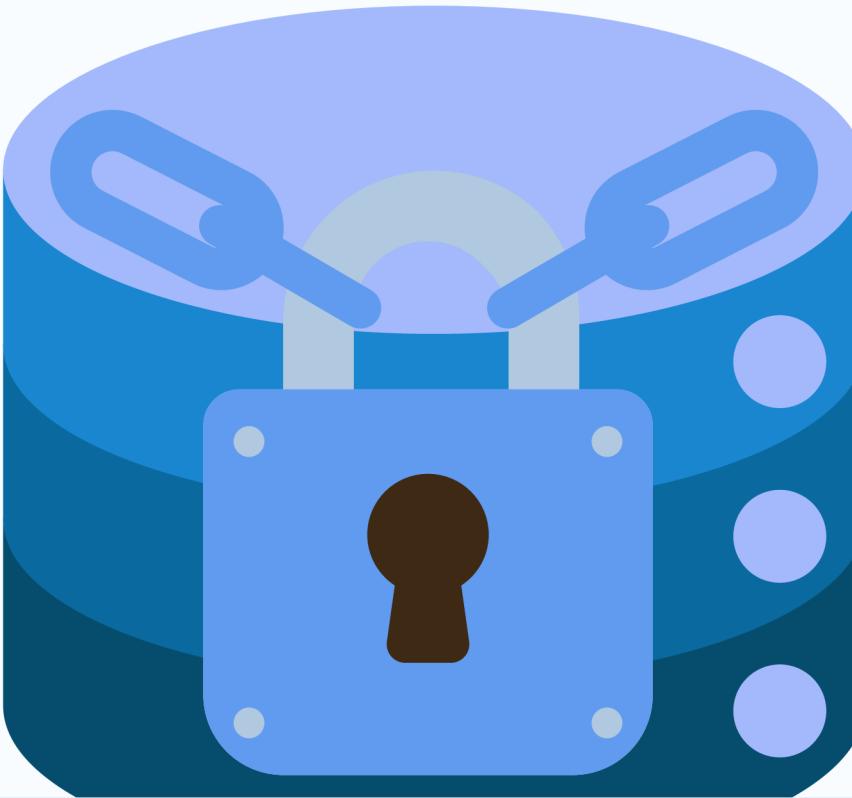




LOCK
co2plant

LOCK이란?



여러 트랜잭션(Transaction:tx)이 동시에 특정자원(테이블, 행 등)에 접근할 때, 데이터의 **일관성(Consistency)**과 **무결성(Integrity)**을 보장하고 **충돌을 방지**하기 위해 필수적으로 사용되는 메커니즘입니다.

왜 필요한가요?

- 동시성 제어를 통한 일관성, 정확성 보장
- 시스템 안정성 보장
- 데이터 정합성 보장



발생 가능한 주요 동시성 문제

갱신 손실(Lost Update)



발생 가능한 주요 동시성 문제

연쇄 복귀(Cascading Rollback)



발생 가능한 주요 동시성 문제

모순성(Inconsistency)



S-ROCK & X-ROCK



SHARED LOCK

S-LOCK, 공유 락, READ LOCK

- 다른 트랜잭션이 해당 데이터를 **읽는 것 허용**
- 변경에는 막혀있음

EXCLUSIVE LOCK

X-LOCK, 배타 락, WRITE LOCK

- 다른 트랜잭션이 해당 데이터에 접근하는 모든 작업을 막음

락 호환성

동일한 데이터에 동시 접근시 허용?

