

# 트랜잭션, 동시성 제어, 외복

## 목차

1. 트랜잭션

2. 동시성 제어

3. 트랜잭션 고립 수준

4. 외복

# 트랜잭션 고립수준

## Transaction Isolation Level

## 학습내용

- 트랜잭션 동시 실행 문제
- 트랜잭션 고립 수준 명령어
- 트랜잭션 고립 수준 실습

## Transaction Isolation Levels

- Isolation : **I** in the acronym ACID
- InnoDB 4개 고립화 수준 제공
  - READ UNCOMMITTED
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE

2023-12-04

컴퓨터공학과

5

## 트랜잭션 동시 실행 문제: 시나리오

- 트랜잭션의 읽기(read)/쓰기(write)

	트랜잭션1	트랜잭션2	발생 문제	동시접근
[상황 1]	읽기	읽기	읽음 (읽기만 하면 아무 문제가 없음)	여용
[상황 2]	읽기	쓰기	<ul style="list-style-type: none"> <li>• 오손 읽기</li> <li>• 반복 불가능 읽기</li> <li>• 유령 데이터 읽기</li> </ul>	여용 옥은 불가 선택
[상황 3]	쓰기	쓰기	경인손실 (절대 허용하면 안 됨)	여용불가 (LOCK을 이용)

2023-12-04

컴퓨터공학과

6

## 오손 읽기(Dirty Read)

- 다른 용어
  - 임시갱신(Temporary Update)
  - Uncommitted Dependency
- 트랜잭션 1: 읽기/트랜잭션 2: 쓰기
  - T2가 작업한 **중간 데이터**를 읽어 문제됨
- 트랜잭션 2가 철회(ROLLBACK)할 경우
  - 트랜잭션 1은 무효가 된 데이터를 읽게 됨
  - 잘못된 결과를 도출하는 연산

2023-12-04

컴퓨터공학과

7

## 오손 읽기(Dirty Read)

- 예시
  - 트랜잭션 T1이 정미림의 잔액을 100000원 감소시킨 후에 트랜잭션 T2는 모든 계좌의 잔액의 평균값을 검색
  - 그 이후에 T1이 어떤 이유로 철회되면 T1이 갱신한 정미림 계좌의 잔액은 원래 상태로 되돌아감
  - 따라서 T2는 완료되지 않은 트랜잭션이 갱신한 데이터, 즉 틀린 데이터를 읽음

T1	T2
UPDATE account SET balance=balance-100000 WHERE cust_name='정미림';	
	SELECT AVG(balance) FROM account;
ROLLBACK;	
	COMMIT;

2023-12-04

컴퓨터공학과

8

## dirty read: 실험 실습테이블 생성

```

/* 실습 테이블 생성 */
Drop TABLE IF EXISTS Users;
CREATE TABLE Users (
  id      INTEGER,
  name    VARCHAR(20),
  age     INTEGER);
INSERT INTO Users VALUES (1, 'HONG GILDONG', 30);

SELECT *
FROM Users;

COMMIT;

```

id	name	age
1	HONG GILDONG	30

## 오손읽기 상황

### • 작업 설명

- 두 개의 트랜잭션을 동시에 실행
  - 트랜잭션 T1, T2가 동시 실행
  - T1은 읽기만 하고 T2는 쓰기를 함
  - T1은 T2가 변경한 데이터를 읽어 작업함
  - T2가 작업 중 철회(ROLLBACK) 함

