8장 트랜잭션, 동시성 제어, 회복

데이터베이스응용
2024-11-25(월)
한문석

1

목차

2

동시성제어
(Concurrency Control)

3

학습내용

- 동시성 제어의 개념
- 갱신손실(Lost Update) 문제
- 락(Lock)

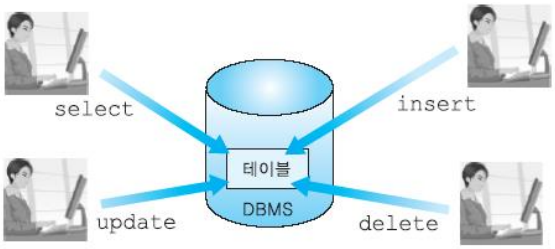
4

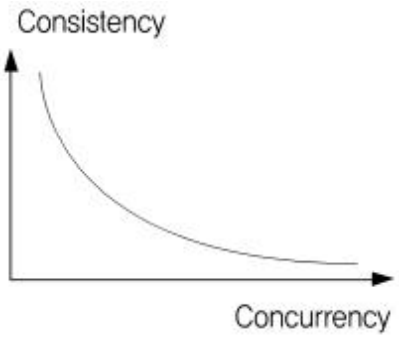
동시성 제어(concurrency control)

- 대부분의 DBMS는 다수 사용자
- 여러 사용자들의 질의나 프로그램 동시 수행 필수
- 트랜잭션이 동시에 수행될 때,
  - 일관성을 해치지 않게 트랜잭션의 데이터 접근을 제어하는 DBMS의 기능
  - 여러 사용자나 프로그램들이 동시에 수행되어도 서로 간섭 못하도록 제어
  - 트랜잭션이 고립되어 수행한 것과 동일한 결과 산출

5

동시성과 일관성의 상관관계





6

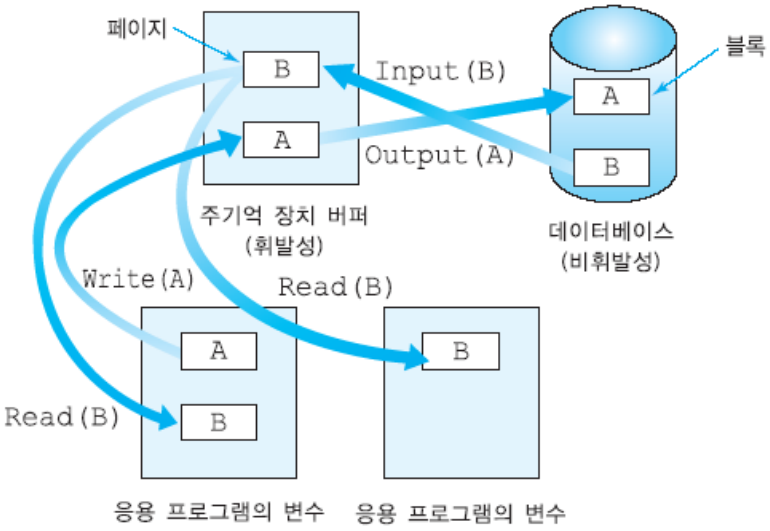
동시성 제어

- 직렬 스케줄(serial schedule)
  - 여러 트랜잭션들의 집합을 한 번에 한 트랜잭션씩 차례대로 수행함
- 비직렬 스케줄(non-serial schedule)
  - 여러 트랜잭션들을 동시에 수행함
- 직렬가능(serializable)
  - 비직렬 스케줄의 결과가 어떤 직렬 스케줄의 수행 결과와 동등함

동시성 제어-데이터베이스 연산

- Input(X) 연산
  - 데이터베이스 항목 X를 포함하고 있는 블록을 주기억장치의 버퍼로 읽어 들임
- Output(X) 연산
  - 데이터베이스 항목 X를 포함하고 있는 블록을 디스크에 기록함
- read_item(X)연산
  - 주기억장치 버퍼에서 데이터베이스 항목 X의 값을 프로그램 변수 X로 복사함
- write_item(X) 연산
  - 프로그램 변수 X의 값을 주기억 치 내의 데이터베이스 항목 X에 기록함

