8장 트랜잭션(1)

데이터베이스 실험실

[목 차]

1. 트랜잭션 개념

1.1. 트랜잭션과 프로그램의 차이점

트랜잭션(transaction)은 DBMS에서 데이터베이스를 다루는 하나의 논리적인 작업이다.

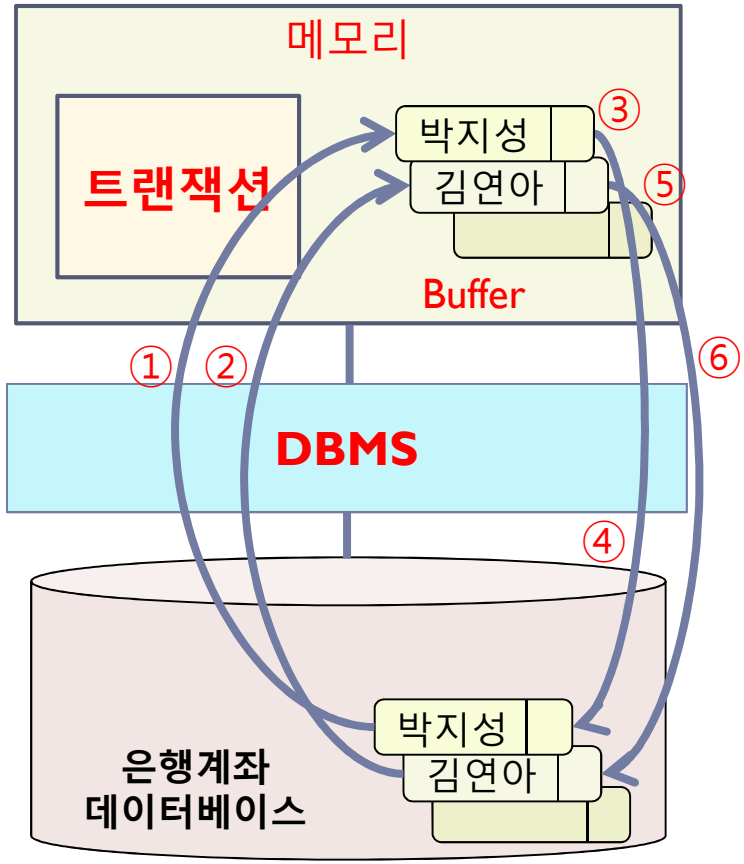
- 데이터베이스에서 트랜잭션을 정의하는 이유

1. 데이터베이스에서 데이터를 다루는 작업이 일어날 때 장애가 일어나는 경우가 있다. 이때 장애에서 올바르게 데이터를 복구하는 작업의 단위를 트랜잭션으로 삼는다.
2. 데이터베이스에서 프로그램이 동시에 실행될 때 프로그램들을 서로 분리하는 단위이다.

```
BEGIN TRANSACTION
(1) A 계좌(박지성)의 금액 인출 SQL UPDATE 문
(2) B 계좌(김연아)의 금액 입금 SQL UPDATE 문
COMMIT TRANSACTION
```

