

18장

MariaDB 트랜잭션과 동시성 제어

트랜잭션

- ❖ 데이터베이스 시스템에서 최소의 업무처리 단위
- ❖ SQL문의 하나의 논리적 작업 단위로 성공하거나 실패하는 일련의 SQL문
- ❖ MariaDB 서버는 트랜잭션을 기본으로 데이터의 일관성을 보증

커밋과 롤백

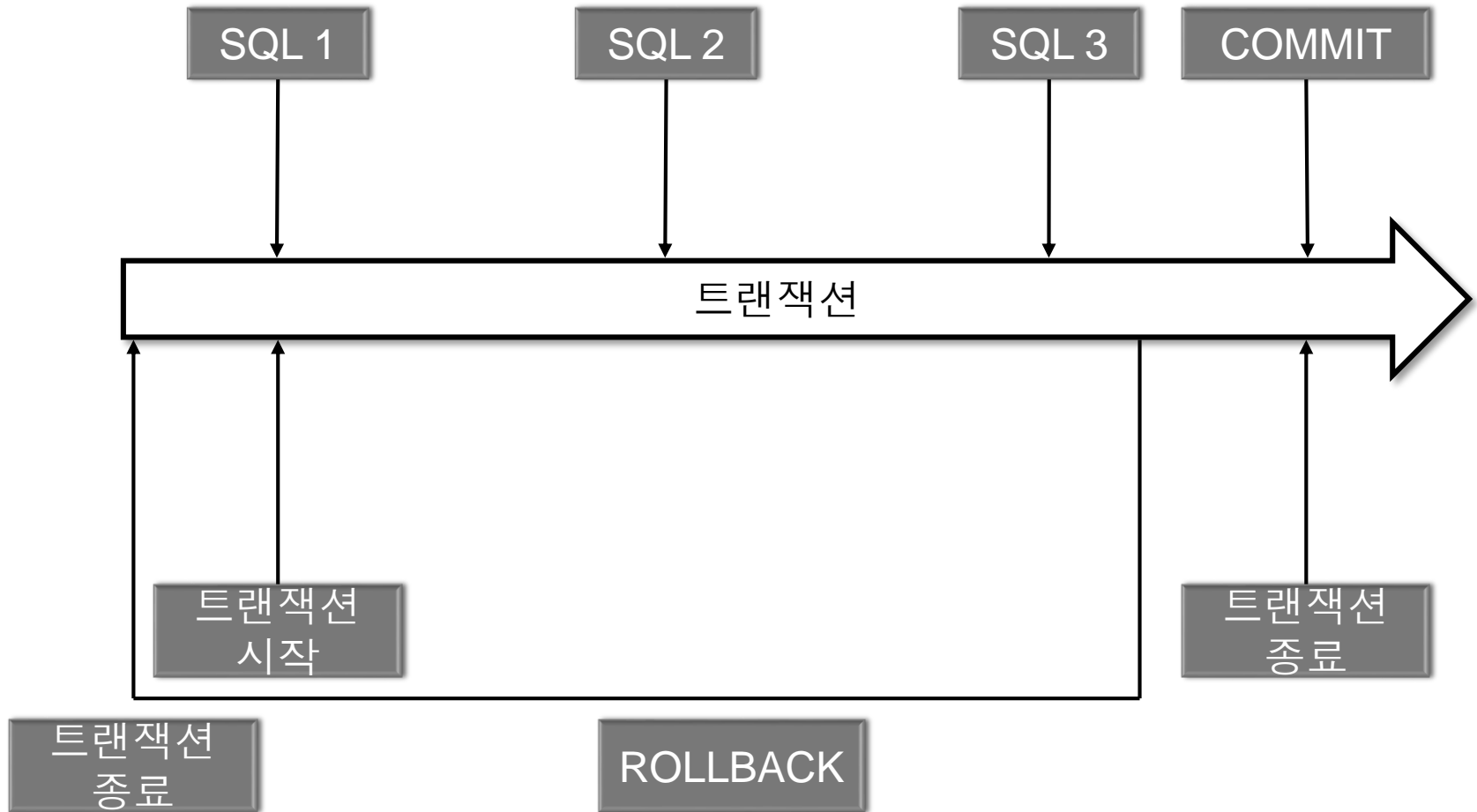
❖ 커밋(Commit)