동시성 제어-데이터베이스 연산



동시 수행 시 문제

- 갱신 손실(lost update)
  - 수행 중인 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어 씀으로써 갱신이 무효가 되는 것
- 오손 데이터 읽기(dirty read)
  - 읽기 작업을 하는 트랜잭션 A가 작업을 하는 트랜잭션 B의 중간 데이터를 읽기 때문에 발생하는 문제
  - 완료되지 않은 트랜잭션이 갱신한 데이터를 읽는 것

동시 수행 시 문제

- 반복할 수 없는 읽기(unrepeatable read)
  - 트랜잭션 A가 데이터를 읽고 트랜잭션 B가 데이터를 쓰고(Update) 트랜잭션 A가 다시 한 번 데이터를 읽을 때 생기는 문제
  - 한 트랜잭션이 같은 데이터를 두 번 읽을 때 서로 다른 값을 읽는 것

- 유령 데이터 읽기(Phantom Data Read)
  - 트랜잭션 A가 데이터를 읽고 트랜잭션 B가 데이터를 쓰고(Insert) 트랜잭션 A가 다시 한 번 데이터를 읽을 때 생기는 문제
  - 트랜잭션 A가 읽기 작업을 다시 한 번 반복할 경우 이전에 없던 데이터(유령데이터)가 나타나는 현상

시나리오

- 오손(汚損): 더럽혀지고 손상되다.
- 트랜잭션의 읽기(read)/쓰기(write)

	트랜잭션1	트랜잭션2	발생 문제	동시접근
[상황 1]	읽기	읽기	읽음 (읽기만 하면 아무 문제가 없음)	허용
[상황 2]	읽기	쓰기	• 오손 읽기 • 반복 불가능 읽기(갱신 시) • 유령 데이터 읽기(삽입 시)	허용 혹은 불가 선택
[상황 3]	쓰기	쓰기	갱신손실(절대 허용하면 안 됨)	허용불가(LOCK을 이용)

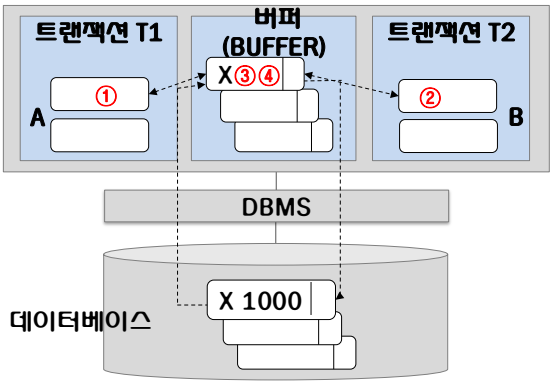
갱신손실 문제

- 두 개의 트랜잭션이 한 개의 데이터를 동시 갱신(update)할 때 발생
- 데이터베이스에서 절대 발생하면 안 되는 현상
- [작업 설명]
  - 한 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업
- [시나리오]
  - 두 개의 트랜잭션이 동시에 작업을 진행
- [문제 발생] 갱신손실
  - T2는 잘못된 데이터로 작업하여 잘못된 결과를 만든 다음 T1의 갱신 작업을 무효화하고 덮쓰기를 수행한 것
  - T1의 갱신이 손실된 **갱신손실(lost update)** 문제 발생

갱신손실 문제 발생 시나리오

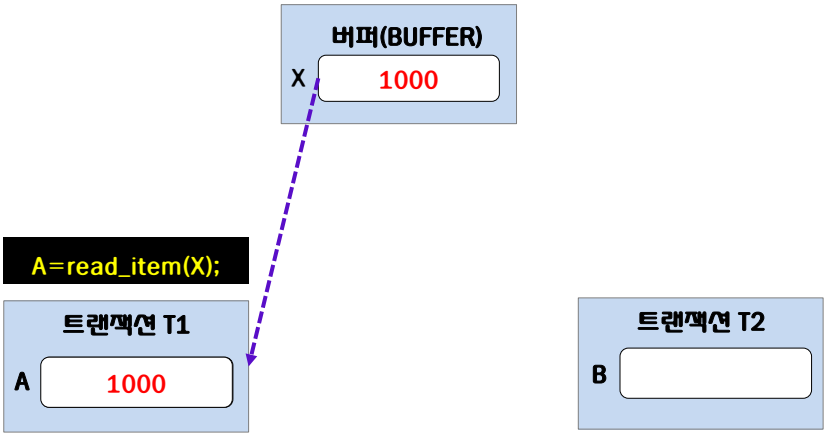
트랜잭션 T1	트랜잭션 T2	버퍼 데이터 값
A=read_item(X); ① A=A−100;		X=1000
	B=read_item(X); ② B=B+100;	X=1000
write_item(A−>X); ③		X=900
	write_item(B−>X); ④	X=1100