	공유 락	배타 락
공유 락	허용(Allow)	대기(Wait)
배타 락	대기(Wait)	대기(Wait)

비관적 락 & 낙관적 락



PESSIMISTIC LOCK

비관적 락

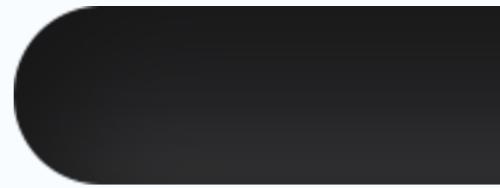
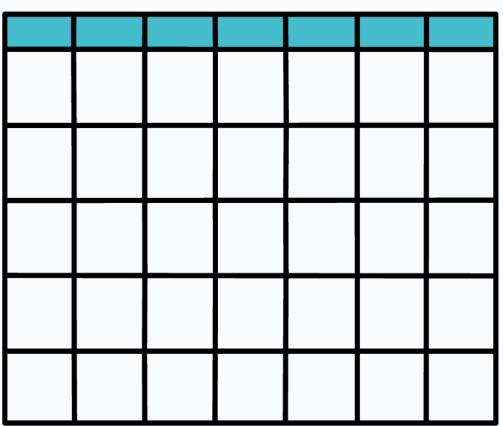
- 충돌이 발생할 것이라고 가정
- 데이터에 접근하는 시점부터 즉시 락을 설정
- 데이터 정합성을 높은 수준으로 보장

OPTIMISTIC LOCK

낙관적 락

- 충돌이 거의 발생하지 않을 것이라고 가정
- 락 없이 작업 수행, 커밋 시점에 데이터 버전 비교
→ 충돌 여부 판별
- 락 대기가 없어 동시성 확보에 이점이 있음

