

1

## 트랜잭션 시스템 개념

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요

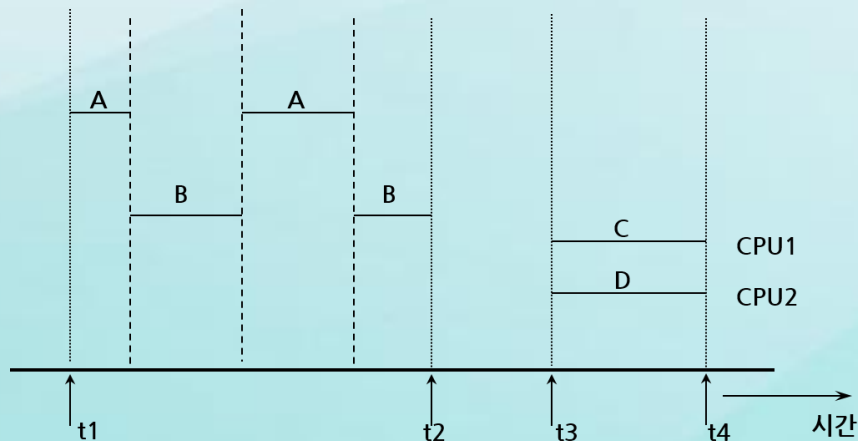
- 🔍 단일 사용자 시스템(Single User System)  
: 한 번에 한 사람만이 사용할 수 있는 시스템
- 🔍 다수 사용자 시스템(Multi User System)  
: 동시에 많은 사용자가 사용할 수 있는 시스템
  - 예 : 은행, 보험회사, 증권 거래소 등에서 사용되는 시스템

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요

다중 프로세스(병행 처리) 개념


- 인터리빙 : 1 프로세서, 다중 처리
- 병렬처리 : 복수개 프로세서, 다중 처리




※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요




 트랜잭션(Transaction)  
: 데이터를 다루는 일련의 연산들의 집합으로  
DBMS에서의 논리적 작업 단위로, ACID의  
특징을 가짐

 트랜잭션의 데이터베이스 접근 연산

- read\_item(X) : 읽기, 검색 연산
- write\_item(X) : 삽입, 삭제, 변경 연산

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요

-  읽기 전용 트랜잭션  
: 데이터를 검색만 하는 트랜잭션
-  트랜잭션의 읽기 집합  
: 그 트랜잭션이 읽는 모든 항목들의 집합
-  트랜잭션의 쓰기 집합  
: 그 트랜잭션이 기록하는 모든 항목들의 집합

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요


### 트랜잭션의 예

(a) 트랜잭션 T1  
read\_item(X);  
X:=X-N;  
write\_item(X);  
read\_item(Y);  
Y:=Y+N;  
write\_item(Y);

(b) 트랜잭션 T2  
read\_item(X);  
X:=X+M;  
write\_item(X);

# 01 트랜잭션 시스템 개념


## 1 트랜잭션 처리의 개요

 read\_item(X)의 수행 과정

- 1) 항목 X 를 포함하는 디스크 블록의 주소를 찾음
- 2) 그 디스크 블록을 주기억장치의 버퍼로 복사함  
(디스크 블록이 주기억장치의 버퍼에 이미 존재하는 것이 아닐 때)
- 3) 항목 X 를 버퍼로부터 프로그램 변수 a 에 복사함

# 01 트랜잭션 시스템 개념

## 1 트랜잭션 처리의 개요

 write\_item(X)의 수행 과정

- 1) 항목 X 를 포함하는 디스크 블록의 주소를 찾음
- 2) 그 디스크 블록을 주기억장치의 버퍼로 복사함  
(디스크 블록이 주기억장치의 버퍼에 이미 존재하는 것이 아닐 때)
- 3) 프로그램 변수 a 의 값을 항목 X 에 해당하는 버퍼의 정확한 위치로 복사함
- 4) 갱신된 블록을 디스크에 저장함  
(즉시 또는 나중의 어느 시점에)