- 트랜잭션에 의해 변경된 데이터에 대해 확정하는 문장
- 수행된 모든 문장에 대한 데이터의 변경 사항을 영구적으로 반영
- 실행 후 트랜잭션이 종료됨

❖ 롤백(Rollback)

- 트랜잭션의 변경을 취소하는 문장
- 실행 후 트랜잭션이 종료됨

커밋과 롤백



실습 #1 Commit과 Rollback 실습

(1) std_heap2 테이블을 생성함

- CREATE TABLE std_heap2 (
 stdid int, name char(20), grade int, age int, addr char(60), dno int
) ENGINE=InnoDB;

(2) 트랜잭션 실행 준비

- SET autocommit = 0;
- SET sql_safe_updates = 0;
- SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

(3) COMMIT/ROLLBACK 실습

- INSERT INTO std_heap2 VALUES (101, 'JS Kim', 3, 22, 'Chuncheon', 3);
- INSERT INTO std_heap2 VALUES (102, 'PK Lee', 2, 21, 'Seoul', 1);
- INSERT INTO std_heap2 VALUES (105, 'AB Park', 2, 20, 'Wonju', 2);
- INSERT INTO std_heap2 VALUES (106, 'CD Hwang', 1, 19, 'Chuncheon', 2);
- INSERT INTO std_heap2 VALUES (110, 'XY Choi', 2, 21, 'Seoul', 3);
- COMMIT;
- SELECT * FROM std_heap2 ;

(3) COMMIT/ROLLBACK 실습 (계속)

- INSERT INTO std_heap2 VALUES (201, 'EE Moon', 2, 22, 'Kangrung', 1);
- INSERT INTO std_heap2 VALUES (202, 'PL Lim', 3, 23, 'Chuncheon', 4);
- ROLLBACK;
- SELECT * FROM std_heap2 ;

- UPDATE std_heap2 SET name = 'XX XXXX', grade = 9
WHERE stid = 101;
- COMMIT;
- SELECT * FROM std_heap2 ;
- UPDATE std_heap2 SET name = 'AA AAAA', grade = 7
WHERE stid = 102;
- ROLLBACK;
- SELECT * FROM std_heap2 ;
- UPDATE std_heap2 SET name = 'BB BBBB', grade = 8
WHERE stid = 102;
- COMMIT;
- UPDATE std_heap2 SET name = 'CC CCCC', grade = 9
WHERE stid = 102;
- ROLLBACK;
- SELECT * FROM std_heap2 ;

