**제 9장 트랜잭션**

**9.1 트랜잭션 개요**

DB의 일관된 상태를 다른 일관된 상태로 바꿔주는 DB 프로그램

중간에 오류가 생겼을 경우 원래 상태로 돌아가야 하고, (recovery)

변화가 완료될 경우 상태가 지속되어야 한다.

**트랜잭션의 특성 (ACID)** -> CIDA! 시다!

**Atomicity (원자성)**: 쪼개지지 않는다. All or Nothing

보장해주기 위한 회복 모듈이 존재한다.

**Consistency (일관성):** 무결성과 관련 → 일관된 상태 유지

잘못된 것이 없다! ex) 은행의 총액 유지

**Isolation (고립성)**: 한 트랜잭션이 수행 중일 때는 다른 트랜잭션 접근X

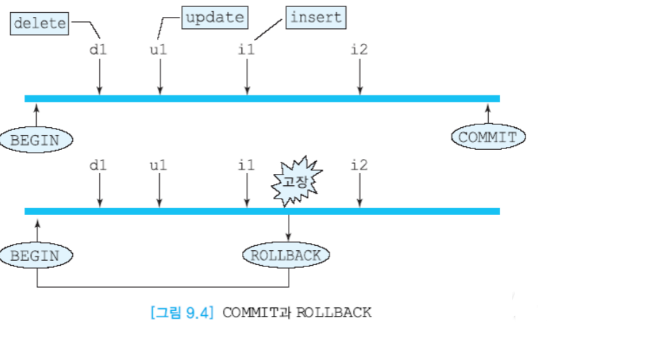
동시에 수행되더라도 그 결과는 순차수행과 같아야 함.

별도의 고립 수준 지정

**Durability (지속성)**: 한 트랜잭션이 완료되면 그 후에 손실X

회복 모듈은 시스템이 다운되더라도 지속성 보장.

**트랜잭션 완료**: 트랜잭션에서 변경하려는 내용이 DB에 완전하게 반영

 COMMIT WORK

**트랜잭션 철회**: 트랜잭션이 수행되기 전 상태로 되돌림

ROLLACK WORK

**9.2 동시성 제어**

여러 사용자들이 다수의 트랜잭션을 동시에 수행할 때, 트랜잭션 간의 간섭X

직렬 스케줄: 한 트랜잭션씩 차례대로 수행.

비 직렬 스케줄: 여러 트랜잭션 동시 수행.

직렬가능: 비 직렬 스케줄의 결과가 어떤 직렬 스케줄의 수행 결과와 동등함.

**데이터베이스 연산**

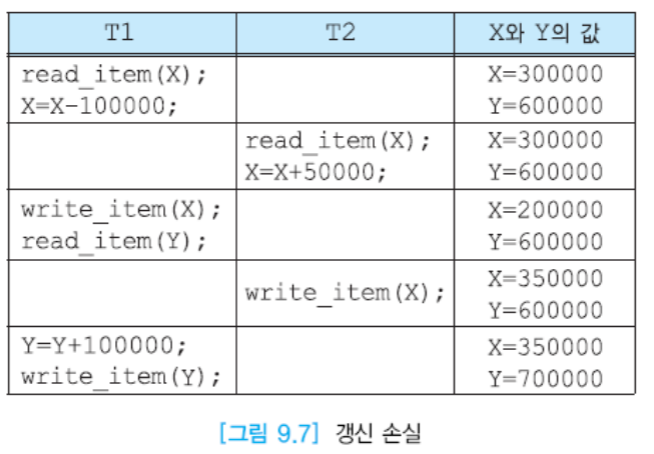
Input(X): 항목 X를 포함하고 있는 블록을 주기억 장치의 버퍼로 읽어 들임