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

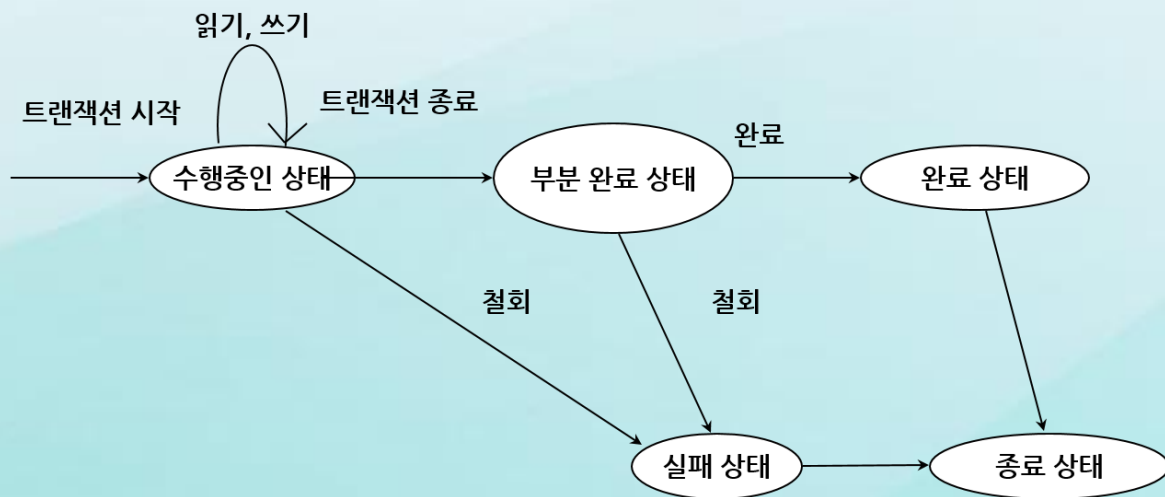
### 트랜잭션의 상태와 연산들

- 하나의 트랜잭션은 완전히 수행을 완료하거나 전혀 수행되지 않아야 함
- 회복을 위한 목적으로 회복 관리자는 트랜잭션에 대해 다음과 같은 연산들이 적용된 시점을 유지해야 함
  - 트랜잭션의 시작(BEGIN\_TRANSACTION)
  - 읽기 또는 쓰기(READ or WRITE)
  - 트랜잭션의 종료(END\_TRANSACTION)
  - 트랜잭션의 완료(COMMIT\_TRANSACTION)
  - 복귀(ROLLBACK) 또는 철회(ABORT)

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

### 🔍 트랜잭션의 상태 전이도



※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년


# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그


- 🔍 시스템 로그 파일(System Log File)  
: 트랜잭션 실패를 회복하기 위해서  
시스템은 로그(log)를 유지함
- 🔍 로그에는 데이터베이스 항목에 관련된  
트랜잭션의 모든 연산들을 기록하며  
이 정보는 트랜잭션 실패를 회복하는데 사용됨
- 🔍 로그는 디스크 오류를 제외한 어떠한 오류에도  
영향을 받지 않도록 하기 위하여 디스크에 유지됨
- 🔍 로그를 주기적으로 테이프 등에 백업하여  
디스크 오류 등의 재해로부터 대비함

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

 로그에 기록되는 항목들의 유형은 다음과 같음  
여기서 T2 는 트랜잭션 식별자임

- [start\_transaction, T2]
- [write\_item, T2, X, old\_value, new\_value]
- [read\_item, T2, X]
- [commit, T2]
- [abort, T2]

 로그를 이용하여 트랜잭션에 의해  
변화된 내용을 취소하거나 재수행 할 수 있음

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

- 연쇄 복귀(Cascading rollback)를 방지하는 회복 프로토콜에서는, 거의 모든 실제 프로토콜이 이에 해당하는데, 읽기 연산을 시스템 로그에 기록할 필요가 없음
- 그러나 모든 데이터베이스 연산을 추적해야 하는 감사(Auditing) 시스템 같이 로그를 다른 목적으로 사용하는 경우에는 읽기 연산을 시스템 로그에 기록해야 함