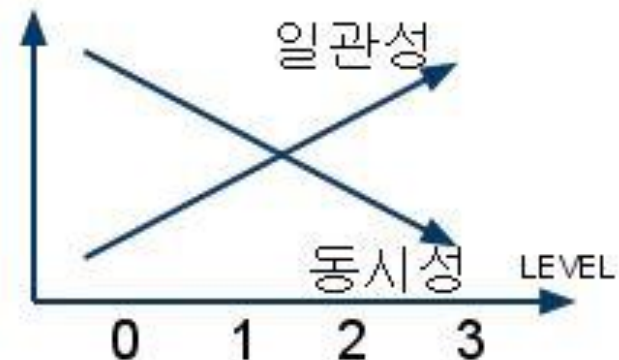
동시성 제어

❖ 트랜잭션이 동시에 수행 되면 발생할 수 있는 문제

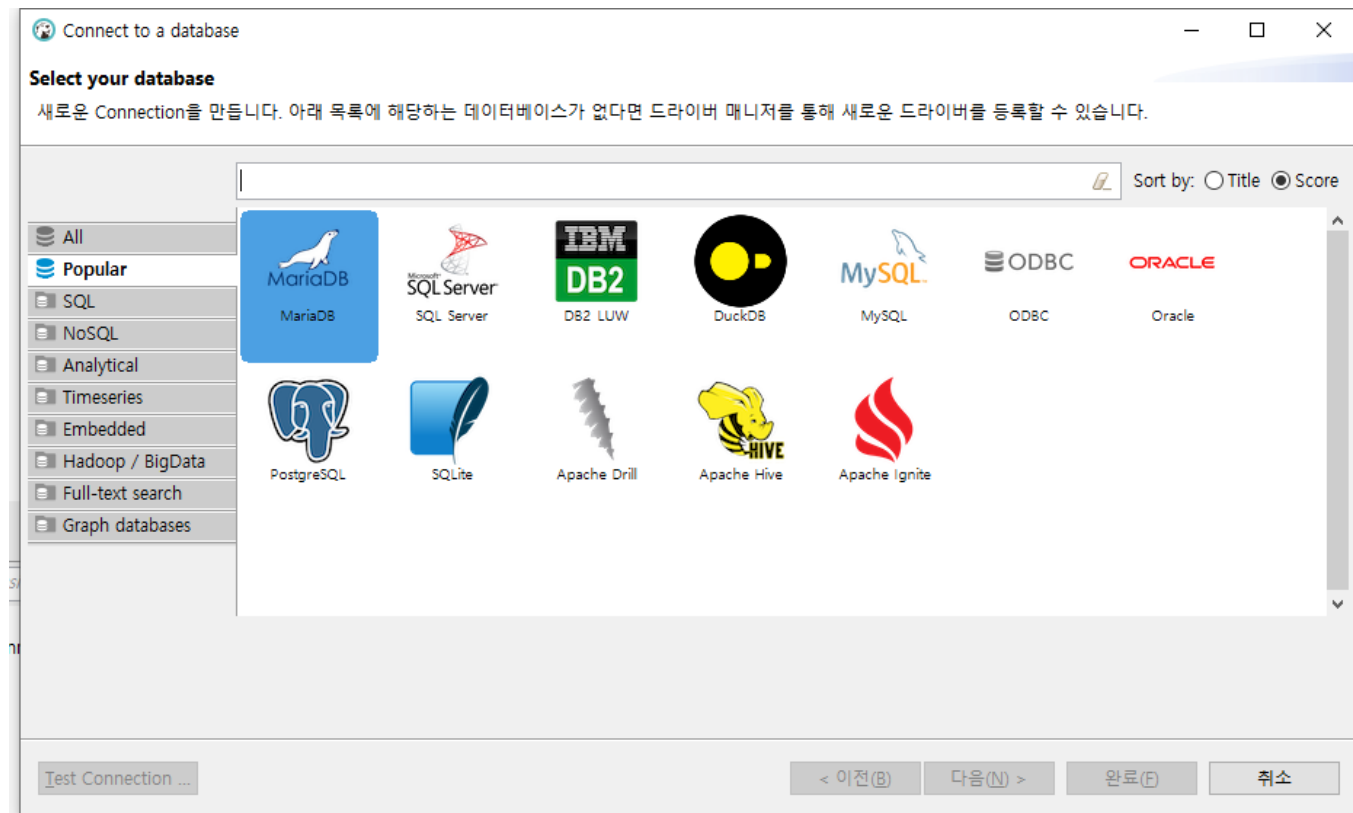
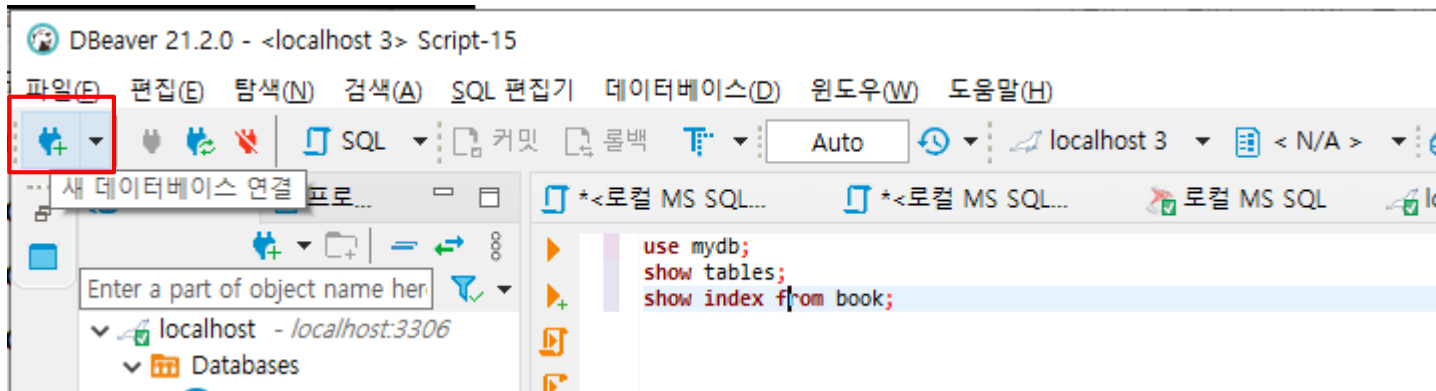
- Lost Update(갱신 손실): 갱신 내용을 잃어버림
- Dirty Read(오손 판독): 존재하지 않는 값을 읽음
- Non-Repeatable Read(비반복 읽기): 여러 번 읽을 때 값이 서로 다름
- Phantom Read(유령 읽기): 읽는 범위 내에 존재하는 삭제/삽입된 유령 레코드를 읽음/읽지 못함