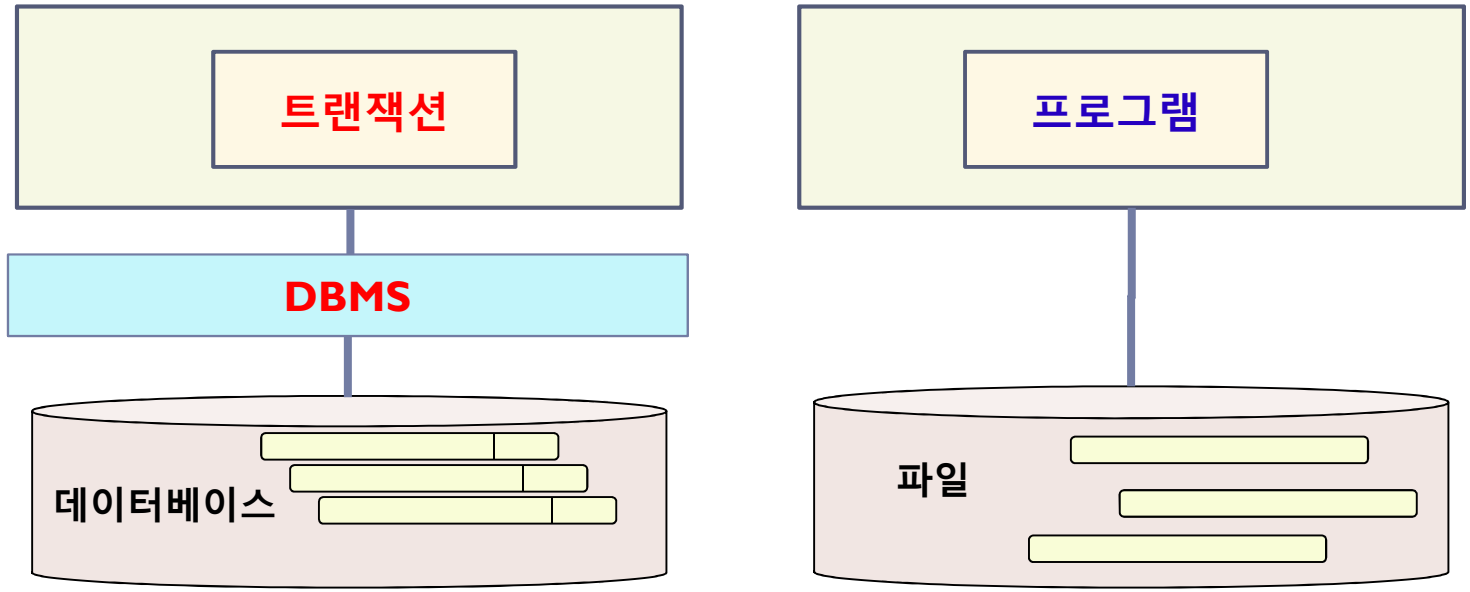
- 트랜잭션의 예

은행의 계좌이체 프로그램
BEGIN TRANSACTION ① /* 박지성 계좌를 읽어온다 */ ② /* 김연아 계좌를 읽어온다 */ /* 잔고 확인 */ ③ /* 예금인출 홍길동 */ UPDATE customer SET Balance=Balance-10000 WHERE customername='홍길동'; ④ /* 홍길동 계좌를 기록한다 */ ⑤ /* 예금입금 김연아 */ UPDATE customer SET Balance=Balance+10000 WHERE customername='김연아'; ⑥ /* 김연아 계좌를 기록한다 */ COMMIT END TRANSACTION



- 트랜잭션과 프로그램의 차이

	트랜잭션	프로그램
구조	BEGIN TRAN … END TRAN	main() { … } 구조이다.
데이터	데이터베이스 저장된 데이터를 다룬다	데이터를 다루거나 다루지 않을 수 있다 데이터를 다룰 때는 파일에 저장된 데이터를 다룬다.
번역기	DBMS가 번역한다.	컴파일러가 번역한다.
성질	원자성, 일관성, 고립성, 지속성 (ACID)	



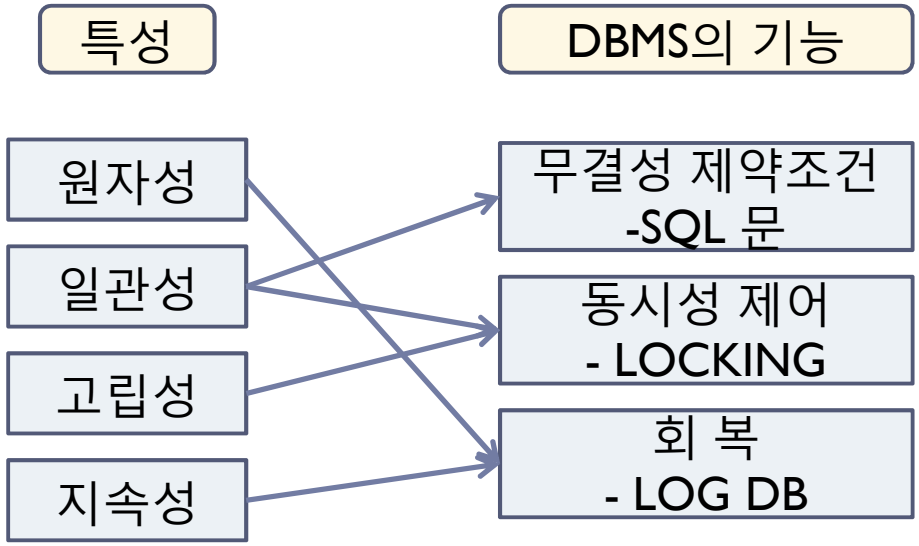
1.2. 트랜잭션이 갖추어야 할 성질(ACID)

(1) **원자성(Atomicity)** : 트랜잭션과 관련된 작업들이 모두 수행되었는지 아니면 모두 실행이 안되었는지를(All or Nothing) 보장하는 능력이다.

(2) **일관성(Consistency)** : 트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 유지하는 것을 의미한다.

(3) **고립성(Isolation)** : 트랜잭션을 수행 시 다른 트랜잭션의 연산 작업이 끼어들지 못하도록 보장하는 것을 의미한다.

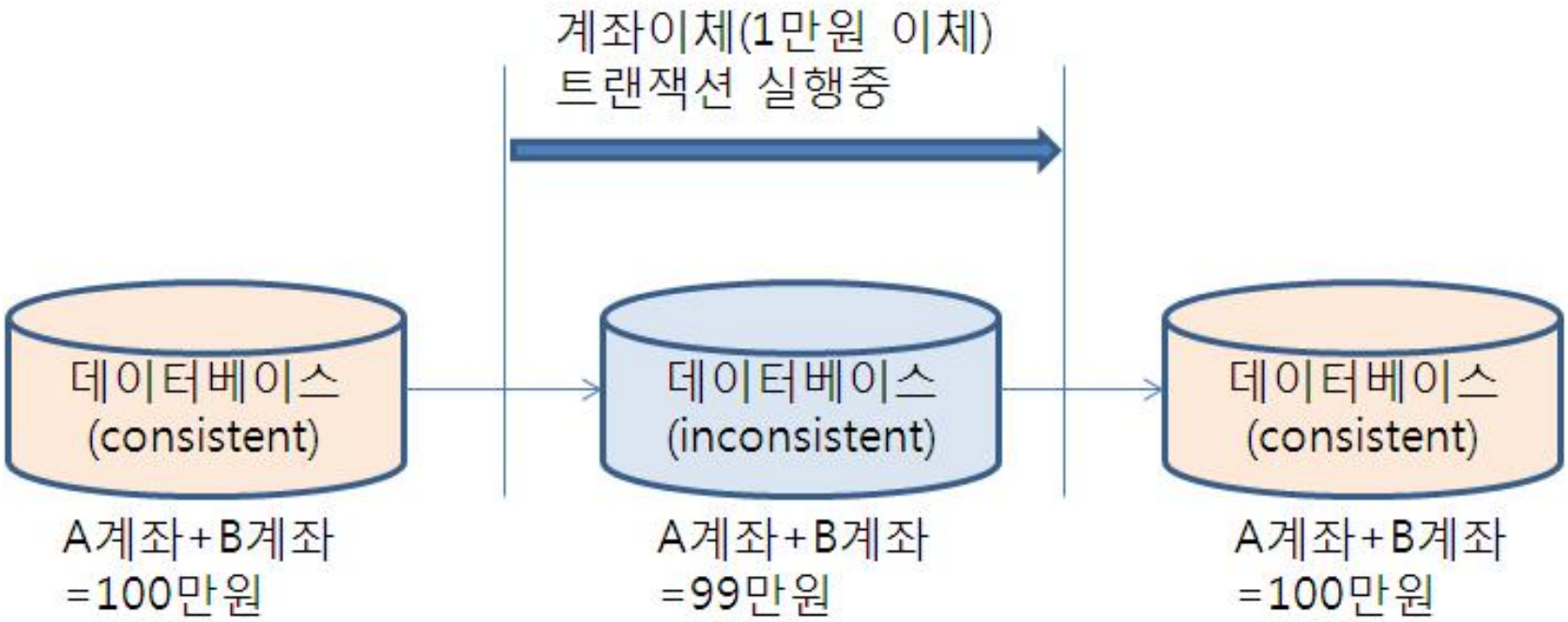
(4) **지속성(Durability)** : 성공적으로 수행된 트랜잭션은 영원히 반영되어야 함을 의미한다.