T1: 인출, T2: 입금



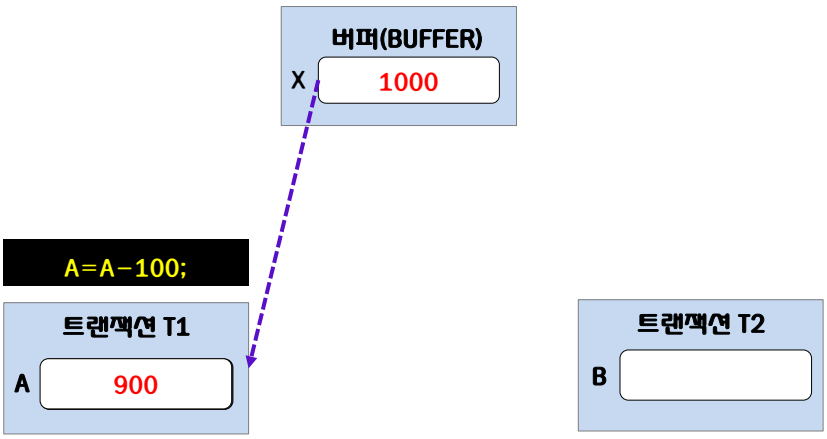
두 트랜잭션 종료 후, X = 1100

갱신손실 문제 발생 시나리오



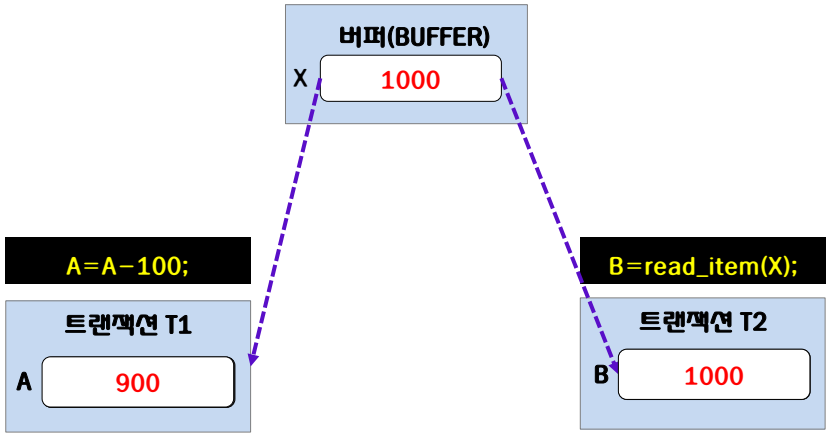
15

갱신손실 문제 발생 시나리오



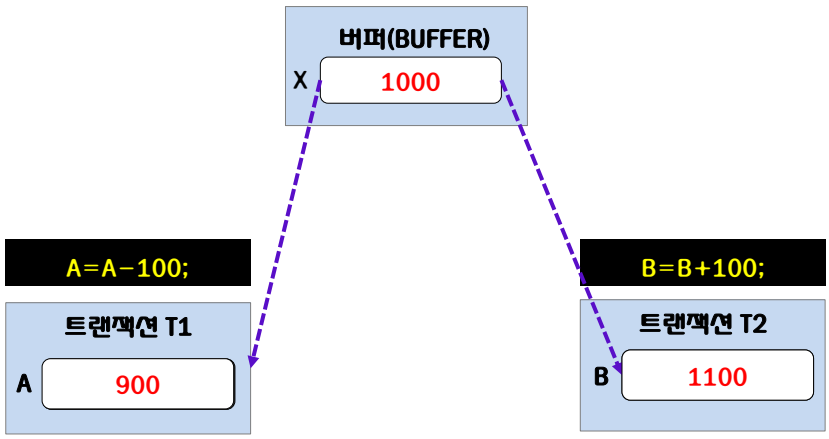
16