❖ 트랜잭션 고립 수준 명령어


- DBMS는 트랜잭션을 동시에 실행시키면서 Lock보다 좀 더 완화된 방법으로 문제를 해결하기 위해 제공하는 명령어




MariaDB 세션 여러개 실행(1)



MariaDB 세션 여러개 실행(2)

 Connect to a database

Connection Settings
MariaDB connection settings

 MariaDB

MainDriver propertiesSSHProxySSL

Server

Server Host:localhostPort:3306

Database:

Authentication (Database Native)


Username:root

Password:●●●●●●☒ Save password locally

Advanced

Server Time Zone:Auto-detect

Local Client:MySQL Binaries

 You can use variables in connection parameters.

Connection details (name, type, ...)

Driver name: MariaDB

Edit Driver Settings

Test Connection ...

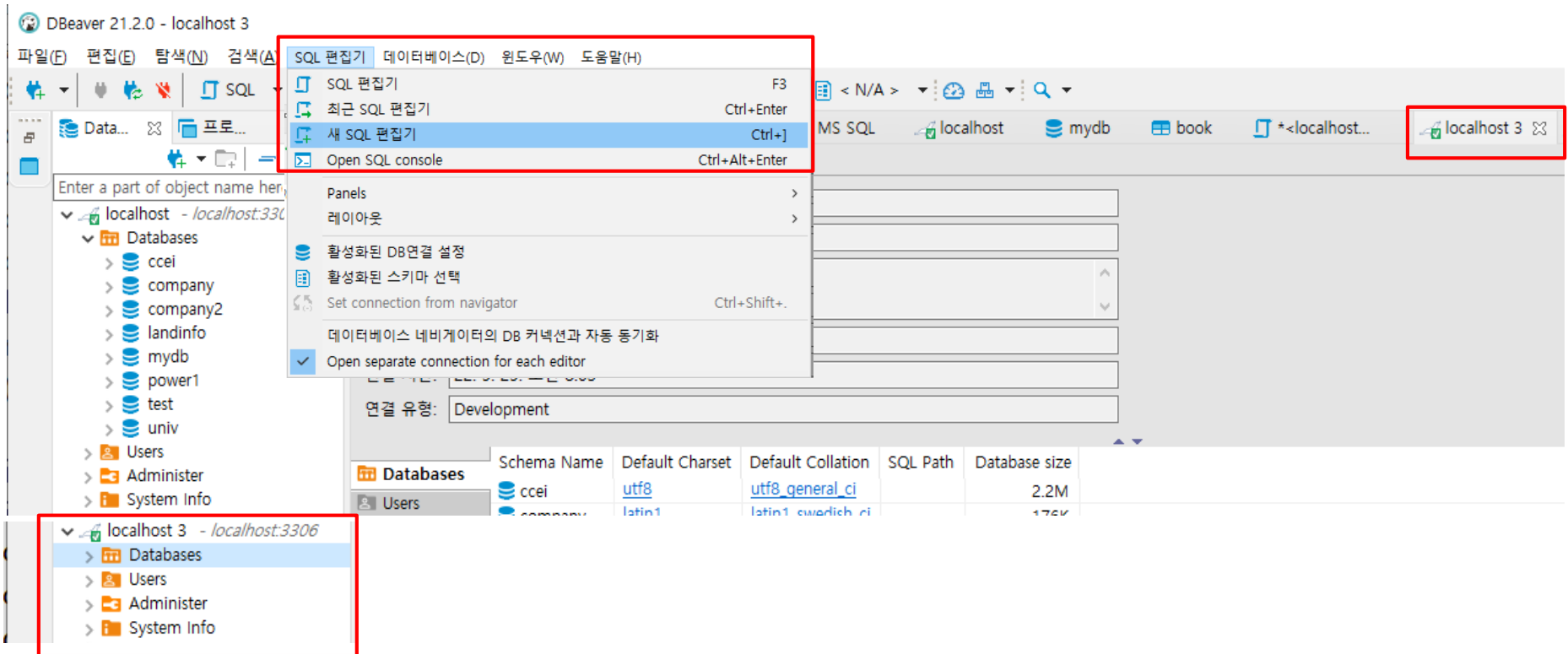
< 이전(B)

다음(N) >

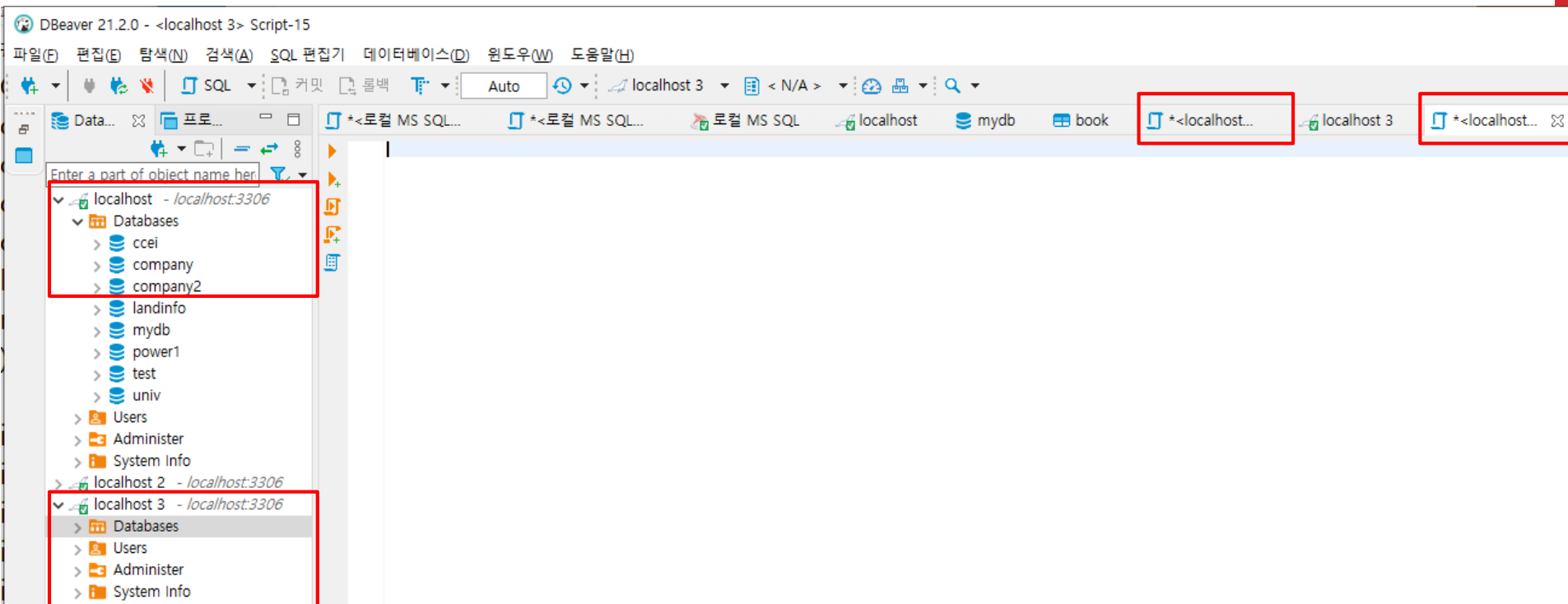
완료(F)