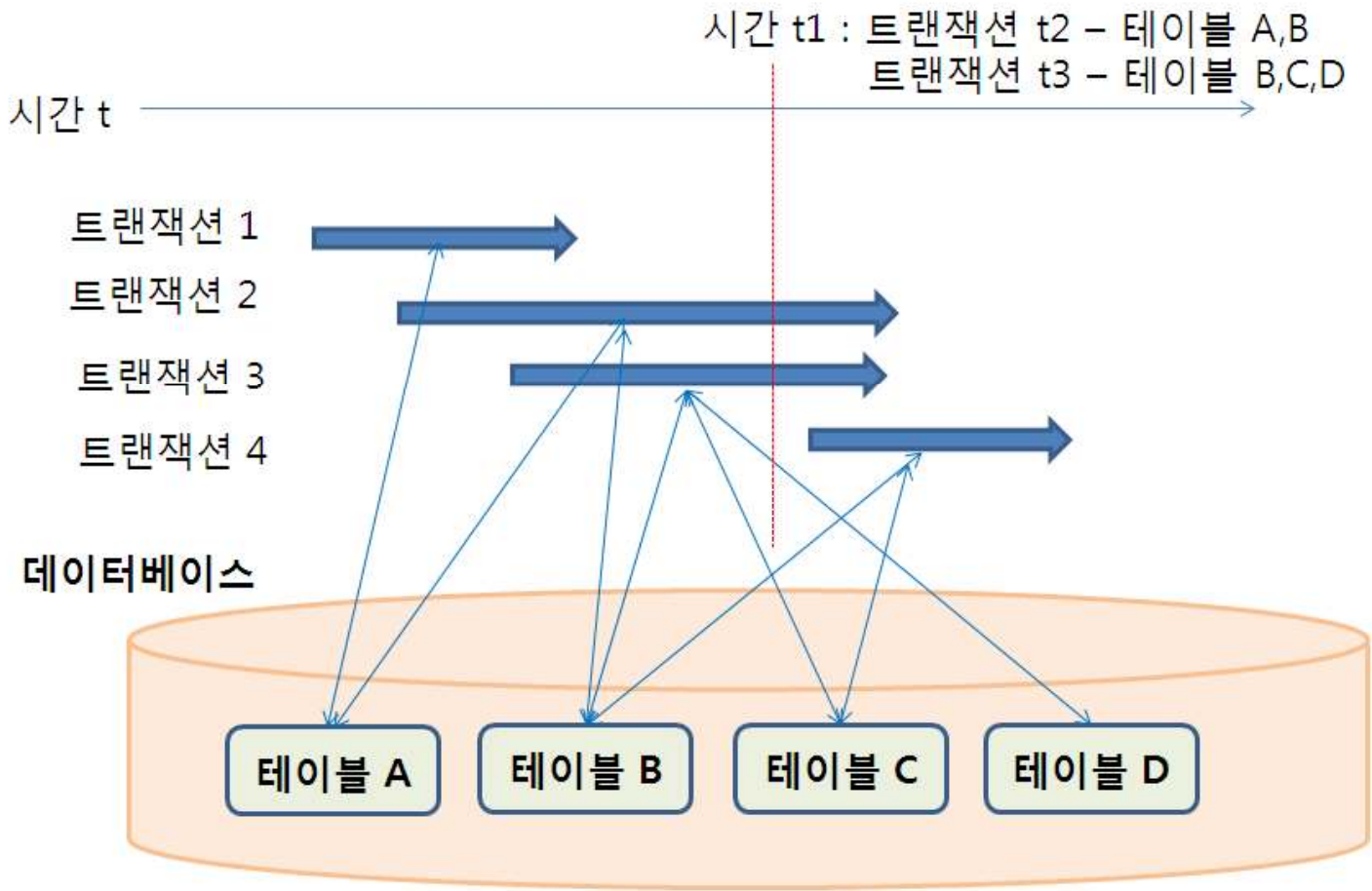
1.2.1. 원자성

```
BEGIN TRANSACTION
insert into emp (empno,ename,sal) values (109,'Sami',3000);
savepoint a;
insert into dept values (50,'Sales','Hyd');
savepoint b;
insert into salgrade values (6,9000,12000);
rollback to a;
/* Then row from salgrade table and dept will be roll backed. Now you can commit the row inserted into emp table or rollback the transaction. */
rollback to b;
/* Then row inserted into salgrade table will be roll backed. Now you can commit t he row inserted into dept table and emp table or rollback to savepoint a or comple tely roll backed the transaction. */
rollback;
/* Then the whole transactions is roll backed. */
commit;
/* Then the whole transaction is committed and all savepoints are removed.*/
```

