COSE371(00) DATABASES

Assignment #4

2022320033 박종혁

1. Schedule 1

Session 1: SERIALIZABLE	Session 2: READ UNCOMMITTED
MariaDB [db2022320033]> set session transaction isolation level serializable; Query OK, 0 rows affected (0.00 sec)	MariaDB [db2022320033]> set session transaction isolation level read uncommitted; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)	
	MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)
	MariaDB [db2022320033]> select class from ships where s_name='캔버라함'; +----------+ | class | +----------+ | 베를린함급 | +----------+ 1 row in set (0.00 sec)
MariaDB [db2022320033]> update ships set class='광개토대왕함급' where s_name='캔버라함'; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	
	MariaDB [db2022320033]> select class from ships where s_name='캔버라함'; +----------+ | class | +----------+ | 광개토대왕함급 | +----------+ 1 row in set (0.00 sec)
MariaDB [db2022320033]> rollback; Query OK, 0 rows affected (0.01 sec)	
	MariaDB [db2022320033]> select class from ships where s_name='캔버라함'; +----------+ | class | +----------+ | 베를린함급 | +----------+ 1 row in set (0.00 sec)

설명:

'READ UNCOMMITTED' 레벨에서는 'dirty read'를 허용한다. 즉, 다른 트랜잭션에서 값을 업데이트하고 해당 행에 대한 X-lock을 아직 해제하지 않았더라도 그 행의 데이터를 읽어올 수 있음을 뜻한다. 따라서 '캔버라함'의 class를 '광개토대왕함급'으로 변경하는 Session1의 업데이트 이후 Session1이 아직 commit하거나 rollback하지 않았음에도(X-lock을 해제하지 않았음에도), Session2의 트랜잭션에서는 그 값을 읽어 '베를린함급'이 아닌 '광개토대왕함급'을 리턴한다. 또한, Session2의 한 트랜잭션 내에서 같은 행에 대한 여러 번의 read를 수행했을 때 매번 결과가 변하는 것으로 보아, 'non-repeatable read'가 발생함을 알 수 있다.

2. Schedule 2

Session 1: SERIALIZABLE	Session 2: READ COMMITTED
MariaDB [db2022320033]> set session transaction isolation level serializable; Query OK, 0 rows affected (0.00 sec)	MariaDB [db2022320033]> set session transaction isolation level read committed; Query OK, 0 rows affected (0.00 sec)
	MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)	
	MariaDB [db2022320033]> select class from ships where s_name='캔버라함'; +-----------------+ | class | +-----------------+ | 베를린함급 | +-----------------+ 1 row in set (0.00 sec)
MariaDB [db2022320033]> update ships set class='광개토대왕함급' where s_name='캔버라함'; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	

설명:

MySQL 8.0 Reference Manual 15.7.2.3 Consistent Nonlocking Reads에 따르면, InnoDB는 multi-versioning을 사용하며, 'READ COMMITTED'와 'REPEATABLE READ'레벨에서는 consistent read가 어떤 lock도 걸지 않는다고 한다. 따라서 위 스케쥴의 Session2에서 '캔버라함'의 데이터를 읽고 있지만 그 행에 대해 S-lock을 걸지 않기 때문에, Session1은 해당 행에 대해 X-lock을 얻어 값을 업데이트하는 것이 가능하다. 만약 위 스케쥴 이후 commit하지 않은 채로 곧바로 Session2에서 다시 한번 '캔버라함'의 class를 읽는다면, 'READ COMMITTED'는 'dirty read'를 허용하지 않으므로 아직 Session1의 트랜잭션이 commit하지 않은(아직 X-lock이 걸려 있는) 변경사항을 읽어올 수 없어 이전 read의 버전을 사용하여 '베를린함급'을 리턴할 것이다.