### • 문제발생

- 오손읽기
- 철회를 하면 T2의 작업은 없던 일이 됨
- T1은 T2가 종료하지 않고 T2가 변경한 데이터를 보고 작업

전주대학교

# 오손읽기 예

- 트랜잭션 T2가 홍길동의 나이를 30에서 21로 변경한 후 철회(ROLLBACK)  
트랜잭션 T1에게 오류를 발생시킨 예

T1(읽는 트랜잭션): READ UNCOMMITTED 모드	T2(쓰는 트랜잭션): READ COMMITTED 모드						
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; START TRANSACTION; USE madang; SELECT * FROM Users WHERE id=1; <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>age</th></tr> </thead> <tbody> <tr> <td>1</td><td>HONG GILDONG</td><td>30</td></tr> </tbody> </table>	id	name	age	1	HONG GILDONG	30	
id	name	age					
1	HONG GILDONG	30					
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang; SET SQL_SAFE_UPDATES=0; UPDATE Users SET age=21 WHERE id=1; SELECT * FROM Users WHERE id=1; <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>age</th></tr> </thead> <tbody> <tr> <td>1</td><td>HONG GILDONG</td><td>21</td></tr> </tbody> </table>	id	name	age	1	HONG GILDONG	21
id	name	age					
1	HONG GILDONG	21					
SELECT * FROM Users WHERE id=1; <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>age</th></tr> </thead> <tbody> <tr> <td>1</td><td>HONG GILDONG</td><td>21</td></tr> </tbody> </table>	id	name	age	1	HONG GILDONG	21	
id	name	age					
1	HONG GILDONG	21					
SELECT * FROM Users WHERE id=1; COMMIT; <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>age</th></tr> </thead> <tbody> <tr> <td>1</td><td>HONG GILDONG</td><td>30</td></tr> </tbody> </table>	id	name	age	1	HONG GILDONG	30	ROLLBACK;
id	name	age					
1	HONG GILDONG	30					

2023-12-04

컴퓨터공학과

11

반복불가능 읽기(non-repeatable read)	
<ul style="list-style-type: none"> <li>트랜잭션 1이 데이터를 읽고</li> <li>트랜잭션 2가 데이터를 쓰고(<b>갱신, UPDATE</b>)</li> <li>트랜잭션 1이 다시 한 번 데이터를 읽을 때 생기는 문제</li> <li>트랜잭션 1이 읽기를 다시 한 번 반복할 경우               <ul style="list-style-type: none"> <li>이전의 결과와 다른 결과가 나오는 현상</li> </ul> </li> </ul>	
2023-12-04	컴퓨터공학과 12

## 반복불가능 읽기

- 먼저 트랜잭션 T2는 모든 계좌의 잔액의 평균값을 검색
- 트랜잭션 T2가 완료되기 전에 트랜잭션 T1이 정미림의 잔액을 100000원 감소시키고 완료
- 트랜잭션 T2가 다시 모든 계좌의 잔액의 평균값을 검색하면 첫 번째 평균값과 다른 값을 보게 됨
- 동일한 읽기 연산을 여러 번 수행할 때 매번 서로 다른 값을 보게 될 수 있음

T1	T2
	SELECT AVG(balance) FROM account;
UPDATE account SET balance=balance-100000 WHERE cust_name='정미림'; COMMIT;	
	SELECT AVG(balance) FROM account; COMMIT;

## 반복불가능 읽기

- [작업 설명] 두 개의 트랜잭션을 동시에 실행
  - 트랜잭션 T1, T2가 동시에 실행됨
  - T1은 읽기만 하고 T2는 쓰기(갱신, UPDATE)를 함
  - T1은 데이터를 읽고 작업을 한 후,
  - T2가 변경한 데이터를 다시 한 번 읽어와 작업

## 반복불가능 읽기

### • [문제 발생] 반복불가능 읽기

- T1이 데이터를 읽고 작업하던 중 T2가 데이터를 변경
- T1은 변경한 데이터를 보고 다시 한 번 작업
- 오손 읽기와 달리 T2가 COMMIT 했기 때문에 **틀린 데이터는 아님**
- 그런데 T1은 **같은 SQL 문**이 다른 결과를 도출하게 됨

2023-12-04

컴퓨터공학과

15

## 반복불가능 읽기: 예

T1(읽는 트랜잭션): READ COMMITTED 모드	T2(쓰는 트랜잭션): READ COMMITTED 모드
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION; USE madang; <div> <div>id</div> <div>name</div> <div>age</div> </div> <div> <div>1</div> <div>HONG GILDONG</div> <div>30</div> </div> SELECT * FROM Users WHERE id=1;	
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION; USE madang; SET SQL_SAFE_UPDATES=0;  UPDATE Users SET age=21 WHERE id=1;  <b>COMMIT;</b> <div> <div>id</div> <div>name</div> <div>age</div> </div> <div> <div>1</div> <div>HONG GILDONG</div> <div>21</div> </div> SELECT * FROM Users WHERE id=1;
SELECT * FROM Users WHERE id=1; <div> <div>id</div> <div>name</div> <div>age</div> </div> <div> <div>1</div> <div>HONG GILDONG</div> <div>21</div> </div> <b>COMMIT;</b>	