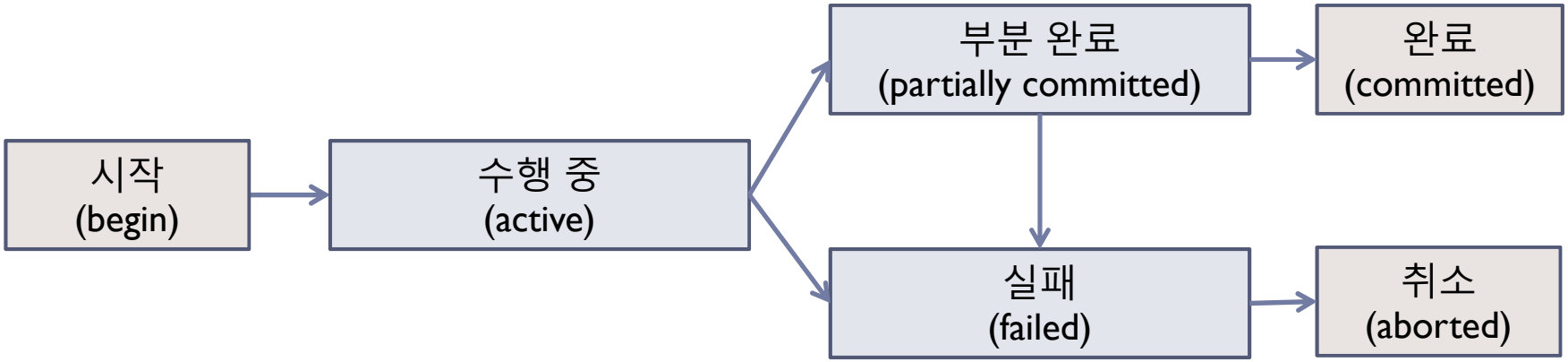
1.2.2. 일관성


계좌이체(1만원 이체)
트랜잭션 실행중
데이터베이스
(consistent)
데이터베이스
(inconsistent)
데이터베이스
(consistent)
A계좌+B계좌
=100만원
A계좌+B계좌
=99만원
A계좌+B계좌
=100만원

