[DB] Transcation & Deadlock

💡 Transaction에 대한 설명과 ACID가 무엇을 뜻하는지를 정확하게 이해하자.

💡 deadlock은 어떤 상황에 발생하는지, 그리고 해결 방법은 어떻게 되는지 이해하자.

트랜잭션의 개념

- 트랜잭션이란 데이터베이스 내에서 수행되는 작업의 최소 단위
- 트랜잭션이 필요한 이유는 데이터를 다룰 때 장애가 일어나는 경우 트랜잭션이 장애 발생 시 데이터를 복구하는 작업의 단위가 되기 때문!
- COMMIT과 ROLLBACK 명령어를 통해 데이터 무결성을 보장
  - COMMIT: 한 트랜잭션의 완료를 뜻하는 명령어
  - ROLLBACK: 한 트랜잭션이 진행되고 있는 중에 문제가 발생했을 때, 그동안의 변경 사항을 취소하고, 이전 COMMIT의 상태로 되돌리는 명령어

트랜잭션 특징

ACID 원칙

특징	해당 성질을 보장하는 방법
원자성	회복
일관성	동시성 제어, 무결성 제약조건
격리성	동시성 제어
지속성	회복

- 원자성(Atomicity): All or Nothing! 트랜잭션이 DB에 모두 반영되거나, 혹은 전혀 반영되지 않아야 된다.
- 일관성(Consistency): 트랜잭션의 작업 처리 결과는 항상 일관성 있어야 한다.
    - 일관성과 고립성을 유지하기 위해서 값에 동시에 접근하지 않도록 하므로 동시성 제어(Locking)를 활용하여 이를 해결한다.
- 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.
    - 어떤 트랜잭션이 실행되다가 장애에 의해 부분 완료되는 상황은 원자성과 지속성이라는 속성에 위배된다. 따라서, DBMS는 이를 유지하기 위해 회복 관리자 프로그램을 이용하는데, 일부만 진행된 트랜잭션을 취소시켜 원자성을 유지할 뿐 아니라 값을 트랜잭션 이전의 상태로 복원시켜 지속성을 유지시켜준다.

- 일관성과 고립성을 유지하기 위해서 값에 동시에 접근하지 않도록 하는 동시성 제어(Locking)를 활용하여 이를 해결한다. 파일 시스템과 같은 경우처럼 값이 덮어쓰기(Overwrite)되는 경우 일관성이 무너질 수 있고, 이는 고립성에 위배되는 경우이므로 Locking을 통해 해결한다.
    - 추가적으로, DBMS는 잘못된 값에 대한 입력이 오면 일관성이 무너질 수 있으므로 이를 유지시키기 위해 무결성 제약조건도 활용한다.

트랜잭션 격리 수준

트랜잭션 격리수준은 고립도와 성능의 트레이드 오프를 조절한다.

- READ UNCOMMITTED: 다른 트랜잭션에서 커밋되지 않은 내용도 참조할 수 있다.
- READ COMMITTED: 다른 트랜잭션에서 커밋된 내용만 참조할 수 있다.

- REPEATABLE READ: 트랜잭션에 진입하기 이전에 커밋된 내용만 참조할 수 있다.
- SERIALIZABLE: 트랜잭션에 진입하면 락을 걸어 다른 트랜잭션이 접근하지 못하게 한다. (성능 매우 떨어짐)

동시성 제어

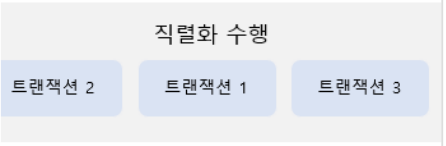
- 개념
  - 동시에 실행되는 여러 개의 트랜잭션이 작업을 성공적으로 마칠 수 있도록 트랜잭션의 실행 순서를 제어하는 기법
  - 동시성 제어 기법의 일종으로 락(Lock, Locking)이 있다.
- 효과
  - 데이터의 무결성 및 일관성을 보장한다.

▼ 참고 자료 (더 자세한 내용)

동시성 제어(Concurrency Control)

1. 프로그래밍 세계에서 '동시성 제어' 의 개념 2. 동시성 제어기법의 종류 3. 동시성 제어기법 종류별 작동 메커니즘

https://velog.io/@ha0kim/동시성-제어



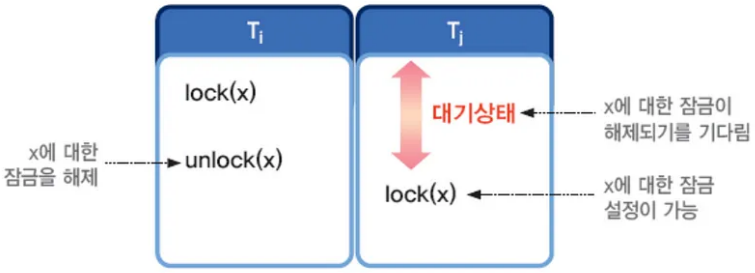
[Database] 8. 트랜잭션, 동시성 제어, 회복

[본 사진은 쉽게 배우는 오라클로 배우는 데이터베이스 개론과 실습 ppt에서 캡처했습니다.]이번 장에서는 트랜잭션(Transaction), 동시성 제어(Locking or Currency Control), 회복(Recovery)에 대해 알

https://mangkyu.tistory.com/30

Lock / Locking

- 데이터의 일관성을 보장하기 위한 방법으로, 가장 대표적인 동시성 제어 기법 중 하나이다.
- LOCK, UNLOCK



[그림1] 잠금의 설정과 해제

교착상태 (Deadlock)