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

- 🔍 트랜잭션의 완료점(Commit)  
: 데이터베이스를 접근하는 그 트랜잭션의 모든 연산들이 성공적으로 수행되어 그 결과가 실제 데이터베이스에 저장되고, 트랜잭션의 모든 연산들이 데이터베이스에 끼친 효과들이 로그에 저장되었을 때, 트랜잭션 T는 완료점에 도달했다고 함
- 🔍 시스템 고장이 일어나면 로그에 [start\_transaction, T]를 기록했지만 [commit, T]를 기록하지 않은 모든 트랜잭션 T를 찾아서 복귀시킴

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

- 🔍 로그에 commit 레코드를 기록한 트랜잭션은 commit 레코드 이전에 먼저 자신의 모든 쓰기 연산들을 반드시 로그에 기록해야 함
- 🔍 로그 레코드가 추가될 때마다 디스크에 직접 기록하는 대신에 로그 화일의 한 블록을 주기억장치에 유지하고 그것이 로그 레코드들로 채워질 때마다 디스크에 한 번만 기록하는 방법이 보편적으로 사용됨

# 01 트랜잭션 시스템 개념

## 2 트랜잭션 상태와 로그

- 🔍 트랜잭션이 완료점에 도달하기 전에 아직 디스크에 저장하지 못한 로그의 내용이 있다면, 블록이 꽉 차지 않은 상태라도 이를 반드시 디스크에 저장해야 함, 이 과정을 트랜잭션의 완료 전 로그 파일의 강제 쓰기 (Force-writing)라고 한다

## 2 트랜잭션의 특징

## 02 트랜잭션의 특징

### 1 트랜잭션의 ACID 성질



원자성(Atomicity)

: 한 트랜잭션은 하나의 원자적 수행 단위임,  
트랜잭션은 완전히 수행되거나 전혀 수행되지  
않아야 함(All or Nothing)



일관성 유지(Consistency Preservation)

: 트랜잭션을 완전히 실행하면 데이터베이스를  
하나의 일관된 상태에서 또 다른 일관된 상태로  
바뀌어야 함

## 02 트랜잭션의 특징

### 1 트랜잭션의 ACID 성질



#### 고립성(Isolation)

: 하나의 트랜잭션은 다른 트랜잭션들과는 독립적으로 실행되는 것처럼 보여야 함, 즉 하나의 트랜잭션의 실행은 동시에 실행 중인 다른 트랜잭션의 간섭을 받아서는 안됨

- 고립의 정도를 나타내는 0 ~ 3 단계의 고립성 등급이 있음




#### 지속성(Durability)

: 일단 한 트랜잭션이 데이터베이스를 변경시키고 그 변경이 완료되면 그 변경은 이후의 어떠한 고장에도 손실되지 않아야 함

### 3 병행제어의 필요성

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성

 적절한 제어 없이 트랜잭션들을 동시에 실행하였을 때 여러 가지 문제가 발생할 수 있음

- 갱신 손실 문제 (Lost Update Problem)
- 오손 읽기/임시 갱신 문제 (Dirty Read/Temporary Update Problem)
- 부정확한 요약 문제 (Incorrect Summary Problem)
- 반복할 수 없는 읽기 문제 (Unrepeatable Read Problem)