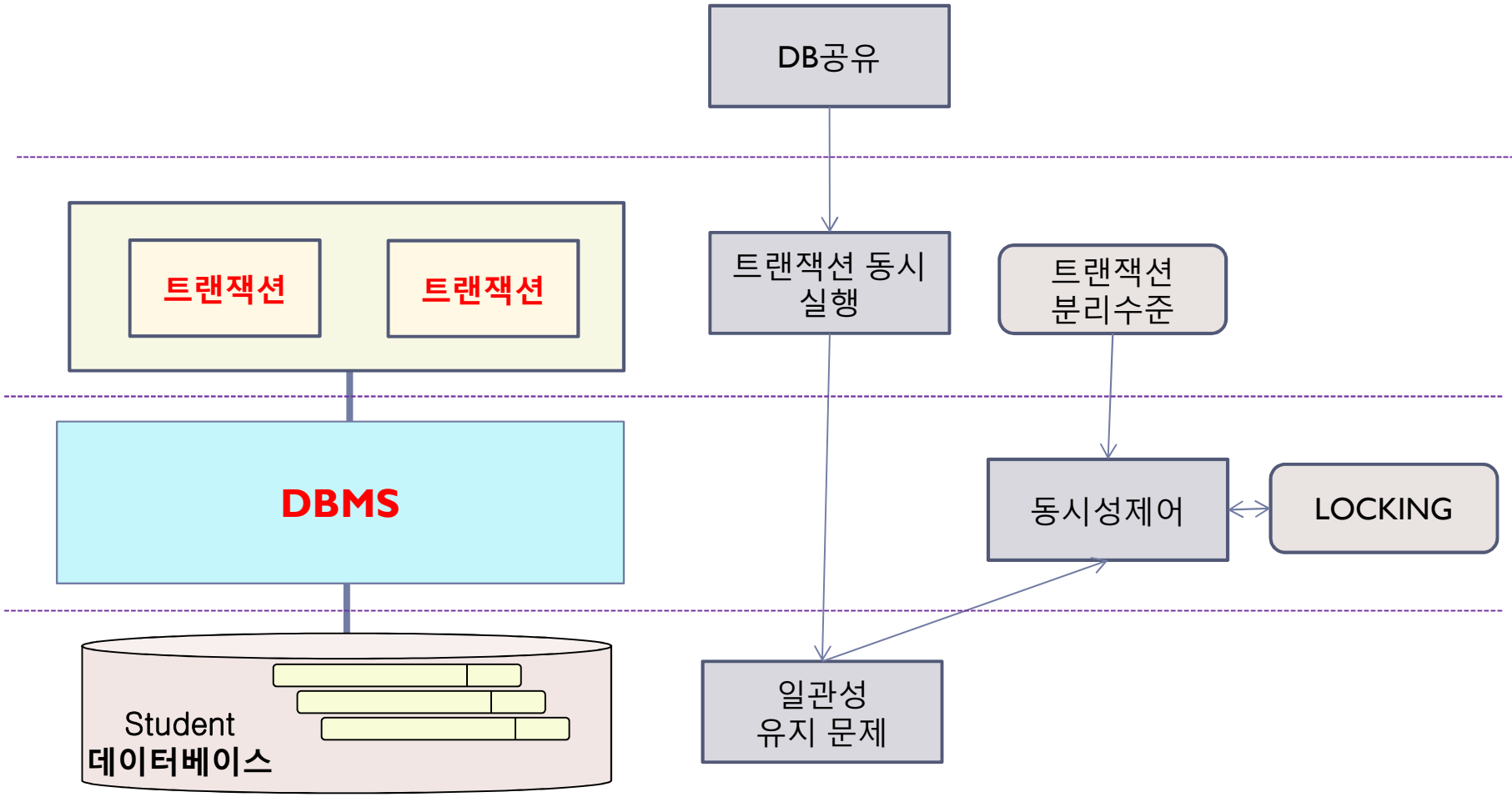
1.2.3. 고립성


시간 t1 : 트랜잭션 t2 – 테이블 A,B
트랜잭션 t3 – 테이블 B,C,D
시간 t
트랜잭션 1
트랜잭션 2
트랜잭션 3
트랜잭션 4
데이터베이스
테이블 A
테이블 B
테이블 C
테이블 D

1.2.4. 지속성

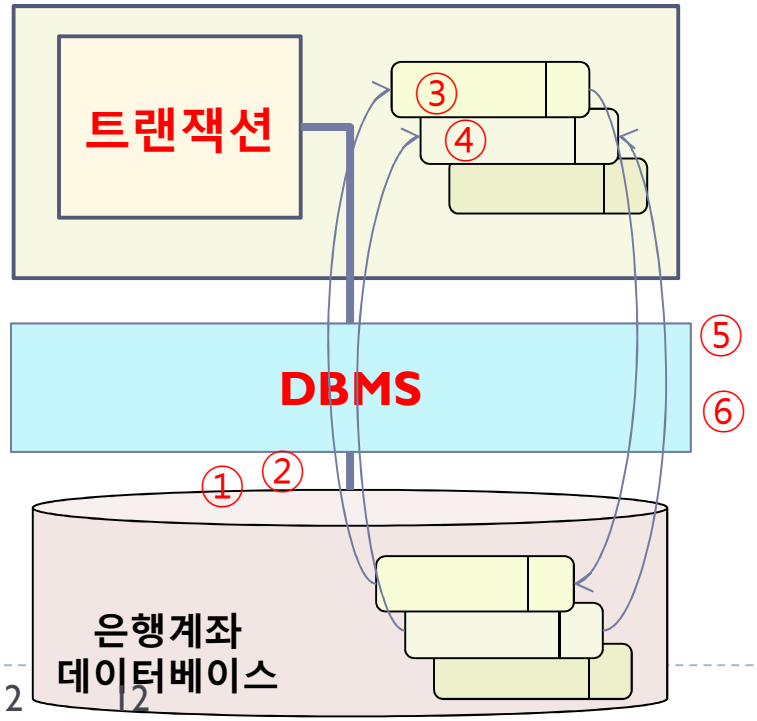

시작
(begin)
수행 중
(active)
부분 완료
(partially committed)
완료
(committed)
실패
(failed)
취소
(aborted)

2. 동시성제어


DB공유
트랜잭션
트랜잭션
트랜잭션 동시 실행
트랜잭션 분리수준
DBMS
동시성제어
LOCKING
Student
데이터베이스
일관성 유지 문제

1.3. 트랜잭션 수행 시 문제
1.3.1. 트랜잭션 1개, 테이블 1개 경우

상태	트랜잭션 1개, 테이블 1개
발생가능문제점과 해결	트랜잭션이 중간에 멈추었을 때 데이터가 문제가 있을 수 있으면 처음상태로 돌아간다. 1. 트랜잭션이 중간에 문제점이 있어서 스스로 멈춰야 할 경우가 있다. => ROLLBACK 해야 함 2. 외부시스템의 문제로 멈춰야 하는 경우가 있다 => RECOVERY 필요함