취소

MariaDB 세션 여러개 실행(3)



MariaDB 세션 여러개 실행(4)



로킹 기법 vs 다중버전 로킹 기법 비교

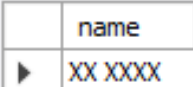
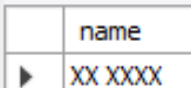
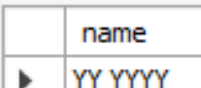
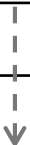
일관성 레벨	로킹 기법	다중버전 로킹 기법	차이점
READ UNCOMMITTED	Lost update 방지 Dirty read 발생 Unrepeatable read 발생 Phantom record 발생	Lost update 방지 Dirty read 방지 (이전 버전 읽음) Unrepeatable read 방지 (이전 버전 읽음) Phantom record 발생	
READ COMMITTED	Lost update 방지 Dirty read 방지 Unrepeatable read 발생 Phantom record 발생	Lost update 방지 Dirty read 방지 Unrepeatable read 방지 (이전 버전 읽음) Phantom record 발생	
REPEATABLE READ	Lost update 방지 Dirty read 방지 Unrepeatable read 방지 Phantom record 발생	Lost update 방지 Dirty read 방지 Unrepeatable read 방지 Phantom record 발생	
SERIALIZABLE	Lost update 방지 Dirty read 방지 Unrepeatable read 방지 Phantom record 방지	Lost update 방지 Dirty read 방지 Unrepeatable read 방지 Phantom record 방지	

실습 #2 갱신 손실 방지 실습



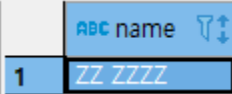
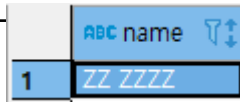
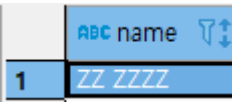
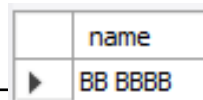
곰돌이	시간	한둥이
Update 통장 set money = 10만 Where 이름 = 커플통장	T1	
	T2	Update 통장 set money = 5만 Where 이름 = 커플통장
뭐야 왜 5만원으로 되어있지?	T3	

❖ 다른 트랜잭션에 의해 값이 덮혀쓰이는지 테스트

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 READ UNCOMMITTED 모드
SET autocommit = 0; SET sql_safe_updates = 0; SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED ;	
	SET autocommit = 0; SET sql_safe_updates = 0; SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED ;
SELECT name FROM std_heap2 WHERE stid = 101; 	SELECT name FROM std_heap2 WHERE stid = 101; 
UPDATE std_heap2 SET name = 'YY YYYY' WHERE stid = 101;	
	UPDATE std_heap2 SET name = 'ZZ ZZZZ' WHERE stid = 101; Lock 대기
SELECT name FROM std_heap2 WHERE stid = 101; COMMIT; 	 COMMIT;

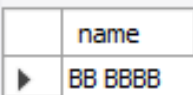
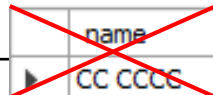
왜 **ZZ ZZZZ**가
아닐까요?