2023-12-04

컴퓨터공학과

16



## 유령(phantom)데이터 읽기

- 트랜잭션 1이 데이터를 읽고
- 트랜잭션 2가 데이터를 쓰고(삽입, INSERT)
- 트랜잭션 1이 다시 한 번 데이터를 읽을 때 문제발생
- 트랜잭션 1이 읽기 작업을 다시 한 번 반복할 경우 이전에 없던 데이터(유령 데이터)가 나타나는 현상
- 반복불가능 읽기와 비슷
  - 없던 데이터가 삽입되었기 때문에 다르게 구분

## 유령(phantom)데이터 읽기

- [작업 설명]
- 두 개의 트랜잭션을 동시에 실행
  - 트랜잭션 T1은 읽기만 하고
  - T2는 쓰기(삽입, INSERT)를 수행
  - T1은 데이터를 읽고 작업을 한 후,
  - T2가 변경한 데이터를 다시 한 번 읽어와 작업 수행

## 유령(phantom)데이터 읽기

- [문제 발생] 유령데이터 읽기
  - 이번에는 T1이 T2가 새로운 데이터를 삽입한 사실을 모르고 작업을 수행
  - T2가 COMMIT을 했기 때문에 틀린 데이터는 아님
  - 그러나 T1 입장에서는 새로운 데이터가 반영되어
  - 반복불가능 읽기와 마찬가지로 **같은 SQL문이 다른 결과를 도출**

2023-12-04

컴퓨터공학과

19

## 유령(phantom)데이터 읽기 예

T1(읽는 트랜잭션): REPEATABLE READ 모드	T2(쓰는 트랜잭션): READ COMMITTED 모드
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; START TRANSACTION; USE madang; SELECT * FROM Users WHERE age BETWEEN 10 AND 30;	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang;  INSERT INTO Users VALUES (3, 'Bob', 27); COMMIT;
SELECT * FROM Users WHERE age BETWEEN 10 AND 30; COMMIT;	SELECT * FROM Users WHERE age BETWEEN 10 AND 30;

id	name	age
1	HONG GILDONG	30

id	name	age
1	HONG GILDONG	30
3	Bob	27

id	name	age
1	HONG GILDONG	30
3	Bob	27

2023-12-04

컴퓨터공학과

20

## 트랜잭션 고립 수준 명령어

- 트랜잭션을 동시에 실행시키면서
- 락보다 좀 더 완화된 방법으로 문제 해결 제공 명령어
- 트랜잭션 고립 수준 명령어와 발생 연상

문제 고립 수준	오손 읽기	반복불가능 읽기	유형데이터 읽기
READ UNCOMMITTED	가능	가능	가능
READ COMMITTED	불가능	가능	가능
REPEATABLE READ	불가능	불가능	가능
SERIALIZABLE	불가능	불가능	불가능

## READ UNCOMMITTED(Level = 0)

- 고립 수준이 가장 낮은 명령어
- SELECT 문장을 수행하는 경우
  - 해당 데이터에 **Shared Lock**이 걸리지 않는 Level
- 자신의 데이터에 아무런 공유락을 걸지 않음
  - 배타락은 갱신손실 문제 때문에 걸어야 함
- 다른 트랜잭션이 COMMIT하지 않은 데이터 읽음
  - 그 때문에 **오손(dirty) 페이지**의 데이터를 읽게 됨

## READ UNCOMMITTED 모드 요약

모드	READ UNCOMMITTED
LOCK	SELECT 문 – 공유락 걸지 않음 UPDATE 문 – 배타락 설정 다른 트랜잭션의 공유락과 배타락이 걸린 데이터를 읽음
SQL 문	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
문제점	오손 읽기, 반복불가능 읽기, 유령데이터 읽기