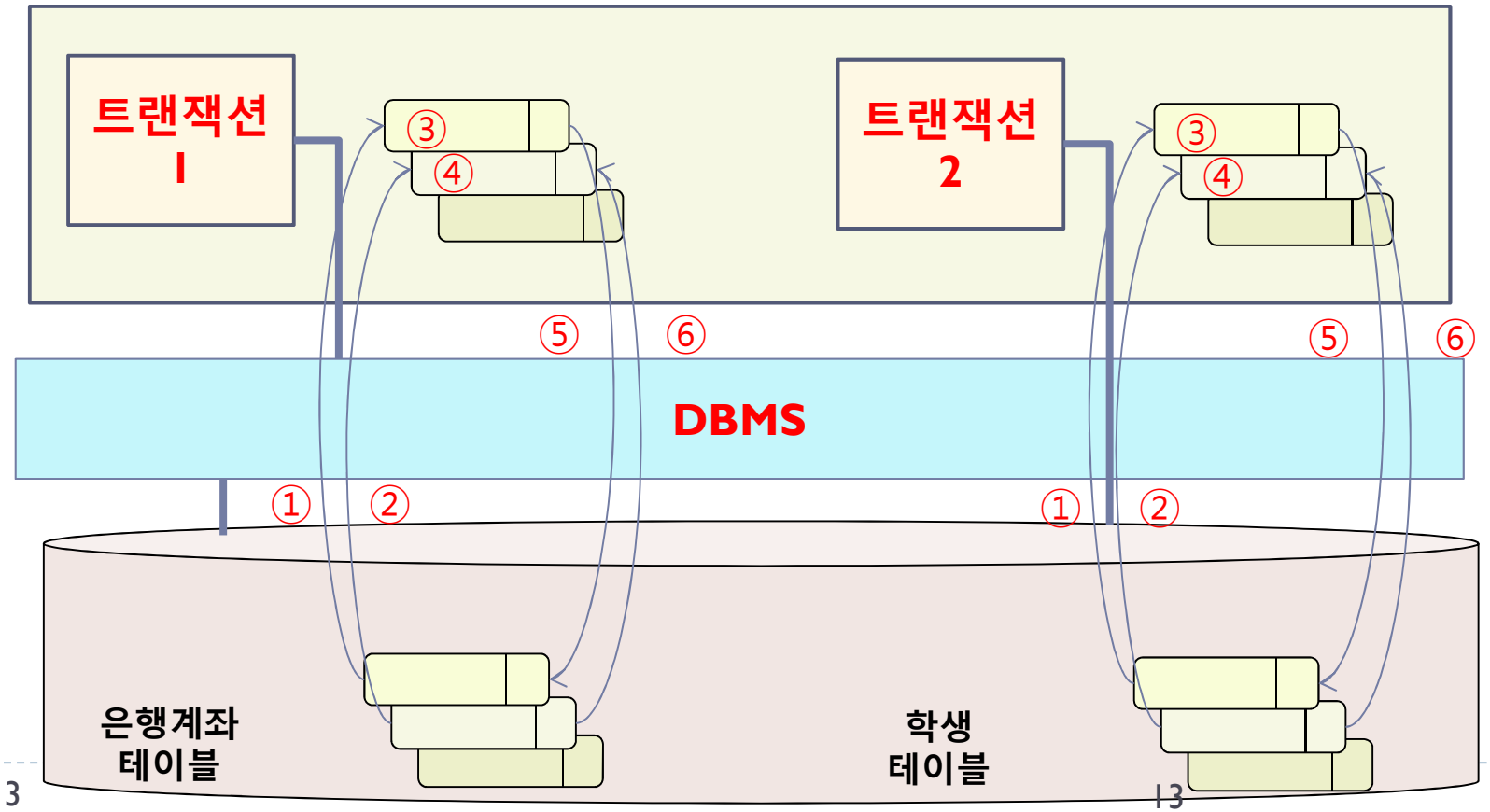
은행의 계좌이체 프로그램

```
BEGIN TRAN
①②
UPDATE customer     ③
SET Balance=Balance-10000
WHERE customername='홍길동';


UPDATE customer ④
SET Balance=Balance-10000
WHERE customername='홍길동';

⑤⑥

COMMIT

END TRAN
```


12

1.3.2. 트랜잭션 2개, 테이블 2개 각각 경우

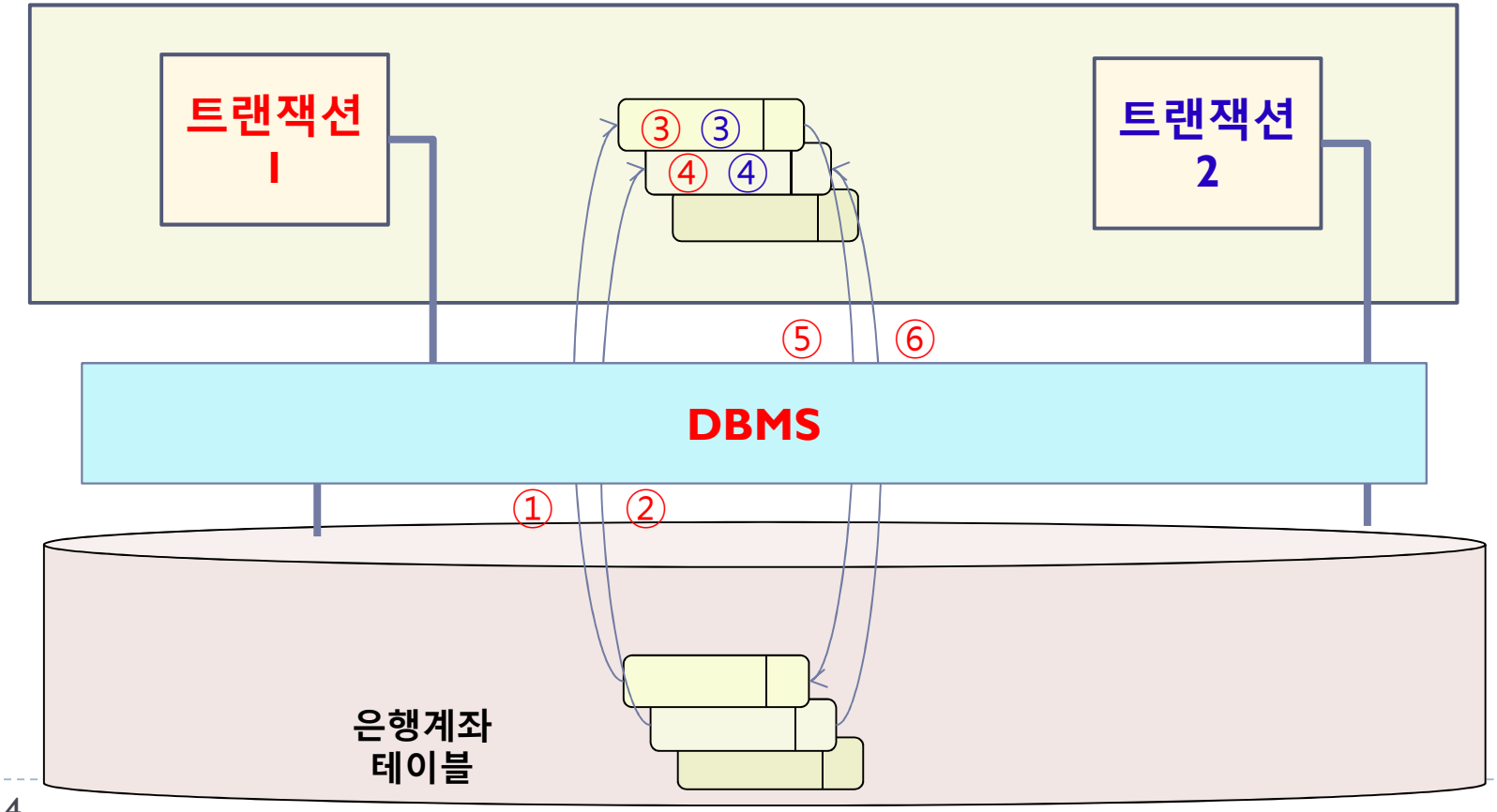
상태	트랜잭션 2개, 테이블 2개 각각
발생가능문제점과 해결	트랜잭션이 각각의 테이블을 다룰 경우 문제점이 없다.(트랜잭션 외부 문제 제외)



