

Read와 Write 연산 처리 과정

- Read(X)의 처리 과정
  - 버퍼 블럭 X가 메인 메모리에 없을 경우 Input(X)를 수행
  - 버퍼 블럭 X의 값을 변수 X에 할당
- Write(X)의 처리 과정
  - 버퍼 블럭 X가 메인 메모리에 없을 경우 Input(X)를 수행
  - 변수 X의 값을 버퍼 블럭 X에 할당

02

로그 기반 회복

- 로그 기반 회복의 개념
- Redo와 Undo 연산
- 체크포인트의 필요

한국방송통신대학교
Korea National Open University

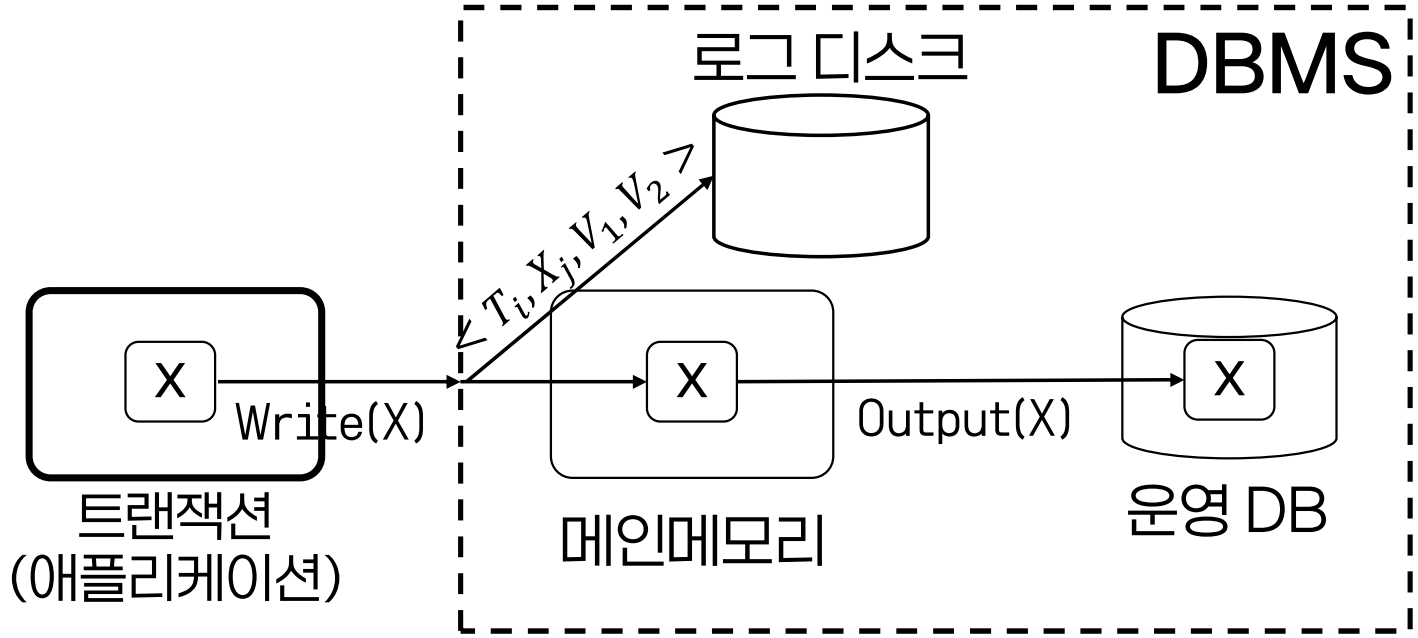
로그 기반 회복의 개념

- 데이터베이스가 수행한 모든 수정 작업을 기록한 여러 종류의 로그를 사용하여 회복하는 시스템
- 로그 레코드의 종류
  - $< T_i, X_j, V_1, V_2 >$: T_i가 데이터 항목 변경 연산을 수행하여 X_j의 값을 V_1에서 V_2로 변경
  - $< T_i, start >$: T_i가 시작
  - $< T_i, commit >$: T_i가 커밋
  - $< T_i, abort >$: T_i가 취소

