데이터베이스

교착상태

학습목표

- 로킹 기법에 따른 교착 상태의 발생 원인을 파악하고, 해결할 수 있다.
- 교착 상태 해결을 위한 다양한 기법을 설명할 수 있다.

학습내용

- 교착 상태의 발생
- 교착 상태의 해결

교착 상태의 발생

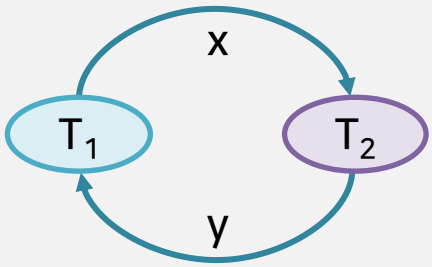
교착 상태

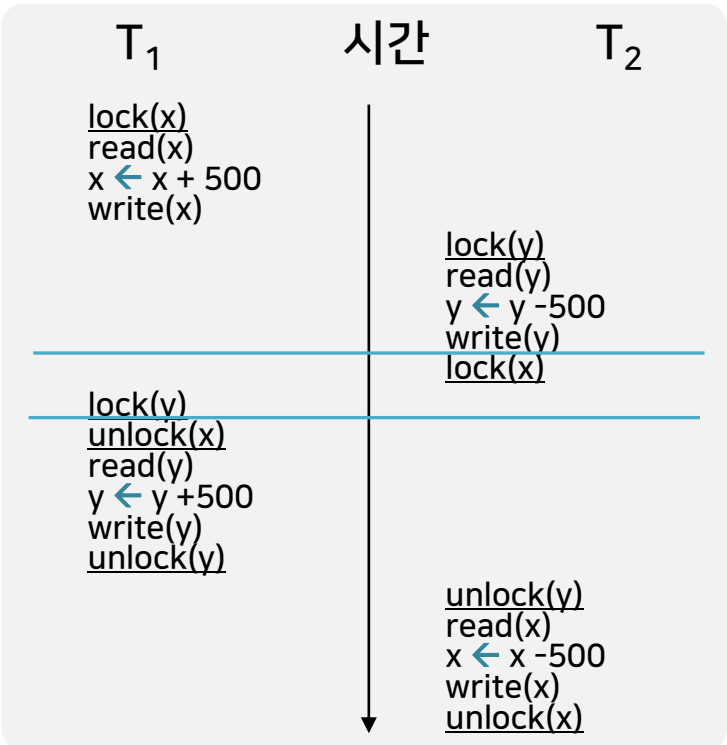
0 교착 상태의 의미

> 모드 트랜잭션이 실행을 전혀 진전 시키지 못하고,
> 무한정 기다리고 있는 상태

T_1 ⋯▸ T_2가 데이터 아이템 x를 unlock하기 위해 대기

T_2 ⋯▸ T_1이 데이터 아이템 y를 unlock하기 위해 대기





2PLP

교착 상태의 발생

교착 상태

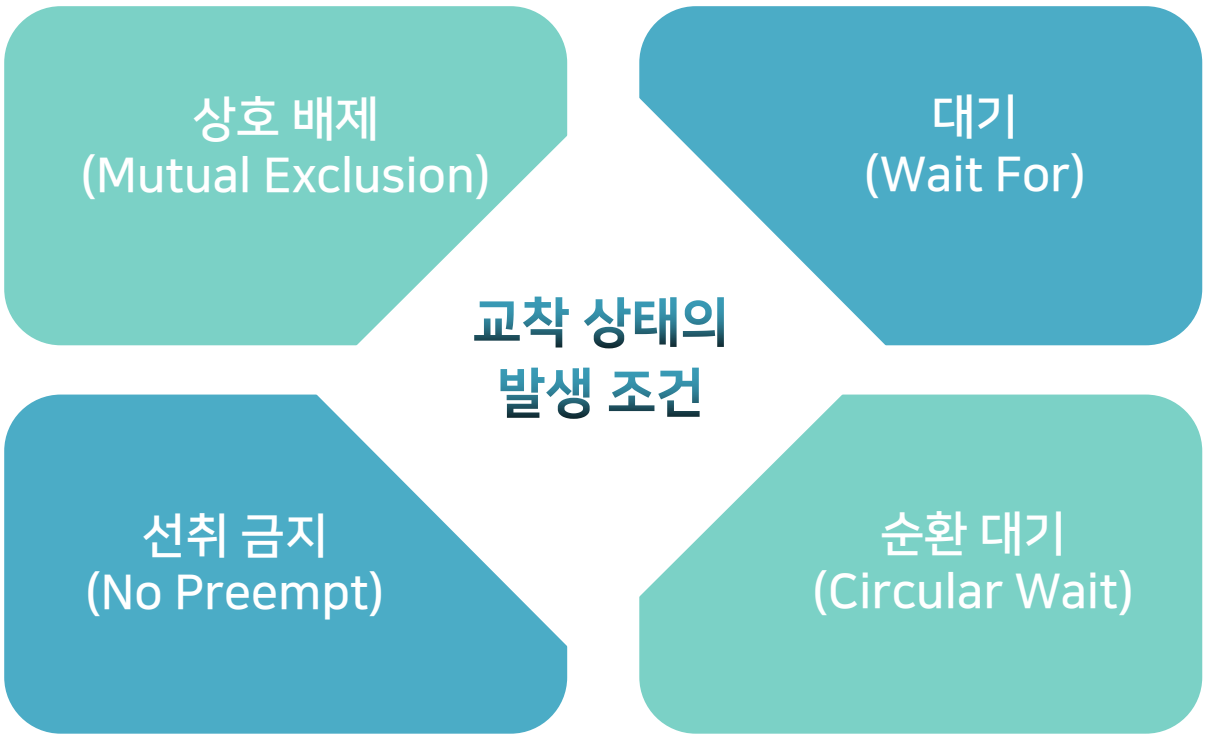
0 교착 상태(Dead Lock)의 발생 조건

상호 배제(Mutual Exclusion)	대기(Wait For)
독점적 로킹	로킹을 얻기 위한 대기
선취 금지(No Preempt)	**순환 대기(Circular Wait)**