Output(X): 항목 X를 포함하고 있는 블록을 디스크에 기록함

read\_item(X): 주기억 장치 버퍼에서 항목 X의 값을 프로그램 변수 X로 복사함

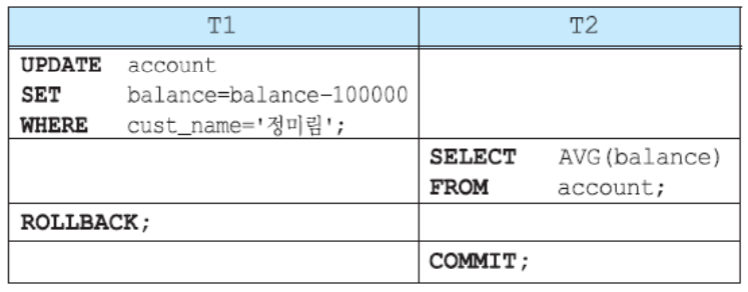
write\_item(X): 프로그램 변수 X의 값을 주기억 장치 내의 항목 X에 기록함

**동시성 제어를 하지 않았을 때 발생하는 문제**

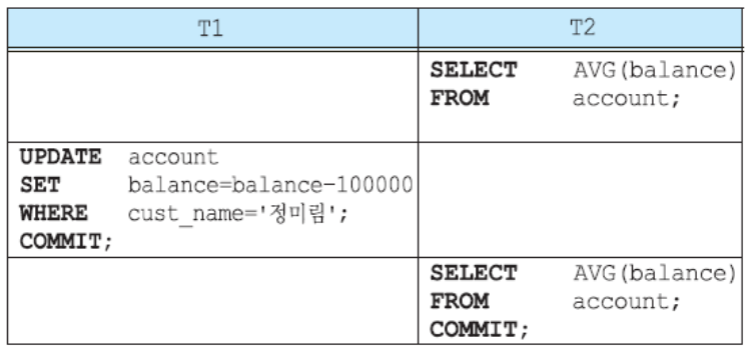


**갱신 손실**: 수행 중인 트랜잭션이 갱신한 내용을

다른 트랜잭션이 덮어 씀으로써 갱신이 무효가 됨.

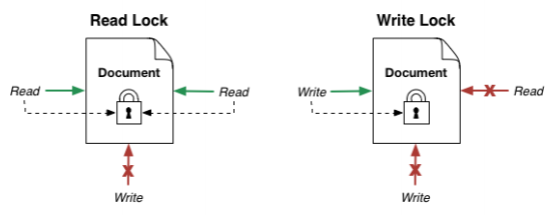


**오손 데이터 읽기**: 트랜잭션이 중간에 갱신한 데이터(잘못된 데이터)를 읽음.



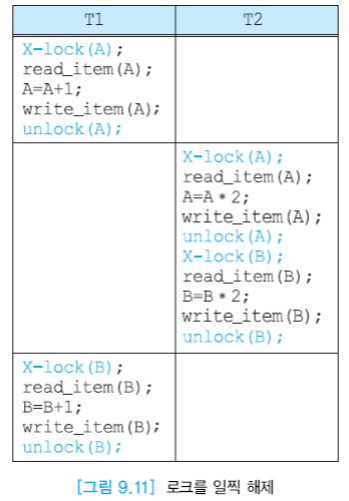
**반복할 수 없는 읽기**: 트랜잭션이 같은 데이터를 다시 읽을 때 다른 값을 읽음.