한국방송통신대학교 Korea National Open University

데이터 항목 변경

- WAL(Write-Ahead Log)
  - 트랜잭션은 데이터베이스 수정 전,
    로그 레코드를 생성하여 기록



한국방송통신대학교
Korea National Open University

회복을 위한 연산

- 회복 기법은 로그에 대해 두 연산을 사용

Redo(T_i)	T_i에 의하여 수정된 새로운 값으로 데이터베이스의 데이터 항목값을 수정
Undo(T_i)	T_i에 의해 수정된 모든 데이터 항목을 이전 값으로 복귀, 완료 후 <T_i, $abort$> 기록

- 시스템 장애 발생 시
  - 로그에 $< T_i, start >$ 가 있지만 $< T_i, commit >$ 또는 $< T_i, abort >$ 를 포함하는 경우 T_i를 Redo
  - 로그에 $< T_i, start >$ 가 있지만 $< T_i, commit >$ 또는 $< T_i, abort >$ 를 포함하지 않는 경우 T_i를 Undo

한국방송통신대학교 Korea National Open University

데이터베이스 변경과 커밋

데이터베이스 변경 시 복구 알고리즘의 고려 사항

- 트랜잭션의 일부 변경 사항이 버퍼 블록에만 반영되고 물리 블록에 기록되지 않은 상태에서 트랜잭션이 커밋되는 상황
- 트랜잭션이 동작 상태에서 데이터베이스를 수정했으나 수정 후에 발생한 실패로 취소가 필요한 상황

트랜잭션 커밋 상황

- $< T_i, commit >$ 로그 레코드가 안정된 저장장치에 기록 완료 시 트랜잭션 커밋으로 간주
- $< T_i, commit >$ 로그 레코드가 기록되기 전에 장애가 발생하면 롤백