트랜잭션이 계속 살아서 존재	서로 대기

4가지 조건이 동시에 모두 발생되어야 교착 상태가 발생

교착 상태의 발생

교착 상태의 해결 방법



4가지 조건 중 하나라도 없애면 교착 상태 해결

교착 상태의 발생

교착 상태의 해결 방법

탐지 (Detection)	교착 상태가 일단 일어난 뒤, 교착 상태 발생 조건의 하나를 제거하는 방법
예방 (Prevention)	트랜잭션 실행 전, 교착 상태 발생을 불가능하게 만드는 방법
회피 (Avoidance)	자원을 할당할 때마다 교착 상태가 일어나지 않도록 실시간 알고리즘을 사용하여 검사하는 방법

교착 상태의 해결

교착 상태 탐지

0 교착 상태 탐지 시 필요 사항

시스템의 정보 유지

- 현재 로크된 데이터
- 로크 요청이 대기 중인 데이터(Pending Data)

시스템 검사 알고리즘

- 교착 상태를 탐지 하기 위해 주기적으로 가동
- 교착 상태 회복 기법

교착 상태의 해결

교착 상태 탐지

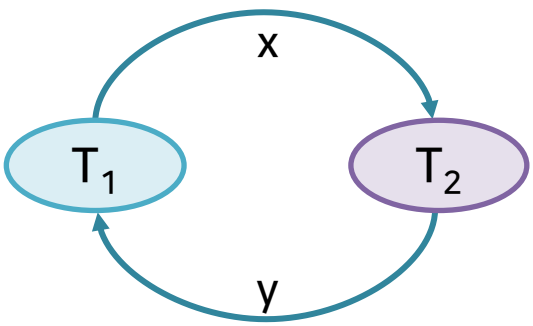
0 교착 상태 탐지 시 대기 그래프(V, E)