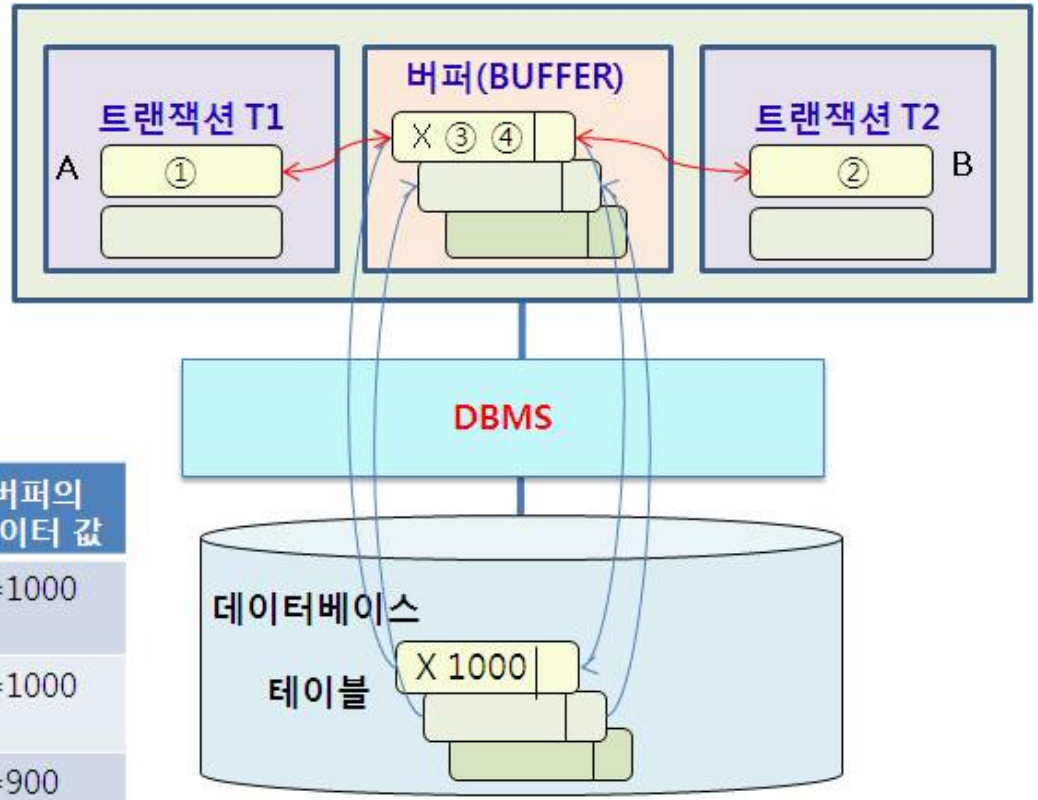
13

1.3.3. 트랜잭션 2개, 테이블 1개

상태	트랜잭션 2개, 테이블 1개
발생가능문제점과 해결	트랜잭션이 같은 테이블을 다룰 경우 문제점이 많다.(트랜잭션 외부 문제 제외) => LOCK을 사용하여 해결한다.