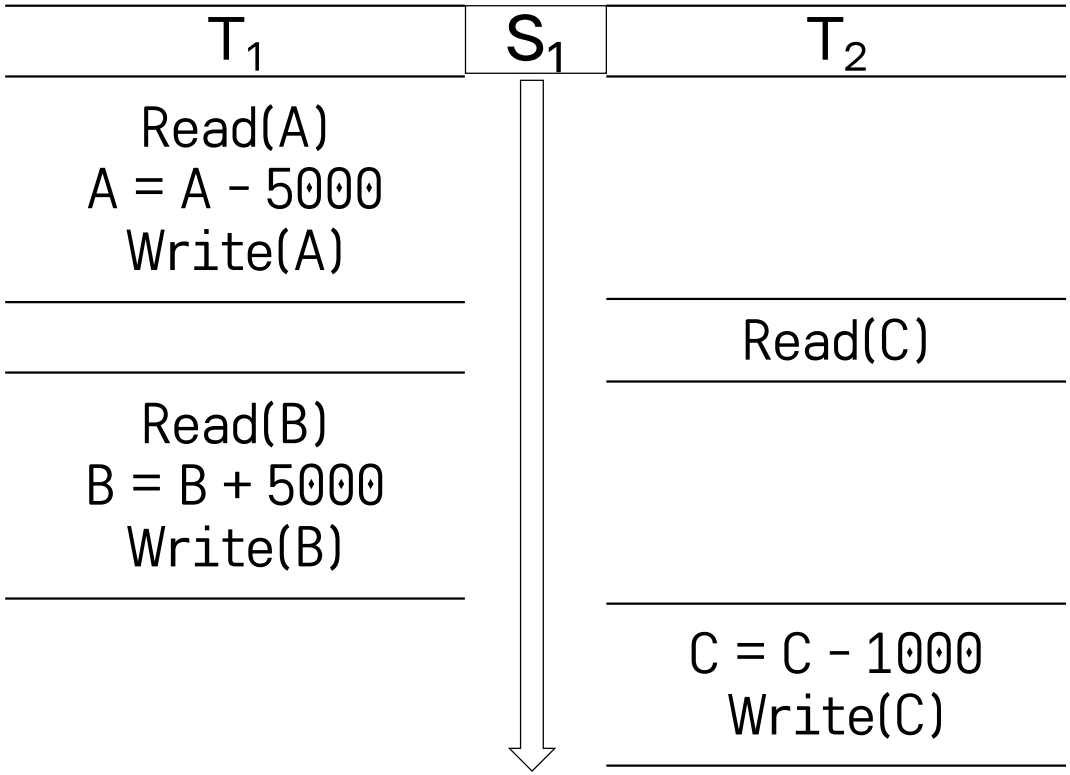
회복의 유형

- 회복은 트랜잭션에 의해 요청된 갱신 작업이 디스크에 반영되는 시점에 따라 구분
- 지연 갱신 회복(deferred update restore)
  - 부분 커밋까지 디스크 반영을 지연시키고 로그에만 기록
  - 실패 시, 별도의 회복 작업 필요 없이 로그만 수정
- 즉시 갱신 회복 (immediate update restore)
  - 갱신 요청을 곧바로 디스크에 반영
  - 실패 시, 디스크에 반영된 갱신 내용을 로그를 바탕으로 회복

한국방송통신대학교 Korea National Open University

은행 시스템의 트랜잭션의 예

- A = 30,000 B = 10,000 C = 50,000



한국방송통신대학교
Korea National Open University

로그와 데이터베이스 상태

로그	지연 갱신 DB	즉시 갱신 DB
$< T_1, start >$	A=30000,B=10000,C=50000	A=30000,B=10000,C=50000
$< T_1, A, 30000, 25000 >$	A=30000,B=10000,C=50000	A=25000,B=10000,C=50000
$< T_2, start >$	A=30000,B=10000,C=50000	A=25000,B=10000,C=50000
$< T_1, B, 10000, 15000 >$	A=30000,B=10000,C=50000	A=25000,B=15000,C=50000
$< T_1, commit >$	A=25000,B=15000,C=50000	A=25000,B=15000,C=50000
$< T_2, C, 50000, 40000 >$	A=25000,B=15000,C=50000	A=25000,B=15000,C=40000
$< T_2, commit >$	A=25000,B=15000,C=40000	A=25000,B=15000,C=40000

한국방송통신대학교 Korea National Open University

시스템 장애 발생 상황

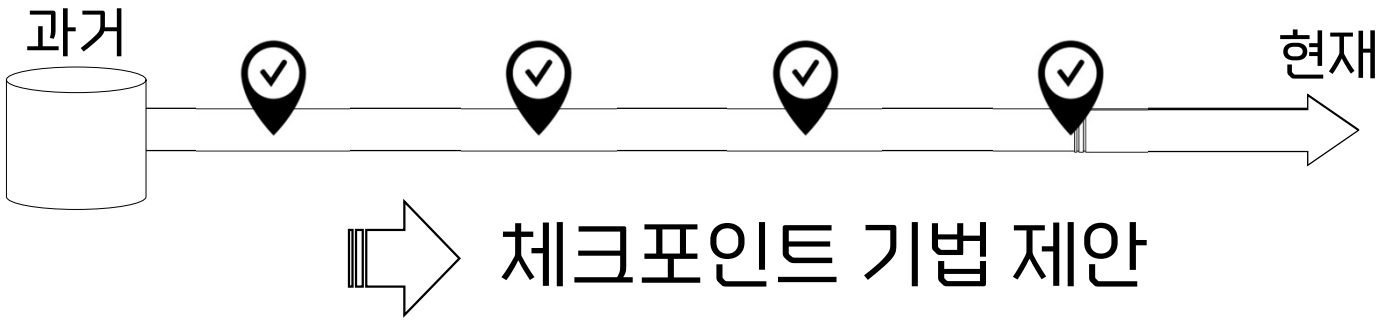
상황 1	상황 2	상황 3
$< T_1, start >$	$< T_1, start >$	$< T_1, start >$
$< T_1, A, 30000, 25000 >$	$< T_1, A, 30000, 25000 >$	$< T_1, A, 30000, 25000 >$
	$< T_2, start >$	$< T_2, start >$
	$< T_1, \mathrm{B}, 10000, 15000 >$	$< T_1, \mathrm{B}, 10000, 15000 >$
	$< T_1, commit >$	$< T_1, commit >$
	$< T_2, C, 50000, 40000 >$	$< T_2, C, 50000, 40000 >$
		$< T_2, \mathrm{commit} >$