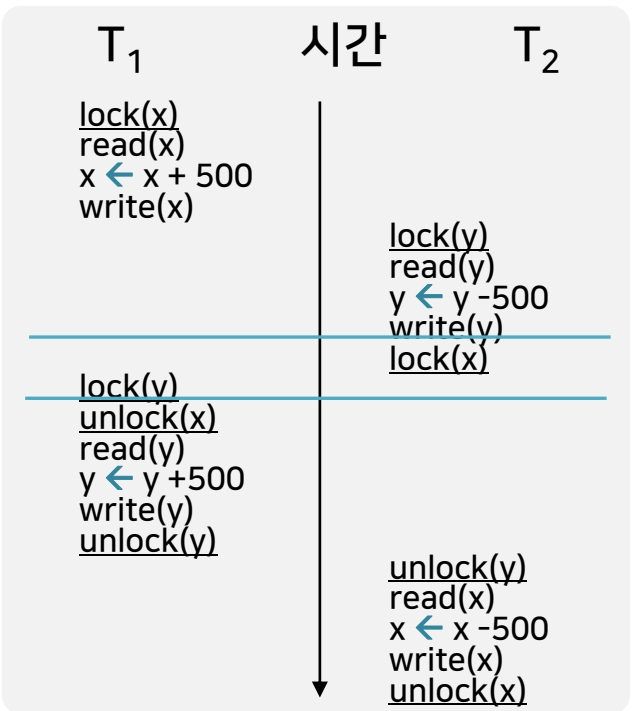
구성 요소

- V: 트랜잭션
- E: ($T_i \rightarrow T_j$) → T_i가 T_j를 대기 중
- 사이클 → 교착 상태

Q 교착 상태 검사 시기의 기준은 무엇인가?

- 실행 중인 트랜잭션의 수 또는 데이터를 로크 하기 위해 트랜잭션들이 대기하는 기간





교착 상태의 해결

교착 상태 예방

0 교착 상태 예방 기법의 의미

> 트랜잭션 실행 전, 교착 상태 발생을 불가능하게 만드는 방법

트랜잭션 스케줄링 기법 ···→	트랜잭션 실행 전 필요한 데이터 아이템들을 모두 로크

교착 상태의 해결

교착 상태 예방

0 교착 상태 예방 기법의 문제점

트랜잭션 스케줄링 기법의 단점

01 충돌되는 데이터를 필요로 하는 트랜잭션은 병행 실행이 아예 불가

02 데이터 요구에 대한 사전 지식이 필요

03 데이터 아이템의 활용도 감소

- 데이터 아이템이 한꺼번에 로크 되기 때문

04 기아(Starvation) 문제 발생

- 자주 사용하는 데이터 아이템이 필요한 트랜잭션은 다른 트랜잭션이 그 데이터 아이템을 쓰는 동안 무한정 대기해야 하는 상황 발생

현실적 활용도 부족

교착 상태의 해결

교착 상태 예방

0 교착 상태 예방 기법의 문제점

Q 데이터베이스에 있는 모든 아이템들에 일정한 순서를 정하면?

- 모든 트랜잭션들이 이 순서에 따라 데이터 아이템을 로크하도록 지시
- 프로그래머가 데이터 아이템의 순서를 숙지하도록 지시

현실성 부족

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피의 의미

자원을 할당할 때마다 교착 상태가 일어나지 않도록
실시간 알고리즘을 사용하여 검사하는 방법

타임스탬프 이용

- 트랜잭션의 시작 순서에 기초하는 식별자(Identifier)
- 트랜잭션이 기다려야 할지 복귀해야 할지 결정하는데 사용
- 트랜잭션 재시도
  → T_2에 의해 로크된 x를 T_1이 요구할 때

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

» 예 | T_j가 로크한 데이터 아이템을 트랜잭션 T_i가 요청할 때

Wait-Die 기법 | Wound-Wait 기법

- T_i의 타임스탬프가 T_j의 것보다 작은 경우($TS(T_i) < TS(T_j)$)
  - T_i가 고참인 경우: T_i는 대기
  - 그렇지 않으면 T_i는 복귀(즉 Die)하고 다시 시작

Wait-Die 기법 | **Wound-Wait 기법**

- T_i의 타임스탬프가 T_j의 것보다 클 경우($TS(T_i) > TS(T_j)$)
  - T_j가 고참인 경우: T_j는 대기
  - 그렇지 않으면 T_j는 복귀해서(T_i는 T_j를 상처 입힘) 다시 시작