갱신손실 문제 발생 시나리오



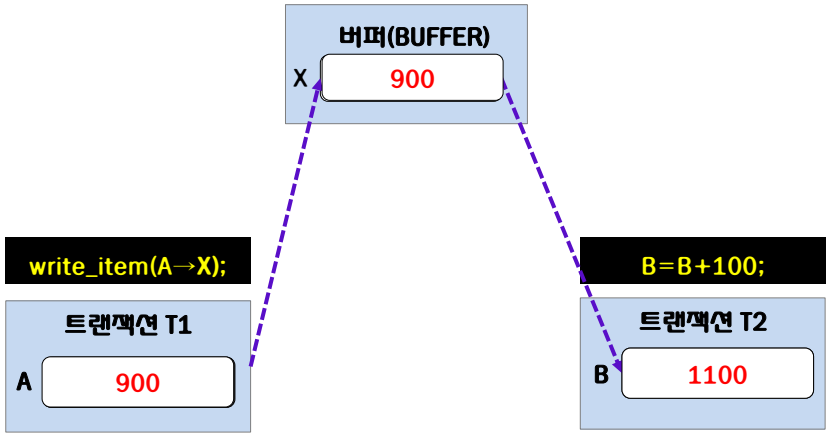
17

갱신손실 문제 발생 시나리오



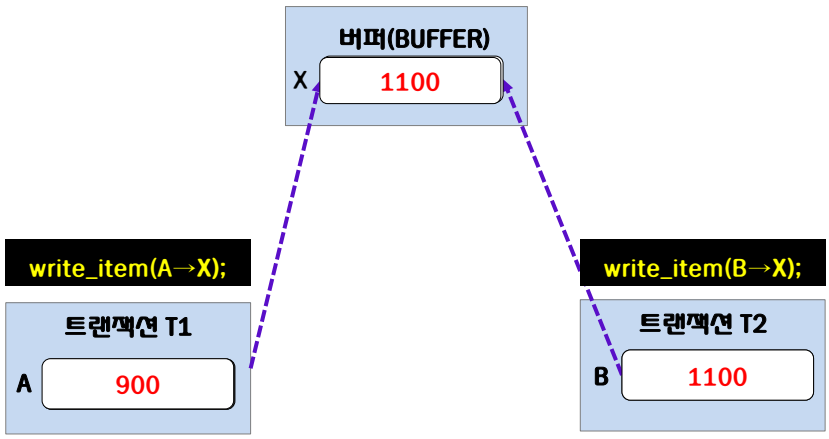
18

갱신손실 문제 발생 시나리오



19

갱신손실 문제 발생 시나리오



20

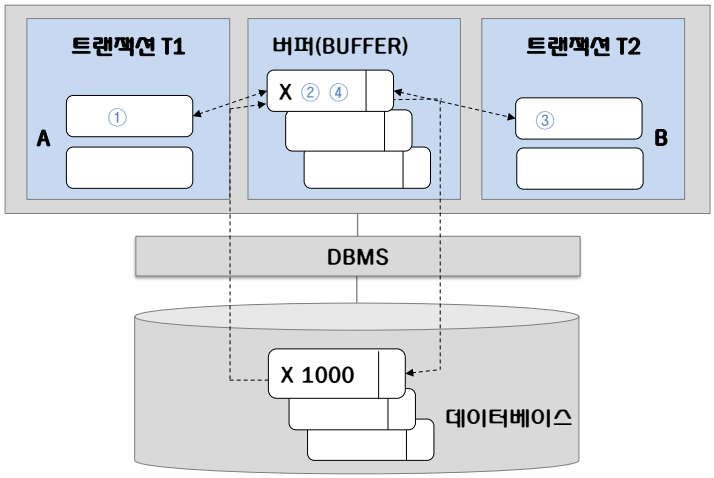
잠금 장치(LOCK)