- 개념
  - 여러 transaction들이 각각 자신의 데이터에 대하여 lock을 획득한 상태에서 상대방 데이터에 대하여 접근하고자 대기를 할 때 교차 대기를 하게 되면서 서로 **영원히** 기다리는 상태
- 해결 방법
  - 예방, 회피, 탐지+회복 3가지 방법 알아두자.

1. **교착 상태 예방 (prevention)**

   교착 상태 발생 조건 중 하나를 제거하면서 해결한다 (자원 낭비 엄청 심함)

   > 상호배제 부정 : 여러 프로세스가 공유 자원 사용
   > 점유대기 부정 : 프로세스 실행전 모든 자원을 할당
   > 비선점 부정 : 자원 점유 중인 프로세스가 다른 자원을 요구할 때 가진 자원 반납
   > 순환대기 부정 : 자원에 고유번호 할당 후 순서대로 자원 요구

2. **교착 상태 회피 (avoidance)**

   교착 상태 발생 시 피해나가는 방법

   > 은행원 알고리즘(Banker's Algorithm)
   >
   > - 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는데서 유래함

> - 프로세스가 자원을 요구할 때, 시스템은 자원을 할당한 후에도 안정 상태로 남아있게 되는지 사전에 검사하여 교착 상태 회피
> - 안정 상태면 자원 할당, 아니면 다른 프로세스들이 자원 해지까지 대기

> 자원 할당 그래프 알고리즘(Resource-Allocation Graph Algorithm)
>
> - 자원과 프로세스에 대해 요청 간선과 할당 간선을 적용하여 교착 상태를 회피하는 알고리즘
> - 프로세스가 자원을 요구 시 요청 간선을 할당 간선으로 변경했을 시 사이클이 생성 되는지 확인한다
> - 사이클이 생성된다 하여 무조건 교착상태인 것은 아니다
>
>> 자원에 하나의 인스턴스만 존재 시 교착 상태로 판별한다.
>> 자원에 여러 인스턴스가 존재 시 교착 상태 가능성으로 판별한다
>
> - 사이클을 생성하지 않으면 자원을 할당한다

3. 교착 상태 **탐지 & 회복**

   교착 상태가 되도록 허용하고, 교착 상태가 발생하게 되면 탐지 후 회복시키는 방법

   1. 탐지(Detection)
      - 자원 할당 그래프를 통해 교착 상태를 탐지함
      - 자원 요청 시, 탐지 알고리즘을 실행시켜 그에 대한 오버헤드 발생함
   2. 회복(Recovery)

      교착 상태 일으킨 프로세스를 종료하거나, 할당된 자원을 해제시켜 회복시키는 방법

> 프로세스 종료 방법
>
> - 교착 상태의 프로세스를 모두 중지
> - 교착 상태가 제거될 때까지 하나씩 프로세스 중지
>
> **자원 선점 방법**
>
> - 교착 상태의 프로세스가 점유하고 있는 자원을 선점해 다른 프로세스에게 할당 (해당 프로세스 일시정지 시킴)
> - 우선 순위가 낮은 프로세스나 수행 횟수 적은 프로세스 위주로 프로세스 자원 선점

예상질문

- **3. 트랜잭션이 무엇이고, ACID 원칙에 대해 설명해 주세요.**
  • ACID 원칙 중, Durability를 DBMS는 어떻게 보장하나요?
  • 트랜잭션을 사용해 본 경험이 있나요? 어떤 경우에 사용할 수 있나요?
  • 읽기에는 트랜잭션을 걸지 않아도 될까요?

- **4. 트랜잭션 격리 레벨에 대해 설명해 주세요.**
  • 모든 DBMS가 4개의 레벨을 모두 구현하고 있나요? 그렇지 않다면 그 이유는 무엇일까요?
  • 만약 MySQL을 사용하고 있다면, (InnoDB 기준) Undo 영역과 Redo 영역에 대해 설명해 주세요.

[Database] Naver D2 DBMS는 어떻게 트랜잭션을 관리할까 정리
Naver D2의 트랜잭션 관련 아티클을 정리해보려 한다. 프레임워크 레벨에서만 적용하던 트랜잭션의 내부 동작 방식을 자세히 공부할 수 있는 좋은 기회였다. 트랜잭션 종료의 3가지 상태 - 문제 없이 정상적으로 커밋 - 롤백(철
https://devjem.tistory.com/73

→ 트랜잭션 관리를 위한 DBMS의 전략, MySQL의 InnoDB, Redo와 Undo에 대해 설명

- 그런데, 스토리지 엔진이 정확히 무엇을 하는 건가요?