한국방송통신대학교
Korea National Open University

체크포인트의 필요

로그 기반 회복 시스템의 한계

- 로그의 크기는 시간이 지남에 따라 계속 증가하므로 대용량 로그의 탐색 비용이 매우 커짐
- Redo를 해야하는 트랜잭션 중 대부분은 이미 데이터베이스에 반영
- 반영된 트랜잭션의 재실행은 시스템 자원의 낭비



체크포인트 기법 제안

한국방송통신대학교
Korea National Open University

체크포인트 기법

- 현재 시점에 메인 메모리의 버퍼 블럭에 존재하는 모든 로그 레코드를 안정 저장장치로 기록
- 수정된 모든 버퍼 블럭을 디스크에 반영
- 로그 레코드 $< checkpoint\ ListT >$를 안정한 저장장치에 기록
  - $ListT$는 체크포인트 시점에 실행 중인 트랜잭션 목록

한국방송통신대학교 Korea National Open University

체크포인트를 이용한 회복

체크포인트 기법을 이용한 회복 과정

- 로그의 마지막부터 역방향으로 탐색하여 $< checkpoint\ ListT >$ 레코드를 찾음
- $ListT$ 에 존재하는 $< checkpoint\ ListT >$ 이후에 실행된 트랜잭션에 대해서만 Redo와 Undo 연산 수행
  - 로그에 $< T_i, commit >$ 또는 $< T_i, abort >$가 없는 $ListT$안의 모든 트랜잭션을 Undo
  - 로그에 $< T_i, commit >$ 또는 $< T_i, abort >$가 있는 $ListT$안의 모든 트랜잭션을 Redo

한국방송통신대학교 Korea National Open University

03

회복 알고리즘

- 트랜잭션 롤백 알고리즘
- 시스템 장애 후 회복 알고리즘

트랜잭션 T_i의 롤백 알고리즘

- 1단계: 로그를 역방향으로 탐색
- 2단계: T_i의 로그 레코드 $< T_i, X_j, V_1, V_2 >$에 대하여
  - 데이터 항목 X_j에 V_1기록
  - 로그 레코드 $< T_i, X_j, V_1 >$을 로그에 기록
- 3단계: $< T_i, start >$를 찾은 이후
  - 역방향 탐색을 중단
  - 로그 레코드 $< T_i, abort >$를 로그에 기록

한국방송통신대학교
Korea National Open University

트랜잭션 롤백 알고리즘

T_0

```
Read(A);
A = A - 5000;
Write(A);
Read(B);
B = B + 5000;
Write(B);
```

트랜잭션

<T_0 start>
<T_2, C, 750, 60>
<T_0, A, 30000, 25000>
<T_5, D, 54900, 10>
<T_0, B, 10000, 15000>
<T_0, B, 10000>
<T_0, A, 30000>
<T_0, abort>

로그

한국방송통신대학교
Korea National Open University

시스템 장애 후 회복 알고리즘

- 시스템 장애 이후, 재시작 시 Redo와 Undo 단계를 수행
- Redo 단계
  - 최근의 체크포인트에서부터 순방향 로그 탐색
  - 롤백 대상할 트랜잭션의 Undo 리스트인 $ListofUndo$를 $ListT$로 초기화
  - $< T_i, X_j, V_1, V_2 >$,$< T_i, X_j, V_1 >$ 형태의 모든 레코드를 재실행
  - $< T_i, start >$ 발견 시, T_i를 $ListofUndo$에 추가
  - $< T_i, abort >$,$< T_i, commit >$ 발견 시 T_i를 Undo 리스트에서 제거