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



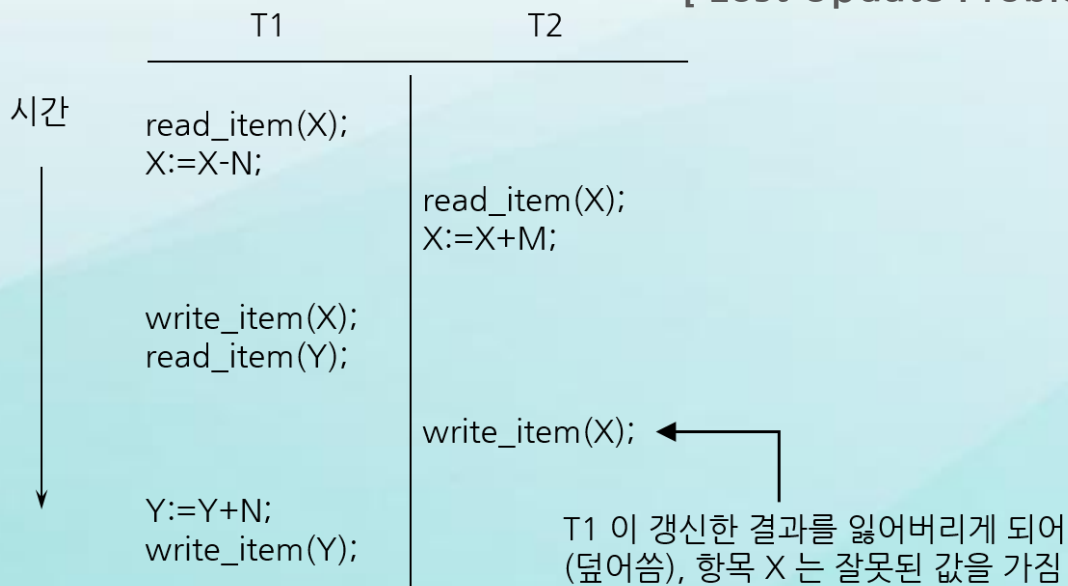
갱신 손실 문제 (Lost Update Problem)

- T1이 변경한 X 값을 데이터베이스에 기록하기 전에 T2가 X 값을 읽기 때문에 발생하는 문제

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성

[ Lost Update Problem ]



※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



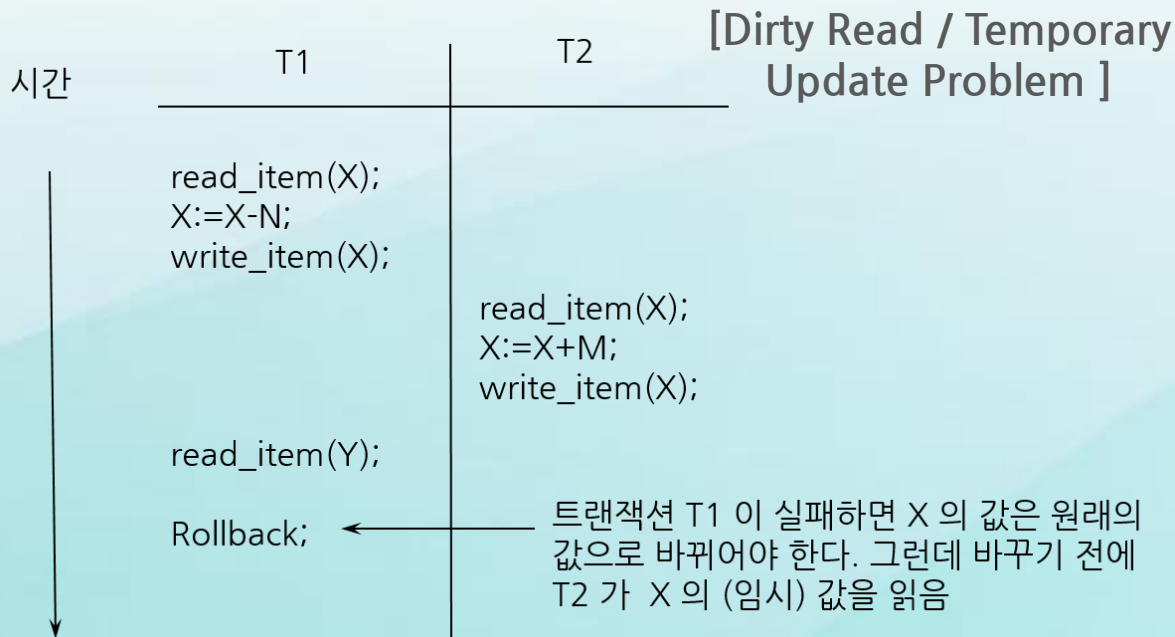
오손 읽기 / 임시 갱신 문제

(Dirty Read/Temporary Update Problem)

- T1 이 항목 X 를 갱신한 후 완료 되기 전에 실패했을 때 시스템은 X 를 원래의 값으로 되돌려야 함, 그런데, 원래값으로 되돌리기 전에 T2 가 X 의 임시값을 읽기 때문에 발생하는 문제

