

# 병행 수행과 병행 제어

↓  
사용자가 DB를 동시에  
공유할 수 있도록 여러  
트랜잭션이 동시에 수행  
(concurrency)

↓  
동시에 실행된 것처럼 보임

실제 여러 트랜잭션이  
"차레로 번갈아" 수행되는  
인터리빙 방식

병행성(동시)성 vs 병렬성

(동시)  
↙ 여러 개 트랜잭션이 병행 수행하면서  
같은 데이터 접근하여 연산을 실행하면  
생길 수 있는 문제를 발생하지 않도록  
제어하는 것을 "병행 제어"

# 병행 수행 문제 (갱신 분실, 모순성, 연쇄 복귀)

갱신 분실 (lost update)

- 데이터 변경 연산을 덮어써 변경 사항이 무효화됨

## 모순성

일관성 없는 DB 데이터를 가져와서 연산을 실행하여 "모순된 결과"가 발생

# 연쇄 복귀

앞서 모순성 이서와 비슷하게, 1 트랜잭션이 수행될 때, 2 트랜잭션이 중간에 연쇄를 수행하고 마쳤다. 이제 다시 1 트랜잭션을 수행하려 하는데 장애가 발생. 이이 2 트랜잭션이 1 트랜잭션의 데이터를 DB에 보냈기에 rollback 할 수 없게 됨.

“공유 데이터 접근 시”

트랜잭션 단위 수행하는 것이 매우 중요

# 트랜잭션 스케줄

병행 수행에서는 트랜잭션 순서에 따라 결과가 달라질수 있기 때문에 "순서"가 중요.

직렬 : 인터리빙 방식을 이용 X. 트랜잭션 별 연산을 순차적 실행

비직렬 : 인터리빙 사용. 트랜잭션들을 병행해서 수행

직렬 가능 : 직렬과 같이 정확한 결과를 생성하는 비직렬

# 직렬 스케줄

· 직렬스케줄은 하나의 트랜잭션이 끝날 때까지 나머지 트랜잭션 기다린다.

⇒ 모순 발생 X . 독립적

# 비직렬 스케줄

· 갱신 분실. 모순성. 연쇄 복귀 등 문제 발생 가능성 ⇒ 최종 결과 정확성 보장 X

· 트랜잭션이 돌아가면서 연산을 실행

# 직렬 가능 스케줄

· 책에서 트랜잭션의 연산수행 순서에 따라 결과값이 정확할 경우, 잘못된 경우가 나온다.

정확한 경우  $\Rightarrow$  직렬 가능 스케줄

잘못된 경우  $\Rightarrow$  직렬 가능 스케줄 X

· 인터리빙 . 병행 수행  $\Rightarrow$  정확한 값

· 직렬 가능 스케줄인지 판단하기 어려움

$\Rightarrow$  대부분 DBMS는 직렬 가능 스케줄인지를 검사하기 보다는

직렬 가능성을 보장하는 "병행 제어 기법" 사용

# 병행 제어 기법

- 병행 수행을 하면서 정확한 결과를 얻을 수 있는 작렬 가능성을 보장받기 위해

## Locking 기법

- 병행 수행되는 트랜잭션들이 동일한 데이터에  
동시에 접근하지 못하도록 lock.unlock 연산 이용  
⇒ 상호 배제하여 작렬 가능성 보장

동일한 데이터에 접근할 때 lock을 걸어 누구도 접근하지 못하게 막고 나서  
연산을 모두 수행한 나을 때 lock을 풀어주는 unlock 연산을 수행하여  
데이터에 접근할 수 있게 한다

lock 연산을 실행하는 대상에 따라 trade-off가 생긴다

↳ 전체 DB. 속성 릴레이션. 튜플 가능

단위 ↑ : 병행성 ↓. 제어 쉽다

단위 ↓ : 병행성 ↑. 제어 어렵다

) 시스템에 따라 적절히 단위 선택..



## 2단계 로킹규약

locking 기법을 사용해도 모순된 결과를 생성할 수 있다.

2단계 로킹규약은 축소·확장 단계로 나뉘어진다.

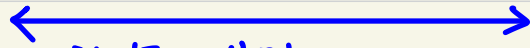
결국 모든 lock 연산들이, unlock 연산보다 먼저 실행되어야 한다.

단점: 교착상태, 연쇄복귀문제

↳  $T_1$ 이 현재  $X$ 를 lock하고  $Y$ 에 접근하려 할 때  
 $T_2$ 가 현재  $Y$ 를 lock하고  $X$ 에 접근하려 할 때 )

# locking & 2단계 locking 규약

lock X  
read X  
 $X += 1000$   
write X  
unlock X  
lock Y



차이를 보면  
일반적인 locking 기법은  
lock과 unlock이 섞여  
있다.

확장·축소 단계가 나뉘지  
않는 것을 알 수 있다.

lock X  
read X  
 $X += 1000$   
lock Y  
unlock X

} 확장 단계

- 축소 단계