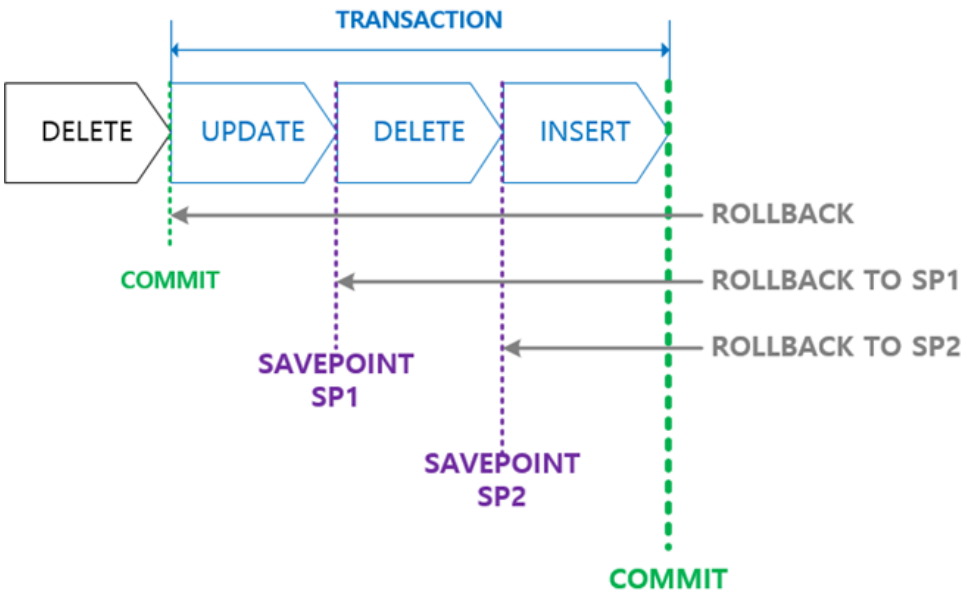
Transaction

Transaction

- DBMS에서 데이터를 다루는 논리적인 작업의 단위
- DB에서 데이터를 다룰 때 장애가 일어난 경우 데이터를 복구하는 작업의 단위가 된다.
- DB에서 여러 작업이 동시에 같은 데이터를 다룰 때가 이 작업을 서로 분리하는 단위가 된다.
- ex) A 계좌에서 B계좌로 송금
    - A에서 돈을 빼고 B에서 돈을 더하는 2가지의 Update문을 하나의 트랜잭션으로 묶으면 1개의 SQL만 실행되는 상황은 발생하지 않음

트랜잭션 제어 명령어 (TCL, TRANSACTION CONTROL LANGUAGE)



- **COMMIT**
    - 하나의 트랜잭션이 성공적으로 끝났고, DB가 일관성 있는 상태일 때 이를 알려주는 연산
- **ROLLBACK**
    - 트랜잭션의 실행 중에 장애가 발생한 경우에 수행
    - 트랜잭션이 행한 모든 연산을 취소시키거나 트랜잭션을 재시작함
- **SAVEPOINT**
    - 현재 트랜잭션을 작게 분할하는 명령어
    - 저장된 SAVEPOINT는 `ROLLBACK TO SAVEPOINT` 문을 사용해서 표시한 곳까지 `ROLLBACK` 가능

ACID

- **원자성**(Atomicity) - 트랜잭션이 DB에 모두 반영되거나, 혹은 전혀 반영되지 않아야 된다. all or nothing
- **일관성**(Consistency) - 트랜잭션의 작업 처리 결과는 항상 일관성 있어야 한다.
- **독립성**(Isolation) - 둘 이상의 트랜잭션이 동시에 병행 실행되고 있을 때, 어떤 트랜잭션도 다른 트랜잭션 연산에 끼어들 수 없다.
- **지속성**(Durability) - 트랜잭션이 성공적으로 완료되었으면, 결과는 영구적으로 반영되어야 한다. 트랜잭션이 완료되어 저장 된 DB는 저장 후에 생기는 장애, 오류 등에 영향을 받지 않아야 한다.