2. 트랜잭션과 동시성제어
2.1. 갱신손실

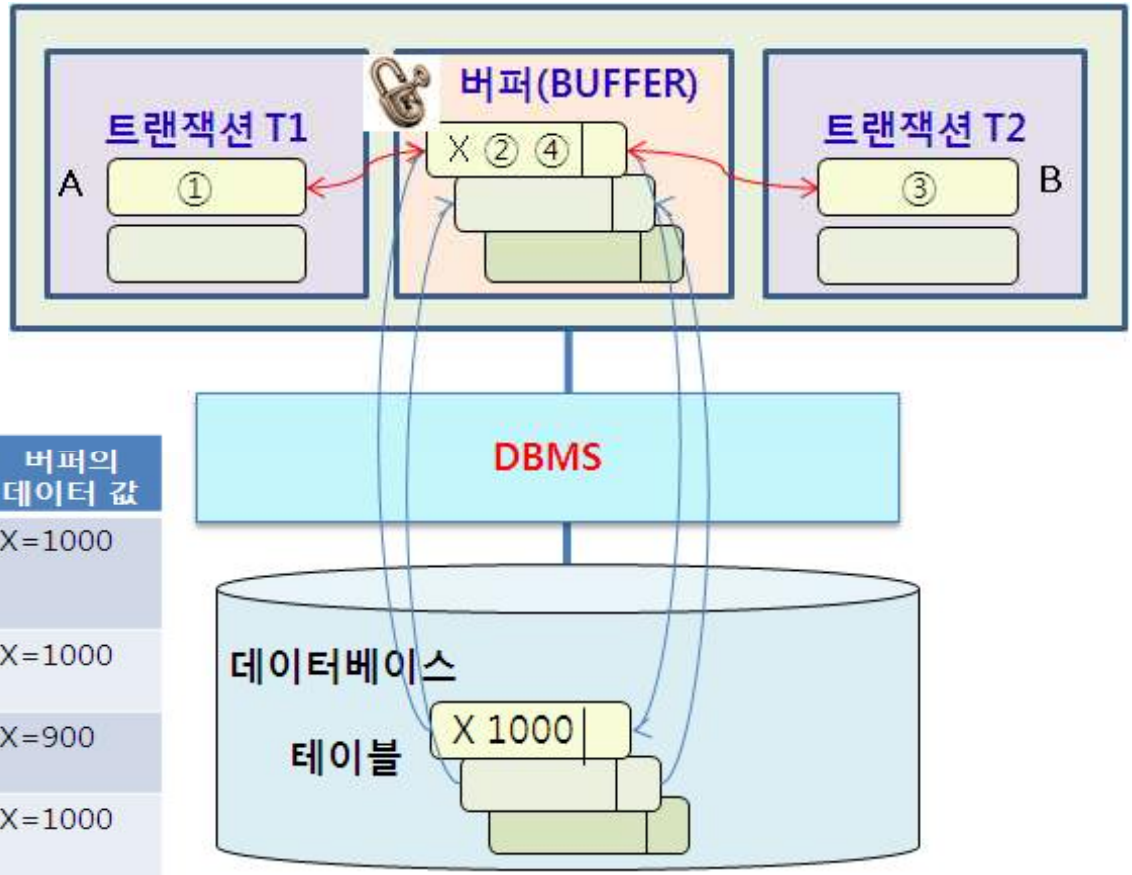


트랜잭션 T1	트랜잭션 T2	버퍼의 데이터 값
A=read_item(X); ① A=A-100;		X=1000
	B=read_item(.X); ② B=B+100;	X=1000
write_item(A->.X); ③		X=900
	write_item(B->X); ④	X=1100

2.2. LOCK

2.2.1 LOCK 개념

트랜잭션 T1	트랜잭션 T2	버퍼의 데이터 값
LOCK(X); A=read_item(X); ① A=A-100;		X=1000
	LOCK(X) (wait ,,,)	X=1000
write_item(A->.X); ② UNLOCK(X);		X=900
	LOCK(X) B=read_item(.X); ③ B=B+100; write_item(B->X); ④ UNLOCK(X)	X=1000



2.2.2 LOCK의 유형

(1) 데이터를 읽을 경우 표시하는 LOCK ⇒ 공유락(Shared LOCK)
(2) 수정할 때 사용하는 LOCK ⇒ 배타락(Exclusive LOCK)

- LOCK을 사용하는 규칙
(1) 트랜잭션은 데이터 X를 읽을 경우 공유락 LS(X)를 요청하고,
쓰거나 혹은 읽고 쓸 경우 배타락 LX(X)를 요청한다.
(2) 데이터가 아무 LOCK이 걸려있지 않으면, 트랜잭션이 락을 걸려고 할 경우 허용한다.
(3) 데이터에 락이 걸려있는 경우, 트랜잭션이 LS(X)를 요청할 경우, LS(X)가 걸려있는 경우
허용하고, 나머지 경우는 LOCK을 허용하지 않는다.
(4) 트랜잭션이 LOCK을 허용받지 못하면 그 트랜잭션은 대기상태가 된다.

요청 \ 상태	LS	LX
LS	허용	대기
LX	대기	대기

- LOCK의 실험 (두 트랜잭션 모두 write를 시도한 경우)

	1	2

	BEGIN TRAN
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
	
	select empno, ename, job, sal
	from emp
	where empno=7900;
	
	update emp
	set Sal=1000
	where empno = 7900;	

		BEGIN TRAN select empno, ename, job, sal from emp where empno=7900;
	select empno, ename, job, sal from emp where empno=7900;	
		update emp set Sal=900 where empno = 7900; .... (waiting)
	ROLLBACK	
		....(실행 진행)

2.2.3 2단계 락킹

T1	T2	

LX(A)
read_item(A);
A=A+1;
write_item(A);
UN(A)
		A=1
		B=1

	LX(A)
	read_item(A);
	A=A*2;
	write_item(A);
	UN(A)
	LX(B)
	read_item(B);
	B=B*2;
	write_item(B);
	UN(B)	

LX(B)
read_item(B;
(B=B+1;
write_item(B);
UN(B)		

T1	T2	

LX(A)
read_item(A);
A=A+1;
write_item(A);
		A=1
		B=1

	LX(A) --- 대기…	

LX(B)
read_item(B;
(B=B+1;
write_item(B);
 	
UN(A)
UN(B)		

2.2.4 DEADLOCK

이렇게
진행되면?

	1	2

	BEGIN TRAN
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
	select empno, ename, job, sal
	from emp
	where empno=7900;
	update emp
	set Sal=1000
	where empno = 7900;	

		BEGIN TRAN
		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
		select empno, ename, job, sal
		from emp
		where empno=7902;
		update emp
		set Sal=1000
		where empno = 7902;

	update emp set Sal=1000 where empno = 7902; .... (waiting)	
		update emp set Sal=1000 where empno = 7900; .... (waiting)