- 갱신손실 문제를 해결 방법
- DB 내의 각 데이터 항목과 연관된 하나의 변수
- 타 트랜잭션의 데이터 사용 여부 인지 규칙 필요
- 데이터 수정 중이라는 사실 알리는 방법
  - 트랜잭션이 데이터를 읽거나 수정할 때
  - 데이터에 표시하는 잠금 장치

락을 이용한 갱신손실 문제 해결

T1	T2	버퍼 데이터 값
LOCK(X) A=read_item(X); ① A=A−100;		X=1000
	LOCK(X) (wait... 대기)	X=1000
write_item(A−>X); ② UNLOCK(X);		X=900
	B=read_item(X); ③ B=B+100; write_item(B−>X); ④ UNLOCK(X)	X=1000

락을 이용한 갱신손실 문제 해결



락의 기능 예시

[작업 설명] 한 데이터에 두 트랜잭션이 접근, 갱신하는 작업

트랜잭션 T1	트랜잭션 T2
START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=7100 WHERE bookid=1; SELECT * FROM Book WHERE bookid=1; COMMIT;	START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=price+100 WHERE bookid=1; SELECT * FROM Book WHERE bookid=1; COMMIT;

락의 기능 예시

[시나리오] 두 트랜잭션을 동시에 실행

트랜잭션 T1	트랜잭션 T2
SET TRANSACTION NAME 'T1'; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=7100 WHERE bookid=1;	
	SET TRANSACTION NAME 'T2'; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=price+100 WHERE bookid=1;
SELECT * FROM Book WHERE bookid=1; COMMIT;	T1에서 갱신을 위한 락을 걸어 대기 상태 ...(대기 상태)... T1 커밋 후 대기중인 업데이트 실행 재개
	SELECT * FROM Book WHERE bookid=1; COMMIT;

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7100

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7200

락의 유형

- 공유락(LS, shared lock)
  - 트랜잭션이 읽기를 할 때 사용하는 락
  - Read-only lock
- 배타락(LX, exclusive lock)
  - 트랜잭션이 읽고 쓰기를 할 때 사용하는 락

공유락과 배타락 사용 규칙

- 데이터에 락이 걸려있지 않으면 트랜잭션은 데이터에 락을 걸 수 있음
  - 트랜잭션이 데이터 X를 읽기만 할 경우 LS(X)를 요청
  - 읽거나 쓰기를 할 경우 LX(X)를 요청
- 다른 트랜잭션이 데이터에 LS(X)을 걸어둔 경우
  - LS(X)의 요청은 허용하고 LX(X)는 허용하지 않는다.
- 다른 트랜잭션이 데이터에 LX(X)을 걸어둔 경우
  - LS(X)와 LX(X) 모두 허용하지 않음
- 트랜잭션이 락을 허용 받지 못하면 대기 상태가 됨

락 호환행렬(Compatibility Matrix)

- 트랜잭션 락의 허용 관계 표현
  - 공유락은 상호 허용
  - 배타락은 허용 불허

요청 \ 상태	LS 상태	LX 상태
LS 요청	허용	대기
LX 요청	대기	대기

2단계 락킹(2 phase locking)

- 두 개의 트랜잭션이 동시에 실행될 때
  - 락을 걸고 해제하는 시점에 제한을 두지 않으면
  - 데이터의 일관성이 깨질 수 있음
    - 이를 방지하는 방법: 2단계 Locking
- 확장단계(Growing phase, Expanding phase)
  - 트랜잭션이 필요한 락을 획득하는 단계
  - 이미 획득한 락을 해제하지 않음
- 수축단계(Shrinking phase)
  - 트랜잭션이 락을 해제하는 단계
  - 새로운 락을 획득하지 않음

2단계 락킹(2 phase locking) 문제

- [작업 설명]
  - 두 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업
- [문제 발생]
  - 락을 사용하되 2단계 락킹 기법을 사용하지 않을 경우[교재 456쪽]
- [문제 해결]
  - 2단계 락킹 기법을 사용할 경우[교재457쪽]

작업설명

- 트랜잭션 T1, T2
  - T1: 예금이체, T2: 이자계산
- 계좌 A, B
  - 모두 잔고가 1000
- T1은 A계좌에서 100을 인출, B에 입금
- T2는 두 계좌 잔고를 10% 증가 시킴
- T1 ⇒ T2, T2 ⇒ T1 모두 A+B=2200