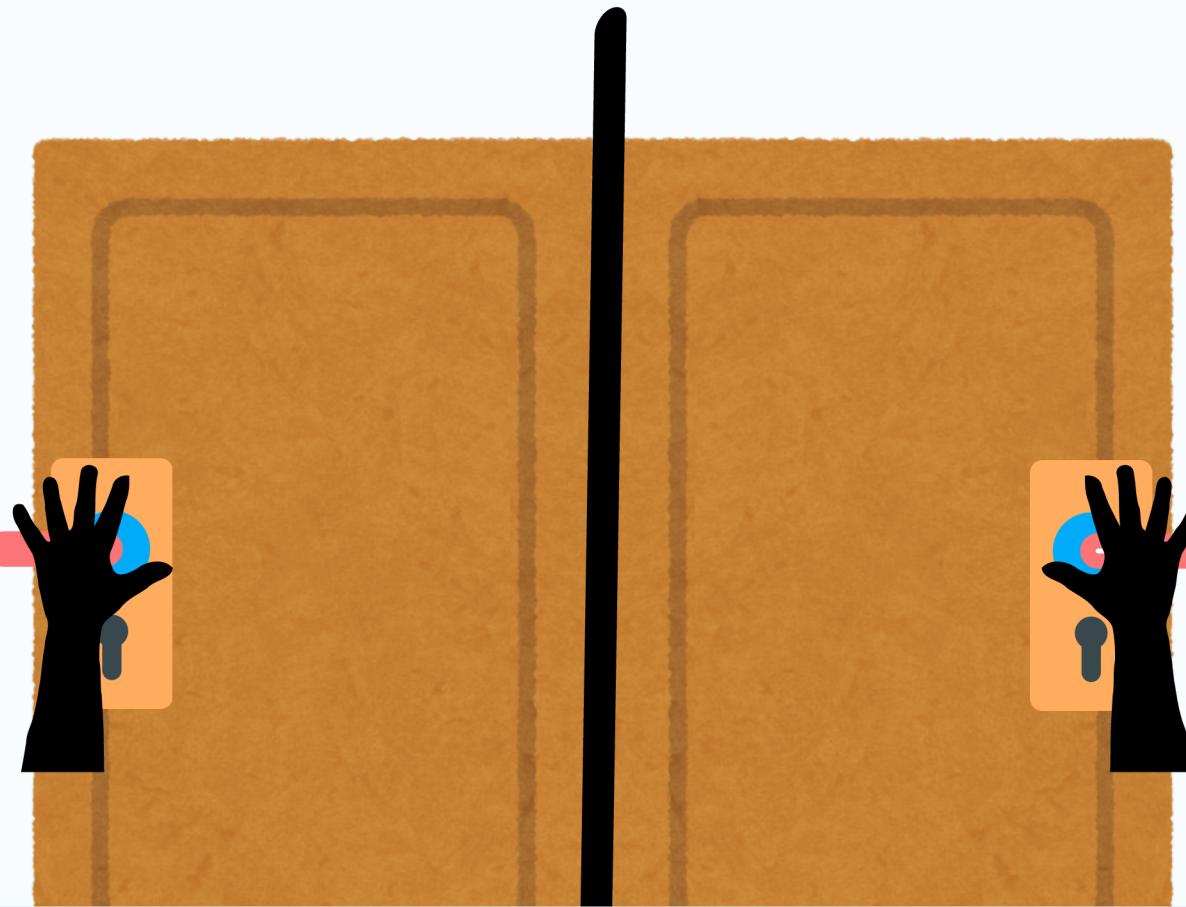
학의 단위



락의 단위

	범위	동시성 수준	관리 오버헤드
테이블 락	테이블 전체	낮음	낮음
페이지 락	데이터 페이지	중간	중간
행 락	개별 행	높음	높음

데드락



둘 이상의 트랜잭션이, 서로 상대방이 점유한 자원의 해제
를 기다리며 무한 대기 상태에 빠지는 것을 의미

데드락 발생 조건

4가지를 모두 충족 하면 데드락 발생

- 상호 배제 (Mutual Exclusion)
- 점유와 대기 (Hold and Wait)
- 비선점 (No Preemption)
- 순환 대기 (Circular Wait)

상호배제

Mutual Exclusion

한번에 1개의 트랜잭션만 특정 자원에 접근할 수 있도록 하는 기법.

사용중인 자원을 다른 트랜잭션에서 접근하려면 지금 그 자원에 접근 중인 트랜잭션이 unlock할때까지 기다려야함.