❖ 다른 트랜잭션에 의해 값의 변경이 취소되는지 테스트


트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 READ UNCOMMITTED 모드
SELECT name FROM std_heap2 WHERE stdid = 101;	
	SELECT name FROM std_heap2 WHERE stdid = 101;
UPDATE std_heap2 SET name = 'AA AAAA' WHERE stdid = 101;	
	UPDATE std_heap2 SET name = 'BB BBBB' WHERE stdid = 101;
	<div>Lock 대기로 인해, COMMIT을 ROLLBACK 전에 실행하는 것이 불가능함!!</div>
ROLLBACK;	
SELECT name FROM std_heap2 WHERE stdid = 101;	
	
COMMIT;	<div>COMMIT;</div> <div>SELECT name FROM std_heap2 WHERE stdid = 101;</div> <div>COMMIT;</div> 

실습 #3 허상 읽기 실습

- ❖ 허상 읽기 발생 테스트 --> 다중 버전 동시성 제어에서는 허상 읽기가 방지됨
(2PL 기법은 T2가 T1의 수정한 값을 읽지만, 다중버전은 이전 버전의 값을 읽는 방법으로 허상 읽기를 방지함)

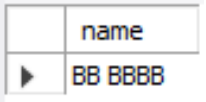
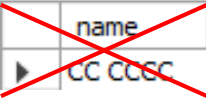
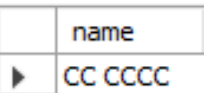
트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 READ UNCOMMITTED 모드
SELECT name FROM std_heap2 WHERE stdid = 1	
UPDATE std_heap2 SET name = 'CC CCCC' WHERE stdid = 101;	
ROLLBACK;	SELECT name FROM std_heap2 WHERE stdid = 101; 이전버전 BB BBBB 읽음 
SELECT name FROM std_heap2 WHERE stdid = 101;	SELECT name FROM std_heap2 WHERE stdid = 101;
COMMIT;	COMMIT;

❖ 허상 읽기 발생 **방지** 테스트 (2PL 기법은 T2가 lock 대기하지만, 다중버전은 lock 대기하지 않고 이전 버전의 값을 읽는 방법으로 허상 읽기를 방지함)

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 READ COMMITTED 모드		
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED ;		
SELECT name FROM std_heap2 WHERE stdid = 101;			
UPDATE std_heap2 SET name = 'CC CCCC' WHERE stdid = 101;	<div><table><tr><td>name</td></tr><tr><td>BB BBBB</td></tr></table></div>	name	BB BBBB
name			
BB BBBB			
새버전 CC CCCC 생성	SELECT name FROM std_heap2 WHERE stdid = 101; 이전버전 BB BBBB 읽음 Lock 대기 없음		
ROLLBACK;	<div></div>		
SELECT name FROM std_heap2 WHERE stdid = 101;	<div><table><tr><td>name</td></tr><tr><td>BB BBBB</td></tr></table></div>	name	BB BBBB
name			
BB BBBB			
<div><table><tr><td>name</td></tr><tr><td>BB BBBB</td></tr></table></div>	name	BB BBBB	COMMIT;
name			
BB BBBB			
COMMIT;			

실습 #4 비 반복적 읽기 실습

- (1) READ COMMITTED 모드: 비반복적 읽기 발생 테스트 --> 다중 버전에서는 방지됨
(2PL 기법은 T2가 T1의 수정한 값을 읽지만, 다중버전은 이전 버전의 값을 읽으므로 비반복적 읽기가 방지됨)