문제 발생

락을 사용하되 2단계 락킹 기법을 사용하지 않을 경우

트랜잭션 T1(예금이체)	트랜잭션 T2(이자계산)	계좌 A, B 값(각각 1000)
LX(A) t1=read_item(A); t1=t1−100; A=write_item(t1); UN(A) /*UnLock*/		A=900 B=1000

	LX(A)
	t2=read_item(A);
	t2=t2*1.1;
	A=write_item(t2);
	UN(A) /*UnLock*/
	LX(B)
	t2=read_item(B);
	t2=t2*1.1;
	B=write_item(t2);
	UN(B) /*UnLock*/
		A=990
		B=1100

LX(B)
t1=read_item(B);
t1=t1+100;
B=write_item(t1);
UN(B) /*UnLock*/
		A=990
		B=1200
		/* A+B=2190 이므로 일관성 제약조건에 위배됨 */

문제 해결

2단계 락킹 기법을 사용할 경우

트랜잭션T1	트랜잭션T2	A, B 값
LX(A) t1=read_item(A); t1=t1−100; A=write_item(t1);		A=900 B=1000
	LX(A) …(대기상태)…	

LX(B)
t1=read_item(B);
t1=t1+100;
B=write_item(t1);
UN(A)
UN(B)
		A=900
		B=1100

	LX(A)
	t2=read_item(A);
	t2=t2*1.1;
	A=write_item(t2);
	LX(B)
	t2=read_item(B);
	t2=t2*1.1;
	B=write_item(t2);
	UN(A)
	UN(B)
		A=990
		B=1210
		/* A+B=2200 이므로 일관성 제약 조건을 지킴 */

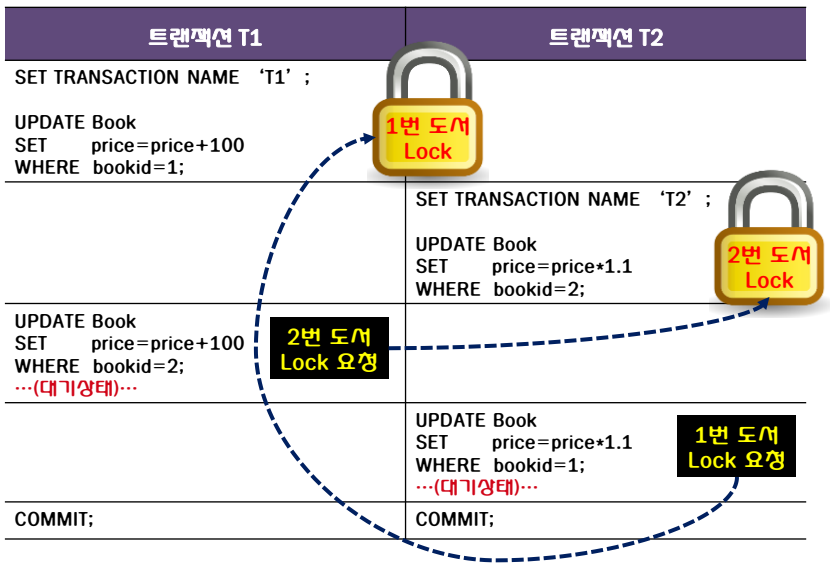
교착상태(deadlock)

- 개념
  - 두 개 이상의 트랜잭션이 각각 자신의 데이터에 대하여 락을 획득
  - 상대방 데이터에 대하여 락을 요청하면 무한 대기 상태에 빠질 수 있는 현상
  - 교착상태라고도 함
- [작업 설명]
  - 두 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업

교착상태(deadlock)

- [문제 발생]
  - MySQL DBMS에서 데드락 발생[교재 458쪽]
- [문제 해결]
  - 데드락 해결
  - 일반적으로 데드락이 발생하면
    - DBMS는 T1 혹은 T2의 작업 중 하나를 강제로 중지
  - 그 결과 나머지 트랜잭션은 정상적으로 실행
  - 이때 중지시키는 트랜잭션에서 변경한 데이터는 원래 상태로 복구