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

Wait-Die와 Wound-Wait 기법의 차이점

	Wait-Die 기법	Wound-Wait 기법
트랜잭션의 대기	고참 트랜잭션이 신참 트랜잭션을 기다림	고참 트랜잭션은 신참 트랜잭션을 기다리지 않음
트랜잭션의 데이터 요구	고참 트랜잭션이 가지고 있는 데이터를 신참 트랜잭션이 요구하면 신참 트랜잭션은 복귀 후 재실행	신참 트랜잭션이 가지고 있는 데이터를 고참 트랜잭션이 요구하면 신참 트랜잭션은 복귀 후 재실행
트랜잭션의 재실행	재실행 시 똑같은 순서로 데이터를 요구 (불필요한 복귀 자주 발생)	신참 트랜잭션은 기다리기만 하면 됨 (불필요한 복귀가 비교적 덜 발생)

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

타임스탬프 순서 기법

교착 상태의 주요 원인	해결
로킹	로킹 대신 동시성 제어 시도

Q 트랜잭션을 인터리브로 실행한다면?

- 타임스탬프 순서로 직렬 가능

- 타임 스탬프 순서로 각 데이터 아이템에 접근하는 것을 보장

Q 접근한 트랜잭션이 가장 최근 접근한 트랜잭션보다 더 오래되었다면?

- 새로운 타임스탬프로 재시작

타임스탬프란?(TS)
시스템의 클록(Clock) 값 또는 논리적 카운터

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

타임스탬프 순서 기법

- 트랜잭션 T_i의 타임스탬프: $TS(T_i)$

시스템이 생성한 유일한 식별자

T_i가 T_j보다 오래되면, $TS(T_i) < TS(T_j)$

시스템이 <T_i, T_j>의 **직렬 실행**과 결과가 일치하도록 보장

- 데이터 아이템 x의 타임스탬프

read_TS(x)	write_TS(x)
▶ 데이터 아이템 x의 판독시간 스탬프 ▶ read(x)를 성공적으로 수행한 트랜잭션의 타임스탬프 중에서 제일 큰 타임스탬프	▶ 데이터 아이템 x의 기록시간 스탬프 ▶ write(x)를 성공적으로 수행한 트랜잭션의 타임스탬프 중에서 제일 큰 타임스탬프

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

타임스탬프 순서 기법

- 데이터 아이템 x의 타임스탬프

01 T_i가 read(x)를 수행하려 할 때

- $TS(T_i) < write_TS(x)$이면: T_i가 너무 늦게 올 경우 T_i 복귀
  - $TS(T_i)$보다 타임스탬프가 큰 트랜잭션이 x의 값을 먼저 변경시킴
- $TS(T_i) \geq write_TS(x)$이면: 실행

$$read\_TS(x) \leftarrow \max\{read\_TS(x), TS(T_i)\}$$

02 T_i가 write(x)를 수행하려 할 때

- $TS(T_i) < read_TS(x)$이면: T_i가 너무 늦게 올 경우 T_i 복귀
  - $TS(T_i)$보다 타임스탬프가 큰 트랜잭션이 x의 값을 먼저 판독함
- $TS(T_i) < write_TS(x)$이면: T_i가 너무 늦게 올 경우 T_i 복귀
  - $TS(T_i)$보다 타임스탬프가 큰 트랜잭션이 x의 값을 먼저 기록함
- $TS(T_i) \geq read_TS(x)$이고, $TS(Ti) \geq write_TS(x)$이면: 실행

$$write\_TS(x) \leftarrow TS(T_i)$$

교착 상태의 해결

교착 상태 회피

0 교착 상태 회피 기법의 종류

타임스탬프 순서 기법

장점	• 교착 상태가 없음 – 대기가 없기 때문
단점	• 연쇄 복귀 – T_i의 복귀가 T_j의 복귀를 유발 – T_1이 복귀, T_1이 write한 데이터를 T_2가 벌써 사용했다면 T_2 또한 복귀해야 함 • 순환적 재시작 – 기아(Starvation) 연속적인 복귀와 재시작

1 교착 상태의 발생

- ✓ 트랜잭션이 더 이상 연산을 진행하지 못하는 상태
- ✓ 상호 배제, 대기, 선취 금지, 순환 대기

2 교착 상태의 해결

- ✓ 탐지: 교착 상태를 탐지하고 이를 해결
- ✓ 예방: 교착 상태가 발생되지 않도록 트랜잭션의 순서를 조정
- ✓ 회피: 자원 할당 때 마다 실시간 검사를 통해 교착 상태의 발생을 방지