**로킹(locking)**: 트랜잭션에서 데이터 항목을 접근할 때 로크를 요청하고, 접근을 끝낸 후에 로크를 해제함. 요청한 로크에 대한 정보는 lock table에 저장



**독점 로크(X-lock, eXclusive lock**): 갱신 목적 접근

**공유 로크(S-lock, Shared lock)**: 판독(읽기) 목적 접근

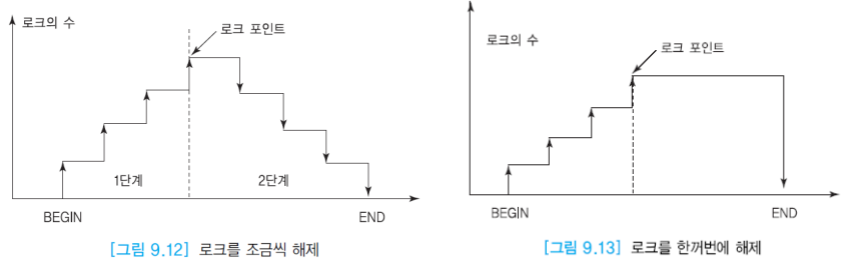


오손 데이터 읽기 발생

A는 증가된 상태(갱신된 데이터)의 2배

B는 증가되지 않은 상태의 2배

**2단계 로킹 프로토콜(2-phase locking protocol)**



로크를 요청하는 것과 로크를 해제하는 것이 2단계로 이루어짐

1. **로크 확장 단계**: 로크를 요청하는 단계, 로크 해제 불가
2. **로크 수축 단계**: 로크를 해제하는 단계, 새로운 로크 요청 불가

한번에 해제 또는 서서히 해제 가능

1. **로크 포인트(lock pont)**: 트랜잭션에서 필요로 하는 로크를 걸어놓은 시점.

**데드록(deadlock)**: 서로 상대방이 보유하고 있는 로크를 요청하면서 대기.

**다중 로크 단위(multiple granularity)**

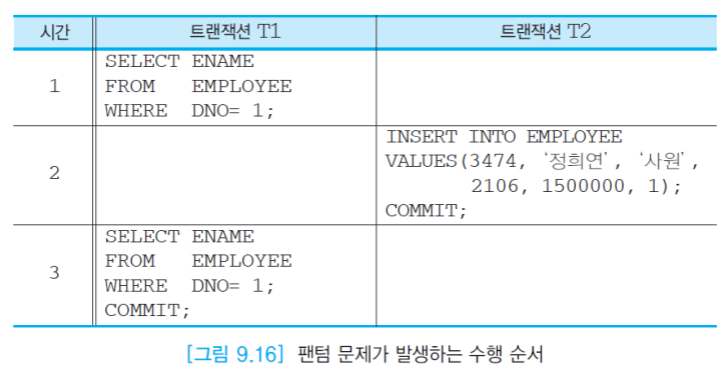
한 트랜잭션에서 로크할 수 있는 데이터 항목이 두 가지 이상 존재

로크할 수 있는 단위로는 데이터베이스, 릴레이션, 디스크 블록, 투플 등

작을수록 오버헤드 증가, 동시성 정도 증가

**팬텀 문제**

T1의 select문의 수행 결과가 다르게 나타난다.