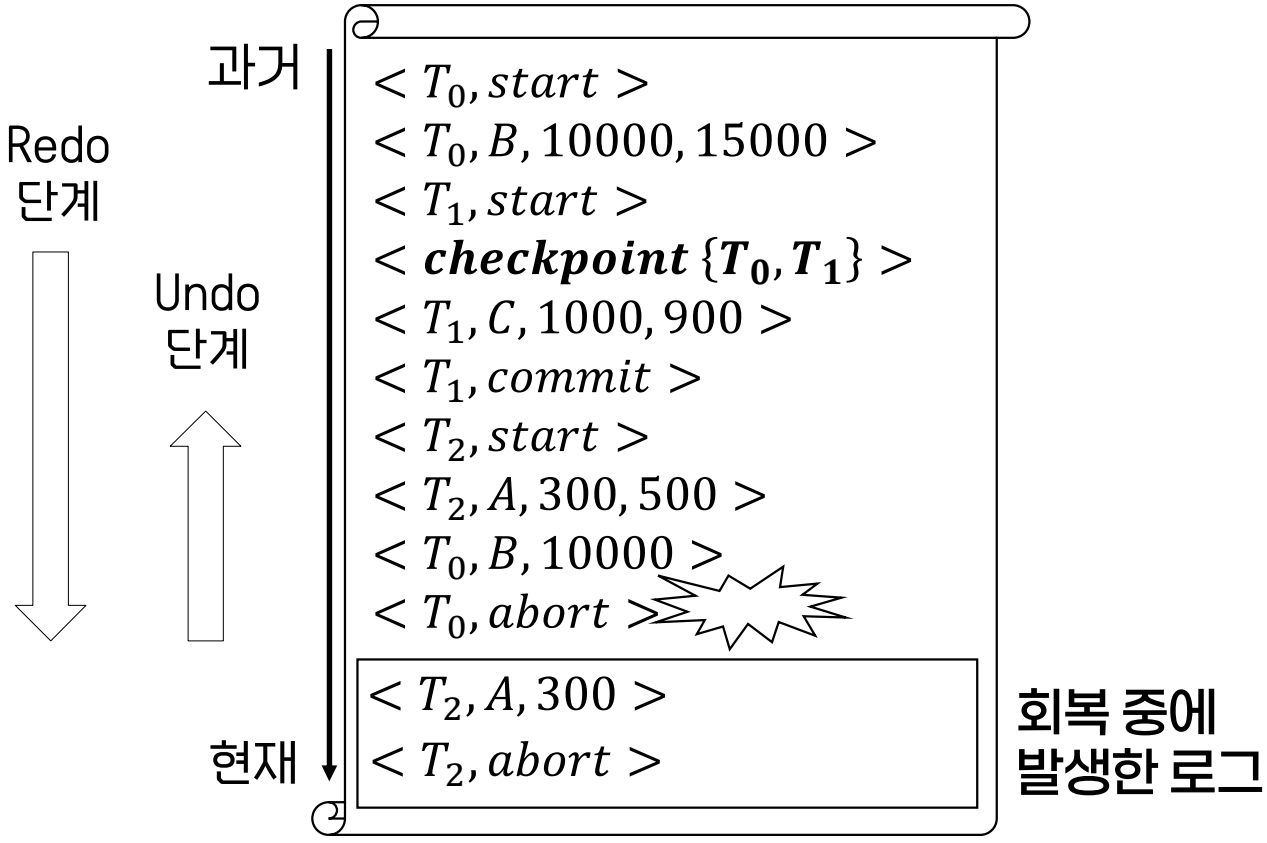
한국방송통신대학교
Korea National Open University

시스템 장애 후 회복 알고리즘

- Undo 단계 (역방향 로그 탐색)
  - $ListofUndo$의 트랜잭션의 로그레코드를 찾으면 트랜잭션 롤백 알고리즘 1단계 수행
  - $ListofUndo$의 트랜잭션 T_i에 대해 $< T_i, start >$를 만나면 로그에 $< T_i, abort >$를 기록하고 $ListofUndo$에서 제거
  - $ListofUndo$에 트랜잭션이 존재하지 않는 상태가 되면 Undo 단계를 종료

한국방송통신대학교 Korea National Open University

로그를 통한 회복 작업 과정



한국방송통신대학교
Korea National Open University

수고하셨습니다