2023-12-04

컴퓨터공학과

23

## READ COMMITTED(Level=1)

- 트랜잭션 T1이 **commit** 을 안 데이터만 다른 트랜잭션 T2가 Read 하는 것을 허용
- 오손(dirty) 페이지의 참조를 피하기 위함
- 자신의 데이터를 읽는 동안 공유락을 걸지만
  - 트랜잭션이 끝나기 전이라도 해지 가능
- 다른 트랜잭션 데이터는 락 오완성 규칙 진행
- 아무 설정을 하지 않으면 **READ COMMITTED** 방식으로 수행

2023-12-04

컴퓨터공학과

24

## READ COMMITTED 모드 요약

모드	READ COMMITTED
LOCK	SELECT 문 – 공유락을 걸고 끝나면 바로 해지 UPDATE 문 – 배타락 설정 다른 트랜잭션이 설정한 공유락은 읽지만, 배타락은 읽지 못함
SQL 문	SET TRANSACTION ISOLATION LEVEL READ COMMITTED
문제점	반복불가능 읽기, 유령데이터 읽기

2023-12-04

컴퓨터공학과

25

## REPEATABLE READ(Level=2)

- default isolation level for InnoDB
- 공유락과 배타락을 **트랜잭션이 종료할 때까지 유지**
  - 다른 트랜잭션이 자신의 데이터를 갱신할 수 없도록 함
  - 단, 삽입(Insert)은 허용 함
- 데이터의 동시성(concurrency) 수준이 낮음
- MySQL
  - 공유락 걸지 않음
  - 최초 트랜잭션 수행 시 Snapshot를 만든 후
  - 해당 스냅샷으로 읽기 수행: 다른 트랜잭션의 변경에도 동일한 결과 생성

2023-12-04

컴퓨터공학과

26

## REPEATABLE READ 모드 요약

모드	REPEATABLE READ:
LOCK	SELECT 문 – 공유락을 걸고 트랜잭션을 끝까지 유지 UPDATE 문 – 배타락 설정 다른 트랜잭션이 설정한 공유락은 읽지만 배타락은 읽지 못함
SQL 문	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
문제점	유형데이터 읽기

2023-12-04

컴퓨터공학과

27

## SERIALIZABLE(Level=3)

- 고립 수준이 가장 높은 명령어
- 실행 중인 트랜잭션은 다른 트랜잭션으로부터 완벽하게 분리
- 데이터 집합에 범위를 지어 잠금을 설정할 수 있음
  - 다른 사용자가 데이터를 변경하려고 할 때
  - 트랜잭션을 완벽하게 분리할 수 있음
- 네 가지 고립와 수준 중 제한이 가장 심함
  - 데이터의 동시성 수준도 낮음
- SELECT 질의의 대상이 되는 테이블에 미리 배타락 설정 효과

2023-12-04

컴퓨터공학과

28

## SERIALIZABLE 모드 요약

모드	SERIALIZABLE
LOCK	<p>SELECT 문 – 공유락을 걸고 트랜잭션을 끝까지 유지</p> <p>UPDATE 문 – 배타락 설정</p> <p>다른 트랜잭션이 설정한 공유락은 읽지만 배타락은 읽지 못함</p> <p>인덱스에 공유락을 설정, 다른 트랜잭션의 INSERT 문이 금지</p>
SQL 문	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
문제점	없음

2023-12-04

컴퓨터공학과

29

## 트랜잭션 고립 수준 실습

1. 반복불가능 읽기 문제 발생 실습 ⇒ [문제발생] 반복불가능 읽기 문제

트랜잭션 T1 READ COMMITTED 모드 (기본 모드)	트랜잭션 T2 READ COMMITTED 모드 (기본 모드)
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang; <b>SELECT SUM(price) 총액</b> <b>FROM Book;</b> <div>총액 144500</div>	
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang; <b>SELECT SUM(price) 총액</b> <b>FROM Book;</b> UPDATE Book SET price=price+500 WHERE bookid=1; <b>SELECT SUM(price) 총액</b> <b>FROM Book;</b> COMMIT; <div>총액 144500</div> <div>총액 145000</div>
<b>SELECT SUM(price) 총액</b> <b>FROM Book;</b> /* 앞의 결과와 다름 */ COMMIT; <div>총액 145000</div>	