점유와 대기

Hold and Wait

하나의 자원을 점유하고 있는 상태에서 다른 자원을 추가로 요청하며 대기하는 상태여야함

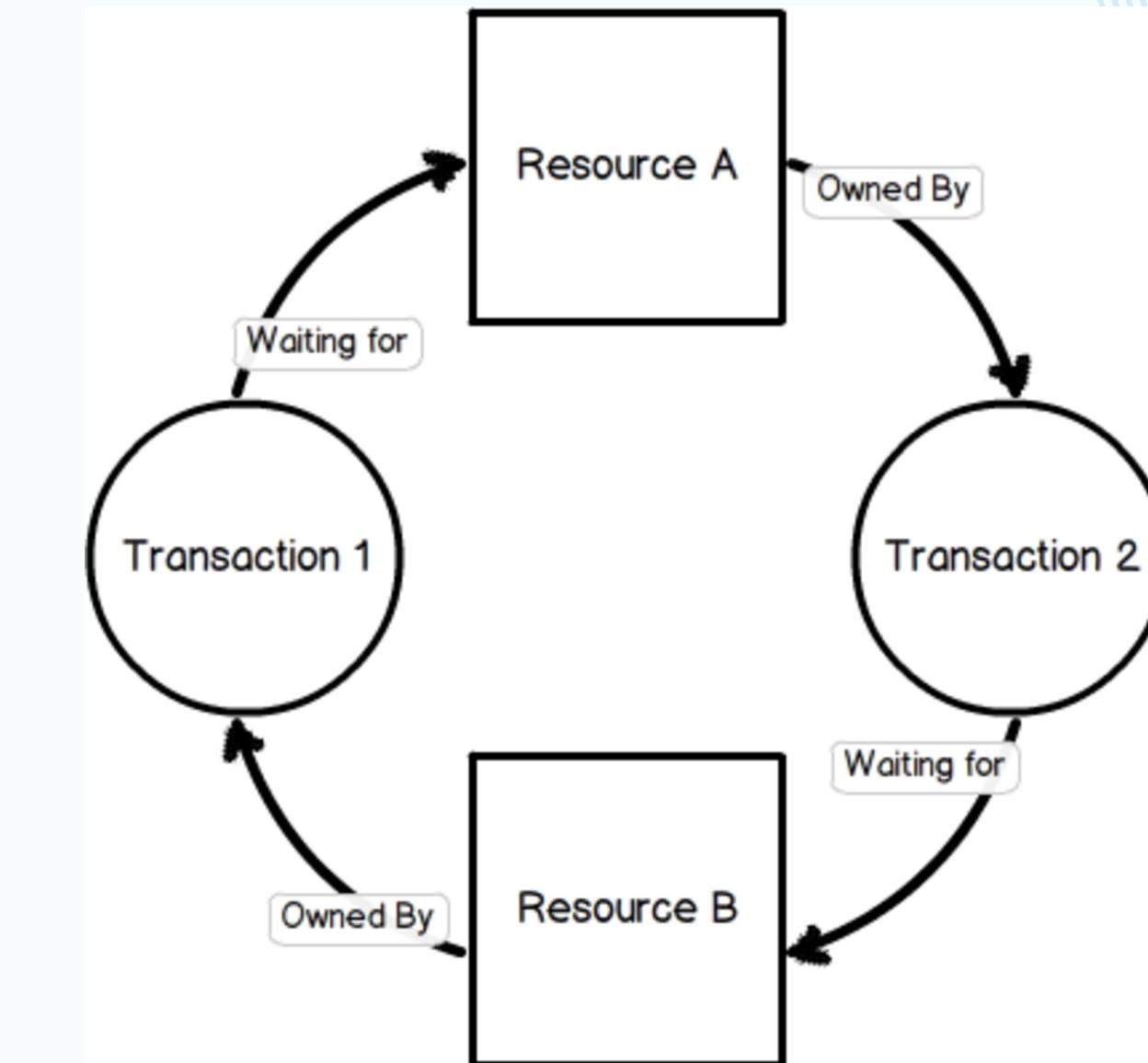
비선점

No preemption

한 트랜잭션이 점유한 자원은 그 트랜잭션이 스스로 점유 해제할때까지 다른 트랜잭션이 자원을 강제로 빼앗을 수 없어야함

순환 대기

Circular wait



자원을 요청하는 트랜잭션들의 관계가 원형고리를 이루어야 함.

데드락 해결법



교착상태에서 벗어나기 위한 전략

탐지와 회복

Detection and Recovery

DBMS에서는 데드락이 발생하는 것을 일단 허용하는 대신 신속한 탐지를 바탕으로 자동 회복시키는 데 집중함.

- 대기 그래프, 주기적 사이클 검사로 탐지
- 희생자 선택으로 회복

예방

Prevention

개발자나 DBA가 애플리케이션 코드나 DB 설계를 통해 예방하는 것이 일반적

- 순환 대기 조건 방지(자원 접근 순서 통일)

예방

- 트랜잭션 크기 최소화(트랜잭션 크기를 짧게 가져감으로 lock상태인 시간을 줄여 데드락 가능성을 낮춤)
- 적절한 격리 수준(불필요하게 높은 격리 수준은 lock상태를 길게 만듦)
- 적절한 인덱스 설계(인덱스가 없다면 테이블을 전체 스캔하는 방법을 쓰게됨, 불필요한 잠금이 늘어남)

MVCC

Multi Version Concurrency Control

데이터를 수정할 때 기존 데이터를 덮어쓰는 대신, 새로운 버전의 데이터를 생성
트랜잭션마다 특정 시점의 데이터 버전을 읽게함

많은 DB에서 사용하는 방법(innoDB, PostgreSQL)