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 READ COMMITTED 모드
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED ;
	SELECT name FROM std_heap2 WHERE stid = 101;
UPDATE std_heap2 SET name = 'CC CCCC' WHERE stid = 101; COMMIT; 새버전 CC CCCC 생성	
	SELECT name FROM std_heap2 WHERE stid = 101;
	이전버전 BB BBBB 읽음 
	COMMIT;
	SELECT name FROM std_heap2 WHERE stid = 101;
	COMMIT ; 

(2) REPEATABLE READ 모드 (2PL 기법은 T1이 lock 대기를 하지만, 다중버전은 lock 대기하지 않고 값을 수정할 수 있으며, T2는 이전 버전의 값을 읽는 방법으로 비반복적 읽기를 방지함)

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 REPEATABLE READ 모드
	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
	SELECT name FROM std_heap2 WHERE stid = 101;
UPDATE std_heap2 SET name = 'DD DDDD' WHERE stid = 101; COMMIT;	<div> <div>name</div> <div>▶ CC CCCC</div> </div>
Lock 대기없이 새버전 DD DDDD 생성	
	SELECT name FROM std_heap2 WHERE stid = 101;
	이전버전 CC CCCC읽음
	COMMIT;
	SELECT name FROM std_heap2 WHERE stid = 101;
	DD DDDD읽음
	COMMIT;

ABC name
1 DD DDDD

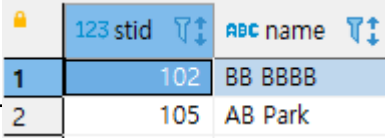
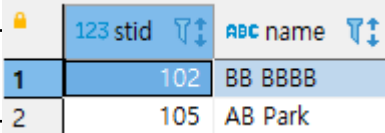
name
▶ CC CCCC

ABC name
1 DD DDDD

실습 #4 직렬화 (유령 레코드 읽기)

❖ 아래 각 트랜잭션의 SELECT 결과를 화면 캡처하고, 그 이유를 작성하여 제출하시오.

➤ (1) REPEATABLE READ 모드 – 유령 레코드 발생

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 REPEATABLE READ 모드
	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
	SELECT stdid, name FROM std_heap2 WHERE stdid >= 102 AND stdid <= 105;
INSERT INTO std_heap2 VALUES (103, 'KK KKKK', 3, 20, 'Chuncheon', 2);	
COMMIT;	
	SELECT name FROM std_heap2 WHERE stdid >= 102 AND stdid <= 105;
	
	COMMIT;

103번이 유령
취급 당함

➤ (2) SERIALIZABLE 모드 – 유령 레코드 방지

트랜잭션 T1 READ UNCOMMITTED 모드	트랜잭션 T2 SERIALIZABLE 모드						
	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;						
DELETE FROM std_heap2 WHERE stid = 103; COMMIT;							
	SELECT stid, name FROM std_heap2 WHERE stid >= 102 AND stid <= 105;						
INSERT INTO std_heap2 VALUES (103, 'KK KKKK', 3, 20, 'Chuncheon', 2); COMMIT;	<div><div><div>🔒</div><div>123 stid ↕</div><div>ABC name ↕</div></div><table><tr><td>1</td><td>102</td><td>BB BBBB</td></tr><tr><td>2</td><td>105</td><td>AB Park</td></tr></table></div>	1	102	BB BBBB	2	105	AB Park
1	102	BB BBBB					
2	105	AB Park					
Lock 대기							
	SELECT stid, name FROM std_heap2 WHERE stid >= 102 AND stid <= 105;						
	COMMIT;						
	<div><div><div>🔒</div><div>123 stid ↕</div><div>ABC name ↕</div></div><table><tr><td>1</td><td>102</td><td>BB BBBB</td></tr><tr><td>2</td><td>105</td><td>AB Park</td></tr></table></div>	1	102	BB BBBB	2	105	AB Park
1	102	BB BBBB					
2	105	AB Park					

103번 삽입이
실행됨

유령 레코드가 없는
정상적인 결과임