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

## 03 병행 제어의 필요성

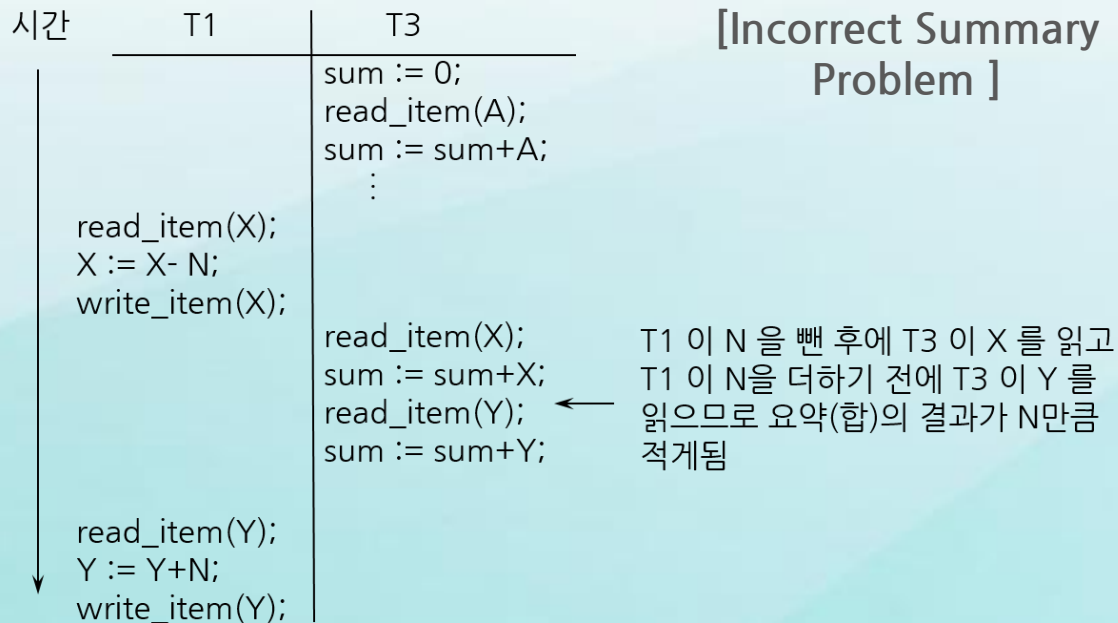
### 1 CC (Concurrency Control)의 필요성

#### 부정확한 요약 문제(Incorrect Summary Problem)

- 하나의 트랜잭션이 여러 항목에 대해서 집단합수를 계산하고 있고 다른 트랜잭션이 그 항목들 중 일부를 갱신하고 있다면 이 집단합수는 항목들 중 어떤 것들은 갱신되기 전의 값으로, 어떤 것들은 갱신된 이후의 값으로 계산할 것임
- T3가 총수를 계산하고 있을 때 T1 도 수행 중이라고 가정하면 T1 이 X 에서 N 만큼 뺀 후에 T3 이 X 를 읽고 T1 이 Y 에 N 만큼 더하기 전에 T3 이 Y 를 읽기 때문에 T3 의 결과는 N 만큼 작아짐

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



반복할 수 없는 읽기 문제  
(Unrepeatable Read Problem)

- 트랜잭션 T1 이 수행 도중 하나의 항목을 두 번 읽을 때 두 번 읽기 연산 사이에 다른 트랜잭션 T2가 그 항목이 값을 바꾸는 것임, 따라서 T1 은 동일한 항목을 서로 다른 값으로 읽게 됨

## 03 병행 제어의 필요성

### 1 CC (Concurrency Control)의 필요성



반복할 수 없는 읽기 문제  
(Unrepeatable Read Problem)

- 예를 들어, 하나의 항공 예약 트랜잭션 T1 이 실행되는 동안 어떤 고객이 여러 항공편에 대해서 남아 있는 좌석 수를 문의한 뒤 특정 항공편을 결정할 수 있음, 그러면 T1 은 해당 예약 작업을 끝내기 전에 그 항공편의 좌석 수를 한번 더 읽게 되는데 T2 가 그 중간 사이에 좌석 수를 갱신하게 되면 다른 값을 읽게 되는 문제가 발생함