동시성 제어

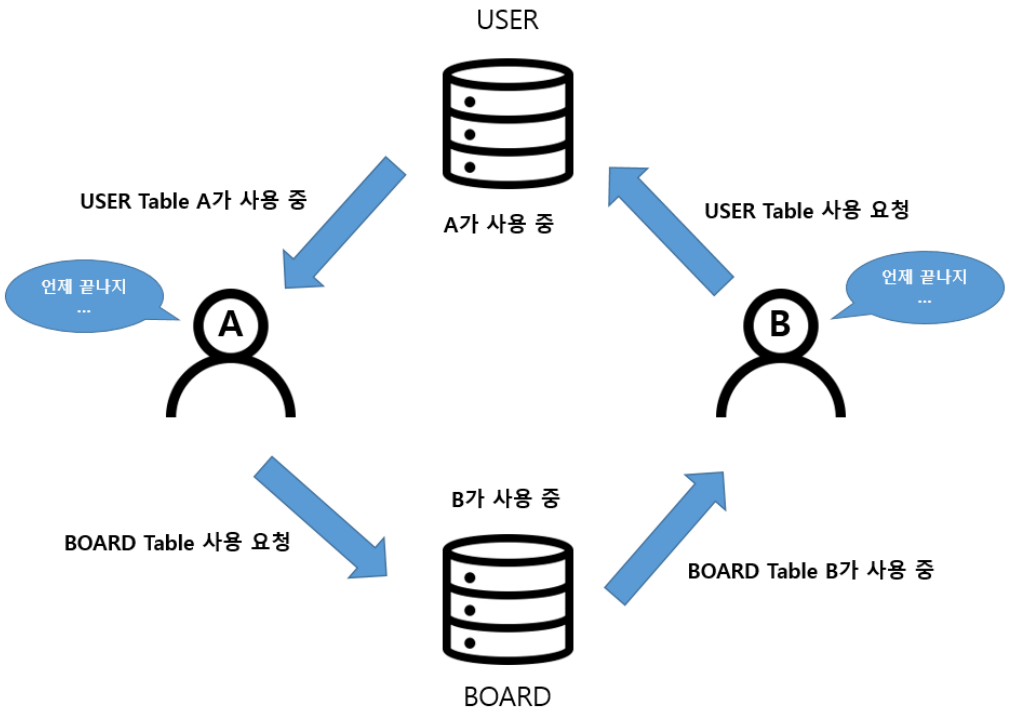
	트랜잭션1	트랜잭션2	발생 문제	동시접근
[상황 1]	읽기	읽기	읽음(읽기만 하면 아무 문제가 없음)	허용
[상황 2]	읽기	쓰기	오손 읽기, 반복불가능 읽기, 유령 데이터 읽기	허용 혹은 불가 선택
[상황 3]	쓰기	쓰기	갱신손실, 모순성, 연쇄 복구	허용불가(LOCK을 이용)

- 다중 사용자 환경에서 둘 이상의 트랜잭션이 동시에 수행될 때, 일관성을 해치지 않도록 트랜잭션의 데이터 접근 제어
- 다중 사용자 환경을 지원하는 DBMS의 경우, 반드시 지원해야 하는 기능
- **갱신손실** - 하나의 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어씀으로써 갱신이 무효화가 되는 것
- 모순성 - 다른 트랜잭션들이 해당 항목 값을 갱신하는 동안 한 트랜잭션이 두 개의 항목 값 중 어떤 것은 갱신되기 전의 값을 읽고 다른 것은 갱신된 후의 값을 읽게 되어 데이터의 불일치가 발생하는 상황
- 연쇄 복구 - 두 트랜잭션이 동일한 데이터 내용을 접근할 때 발생, 한 트랜잭션이 데이터를 갱신한 다음 실패하여 Rollback 연산을 수행하는 과정에서 갱신과 Rollback 연산을 실행하고 있는 사이에 해당 데이터를 읽어서 사용할 때 발생할 수 있는 문제
- **해결법** : Lock을 걸어 다른 트랜잭션이 접근하지 못하게 막음, unlock 되기 전까지 다른 트랜잭션은 접근하지 못하고 기다려야 함