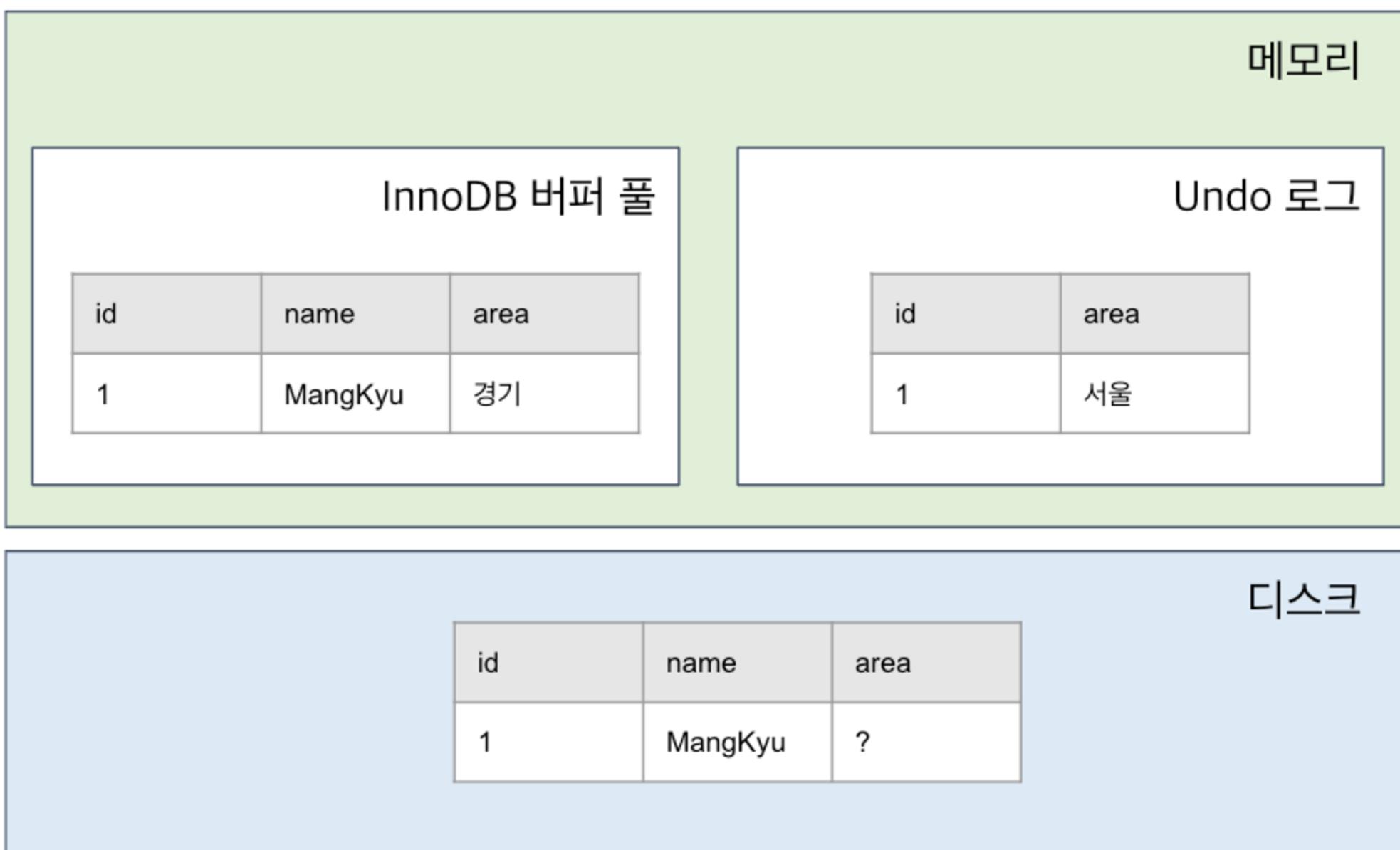
MVCC

INSERT INTO member(id, name, area) VALUES (1,



MVCC

UPDATE member SET area = "경기" where



2PL

2 Phase Locking

추가적으로 innodb에서는 MVCC와 2PL을 동시에 사용함.

이미지 출처

[망나니 개발자]([https://mangkyu.tistory.com/30?
category=761304](https://mangkyu.tistory.com/30?category=761304))