**9.3 회복**

필요성: 트랜잭션 수행 중 시스템 다운에 대하여 원자성과 지속성 보장

**재수행(REDO)**: 완료 명령을 수행했었다면 재 수행하여 지속성 유지

**취소(UNDO):** 수행 도중 이였다면 갱신 사항을 취소하여 원자성 보장

**안전 저장 장치**: 모든 유형의 고장을 견딜 수 있는 저장 장치

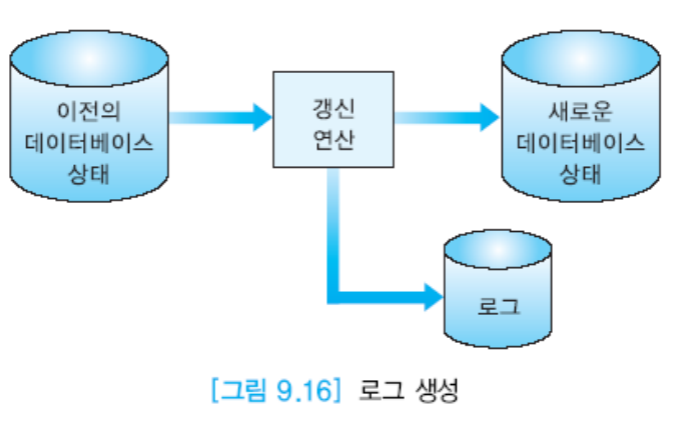
비 휘발성 저장 장치에 두 개 이상의 사본을 중복해서 저장해서 구현

**재해적 고장**: 디스크 손상 → 데이터베이스를 백업해 놓은 자기 테이프로 회복

**비재해적 고장**: 대부분의 고장 → 여러 회복 알고리즘

Ex) 로그를 사용한 즉시 갱신 및 지연 갱신, 그림자 페이징 등

**로그를 사용한 즉시 갱신**



원자성과 지속성을 보장하기 위해 DBMS는 **로그(log)**라고 부르는 파일을 유지함

영향을 미치는 모든 트랜잭션의 연산들에 대해서 로그 레코드를 기록함

로그는 일반적으로 안전 저장 장치에 저장됨.

각 로그 레코드마다 트랜잭션 ID를 포함시키고,

동일한 로그 레코드는 연결 리스트로 유지.

각 로그 레코드는 **로그 순서 번호(LSN**: Log Sequence Number)로 식별됨

**\* 이중 로그**: 로그를 두 개의 디스크에 중복해서 저장하는 것.

\* **트랜잭션의 완료점**: 갱신 연산이 모두 끝나고 로그에 기록되었을 때.

\* WAL(Wrte-Ahead Logging): 데이터 베이스 버퍼보다 로그 버퍼를 먼저 디스크에 기록. 즉, 로그가 중요하다!

**체크 포인트 (checkpoint)**

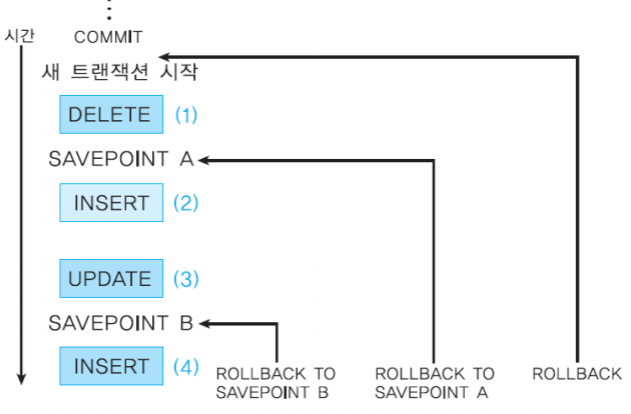
회복 시 재 수행할 트랜잭션의 수를 줄이기 위해서 주기적으로 체크포인트 수행.

체크포인트 시점에는 버퍼 내용이 디스크에 강제로 기록

= 디스크 상에서 로그와 데이터베이스의 내용 일치

작업이 끝나면 로그에 checkpoint 로그 레코드 기록

**9.4 PL/SQL의 트랜잭션**



**COMMIT**: 현재의 트랜잭션에서 결과를 모두 반영하고 트랜잭션을 완료.

**ROLLBACK**: 현재의 트랜잭션 결과를 모두 되돌리고 트랜잭션을 철회.

**SAVEPOINT**: 현재의 트랜잭션 내에 저장 점을 표시

트랜잭션을 더 작은 부분으로 나눔

**ROLLBACK TO SAVEPOINT:** 지정된 저장 점 이후에 갱신된 내용만 되돌림

**고립 수준**

**READ UNCOMMITTED**: 모든 질의에 로크 걸지 않음. 갱신에만 독점로크를 건다.

**READ COMMITTED**: 읽으려는 데이터에 로크를 걸고 로크를 해제함.

**REPEATABLE READ**: 범위에 해당하는 데이터 UPDATE 불가능

나머지 범위에 해당하는 row는 insert 가능

**SERIALIZBLE**: 데이터 수정/입력 불가능