  Lock
  - 트랜잭션들이 동일한 데이터 항목에 대해 임의적인 병행 접근을 하지 못하도록 제어하는 것
  - 트랜잭션 T가 데이터 항목 X에 대해 Read(X) or Write(X)연산을 수행하려면 반드시 lock(X) 연산을 해주어야 함
  - 트랜잭션 T가 실행한 lock(X)에 대해서는 해당 트랜잭션이 종료되기 전에 반드시 unlock(x)연산을 해주어야 함
  - 트랜잭션 T는 다른 트랜잭션에 의해 이미 lock이 걸려 있는 X에 대해 다시 lock(X)를 수행시키지 못한다.
  - 트랜잭션 T가 X에 lock을 걸지 않았다면, unlock(X)를 수행시키지 못한다.
  - **Shared Lock (공유 lock, Read Lock)**: 보통 **데이터를 읽을 때** 사용하는 lock
    - Shared lock의 경우, 같은 데이터에 여러 개의 shared lock을 설정 가능
    - Shared lock이 설정된 데이터에 대해 exclusive lock을 설정 불가
  - **Exclusive Lock (배타 lock, Write Lock)**: 보통 **데이터를 변경할 때** 사용하는 lock
    - Exclusive lock이 해제될 때까지 다른 트랜잭션은 해당 데이터에 접근 불가
    - 다른 트랜잭션이 수행되고 있는 데이터에 대해 exclusive lock 설정 불가

Deadlock (교착상태)

- 두 개 이상의 트랜잭션이 각각 자신의 데이터에 대하여 락을 획득하고 상대방 데이터에 대하여 락을 요청하면 무한 대기 상태에 빠질 수 있는 현상



교착상태 해결 방법

[방지 기법]

1. MySQL의 경우, isolation level를 READ-COMMITED로 변경해 교착상태 발생 횟수를 줄일 수 있음
    - READ-COMMITED로 변경할 경우, 쿼리문 단위로 exclusive lock을 걸어 트랜잭션 진행 중이더라도 exclusive lock이 걸린 데이터가 아니라면 다른 트랜잭션이 접근해 작업 가능
2. LOCK_TIMEOUT 시간을 설정해 lock이 지속할 수 있는 최대 시간 설정
    - 해당 시간 지나면 트랜잭션 및 쿼리 취소

[회피 기법]

- 자원 할당 시, 트랜잭션의 시간 스탬프(Time Stamp)를 활용해 교착 상태를 회피하는 방법

1. **Wait-Die 방식**: 다른 트랜잭션이 lock을 건 데이터에 대해서 접근하려고 할 때,
    - 현재 트랜잭션이 먼저 실행된 트랜잭션이라면 대기
    - 현재 트랜잭션이 나중에 실행된 트랜잭션이라면 포기, 나중에 재요청
2. **Wound-Wait 방식:** 다른 트랜잭션이 lock을 건 데이터에 대해서 접근하려고 할 때,
    - 현재 트랜잭션이 먼저 실행된 트랜잭션이라면 선점(빼앗기)
    - 현재 트랜잭션이 나중에 실행된 트랜잭션이라면 대기

[낙관적 병행 제어 기법]

- 트랜잭션이 실행되는 동안에는 검사 X
- 트랜잭션이 다 실행된 이후에 검사해서 문제가 있다면 되돌리고, 문제가 없어야 커밋 가능