교착상태 문제 발생



교착상태 해결

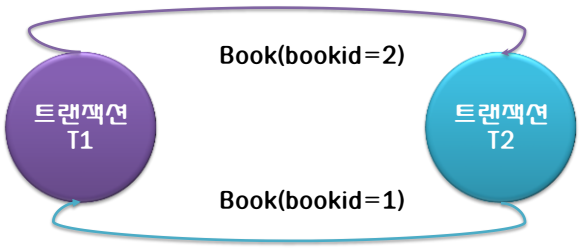
- 교착상태 발생 시
  - DBMS는 T1 또는 T2 작업 중 하나 강제 중지
  - 나머지 트랜잭션 정상 실행
  - 중지 트랜잭션이 변경한 데이터 원상복구
- T2 MySQL 오류보고

  Error Code: 1213. Deadlock found when trying to get lock: try restarting transaction

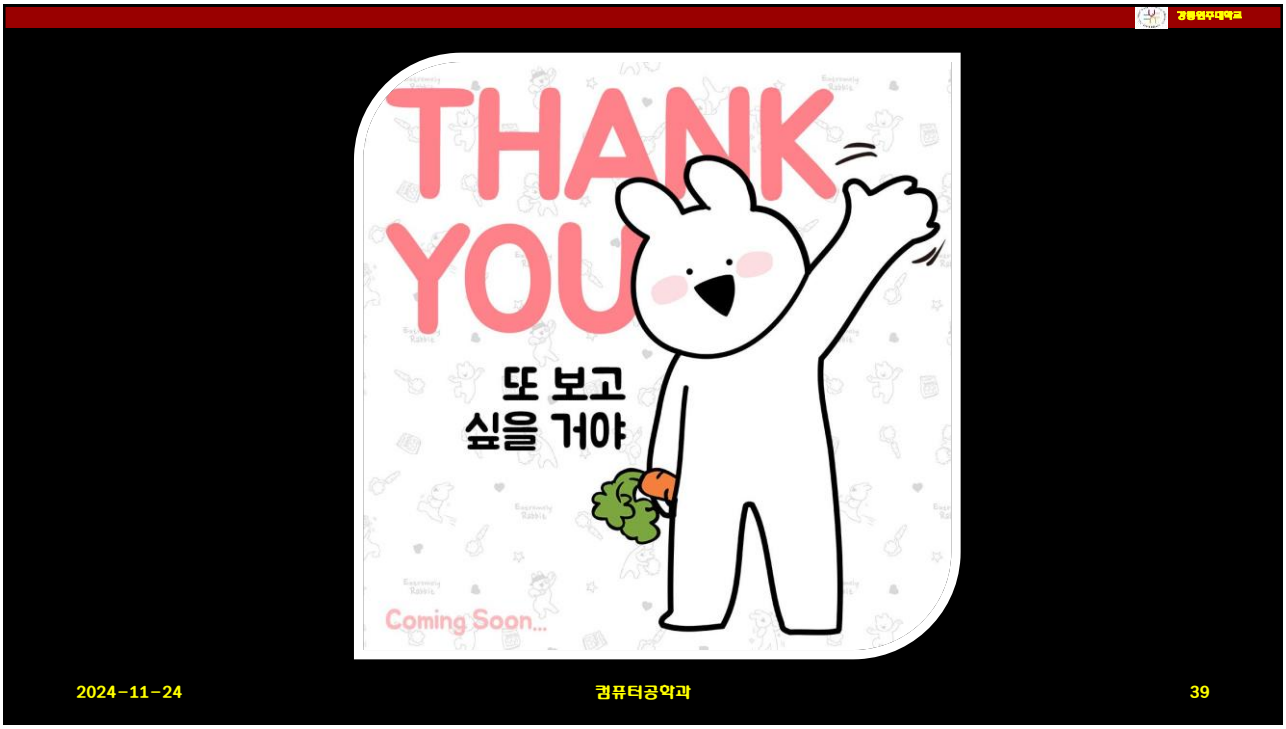
37

교착상태 해결

- 교착상태 발생 시
  - DBMS는 T1 또는 T2 작업 중 하나 강제 중지
  - 나머지 트랜잭션 정상 실행
  - 중지 트랜잭션이 변경한 데이터 원상복구
- 교착상태 대기 그래프(wait-for graph)



38



39