2023-12-04

컴퓨터공학과

30

강릉원주대학교

## 트랜잭션 고립 수준 실습

1. 반복불가능 읽기 문제와 방지를 위한 고립수준 상양명령어 ⇒  
[문제방지] REPEATABLE READ 모드

트랜잭션 T1 REPEATABLE READ 모드	트랜잭션 T2 READ COMMITTED 모드 (기본 모드)
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; START TRANSACTION; USE madang; SELECT SUM(price) 총액 FROM Book; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div>	
<b>MySQL</b> - repeatable read에서 Snapshot으로 데이터 조회 - Snapshot: 임시로 생성된 복사본	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang; SET SQL_SAFE_UPDATES=0; UPDATE Book SET price=price+500 WHERE bookid=1; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div>
	(쿼리를 실행하는 중 ...) /* 대기 상태가 됨, T1이 COMMIT하면 실행됨 */
SELECT SUM(price) 총액 FROM Book; /* 앞의 결과와 같음 */ COMMIT; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div>	
	SELECT SUM(price) 총액 FROM Book; COMMIT; <div style="float: right; border: 1px solid black; padding: 2px;">총액 145000</div>

2023-12-04
컴퓨터공학과
31

강릉원주대학교

## 트랜잭션 고립 수준 실습

2. 유령데이터 읽기 문제 발생 실습 ⇒ [문제발생] 유령데이터 읽기

트랜잭션 T1 REPEATABLE READ 모드	트랜잭션 T2 READ COMMITTED 모드(기본 모드)
SET TRANSACTION ISOLATION LEVEL <b>REPEATABLE READ;</b> START TRANSACTION; USE madang; SELECT SUM(price) 총액 FROM Book; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div>	
<b>MySQL</b> - 최초 Select 수행 시 Snapshot 생성하여 데이터 조회 - 커밋 전까지 최초 스냅샷으로 처리 - Snapshot: 임시로 생성된 복사본	SET TRANSACTION ISOLATION LEVEL READ COMM ITTED; START TRANSACTION; USE madang; SELECT SUM(price) 총액 FROM Book; <b>INSERT INTO Book VALUES (11, '테스트', '테스트 출판사', 5500);</b> SELECT SUM(price) 총액 FROM Book; COMMIT; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div> <div style="float: right; border: 1px solid black; padding: 2px;">총액 150000</div>
SELECT SUM(price) 총액 FROM Book; COMMIT; <div style="float: right; border: 1px solid black; padding: 2px;">총액 144500</div>	

2023-12-04
컴퓨터공학과
32



강원원주대학교


## 트랜잭션 고립 수준 실습

### 2. 유령데이터 읽기 문제 방지를 위한 고립 수준 상향 명령어 ⇒[문제방지] SERIALIZABLE 모드

<b>트랜잭션 T1</b> SERIALIZABLE 모드 SET TRANSACTION ISOLATION LEVEL <b>SERIALIZABLE</b> ; START TRANSACTION; USE madang; SELECT SUM(price) 총액 FROM Book;	<b>트랜잭션 T2</b> READ COMMITTED (기본 모드)
<div style="float: right;">총액</div> <div style="border: 1px solid black; padding: 2px;">144500</div>	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; USE madang; SELECT SUM(price) 총액 FROM Book; <div style="float: right;">총액</div> <div style="border: 1px solid black; padding: 2px;">144500</div>
SELECT SUM(price) 총액 FROM Book; /* 앞의 결과와 같음 */ COMMIT;	/* 여기까지 실행해본 후 진행 */ INSERT INTO Book VALUES (12, '테스트', '테스트출판사', 5500);  /* Lock t1 commit 30초 대기 후 종료 */
<div style="float: right;">총액</div> <div style="border: 1px solid black; padding: 2px;">144500</div>	SELECT SUM(price) 총액 FROM Book; COMMIT; <div style="float: right;">총액</div> <div style="border: 1px solid black; padding: 2px;">150000</div>

2023-12-04
컴퓨터공학과
33

강원원주대학교



# ENJOY YOUR DB!!!

2023-12-04
컴퓨터공학과
34



2023-12-04

컴퓨터공학과

35