3. Schedule 3

Session 1: SERIALIZABLE	Session 2: REPEATABLE READ
MariaDB [db2022320033]> set session transaction isolation level serializable; Query OK, 0 rows affected (0.00 sec)	MariaDB [db2022320033]> set session transaction isolation level repeatable read; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)	
	MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> update ships set class='광 개 토 대 왕 함 급 ' where s_name='캔 버 라 함 '; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	
	MariaDB [db2022320033]> select class from ships where s_name='캔 버 라 함 '; +-----------------+ | class | +-----------------+ | 베 를 린 함 급 | +-----------------+ 1 row in set (0.00 sec)
MariaDB [db2022320033]> commit; Query OK, 0 rows affected (0.01 sec)	
	MariaDB [db2022320033]> select class from ships where s_name='캔 버 라 함 '; +-----------------+ | class | +-----------------+ | 베 를 린 함 급 | +-----------------+ 1 row in set (0.00 sec)

설명:

우선, Session1의 트랜잭션에서 '캔버라함' 행에 대해 X-lock을 얻고 값을 업데이트한다. 이후 Session2에서 '캔버라함'의 데이터를 읽으려 할 때, 그 행은 Session1 트랜잭션에 의해 X-lock이 걸려 있는 상태이므로(Session1 트랜잭션이 아직 commit하지 않은 상태이므로) 업데이트된 값인 '광개토대왕함급'을 읽어오지 못하고 이전 버전의 데이터를 이용하여 업데이트 전 값인 '베를린함급'을 리턴한다(Multi-Versioning). 이후 Session1 트랜잭션이 commit하면서 X-lock을 해제해 업데이트된 값인 '광개토대왕함급'을 읽어올 수 있게 되었지만, 'REPEATABLE READ' 레벨에서는 한 트랜잭션 내에서 같은 데이터를 읽을 경우 그 값이 변하지 않아야 하므로(non-repeatable read 금지) Session2 트랜잭션에서 다시 '캔버라함'의 class를 읽을 때는 이전 read에서의 값인 '베를린함급'을 리턴해야 한다.

4. Schedule 4

Session 1: SERIALIZABLE	Session 2: SERIALIZABLE
MariaDB [db2022320033]> set session transaction isolation level serializable; Query OK, 0 rows affected (0.00 sec)	MariaDB [db2022320033]> set session transaction isolation level serializable; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)	
	MariaDB [db2022320033]> begin; Query OK, 0 rows affected (0.00 sec)
MariaDB [db2022320033]> update ships set class='워 싱 턴 함 급 ' where s_name='런 던 함 ' -> ; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	
	MariaDB [db2022320033]> update ships set class='콜 래 트 대 왕 함 급 ' where s_name='아 메 리 카 호 '; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0
MariaDB [db2022320033]> select class from ships where s_name='아 메 리 카 호 '; +------------------+ | class | +------------------+ | 베 이 징 함 급 | +------------------+ 1 row in set (16.99 sec)	
	MariaDB [db2022320033]> select class from ships where s_name='런 던 함 '; ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction

설명:

'SERIALIZABLE' 레벨에서는 S-lock과 S-lock의 경우를 제외하고는 각 lock들이 incompatible하므로, 해당 행에 대한 lock을 얻으려면 이미 걸려 있는 lock이 해제될 때까지 기다려야 한다. 위 스케쥴에서, Session1 트랜잭션은 '런던함' 행에 대한 X-lock을, Session2 트랜잭션은 '아메리카호' 행에 대한 X-lock을 얻은 상태이다. 이 상태에서 Session1 트랜잭션이 '아메리카호' 행에 대한 S-lock을 요청하지만 해당 행은 Session2 트랜잭션이 X-lock을 걸어 둔 상태이므로 Session2 트랜잭션이 commit하거나 rollback할 때까지 Session1 트랜잭션은 기다려야 한다. 반대로, Session2 트랜잭션이 '런던함' 행에 대한 S-lock을 요청하지만 해당 행은 Session1 트랜잭션이 X-lock을 걸어 둔 상태이므로 Session1 트랜잭션이 commit하거나 rollback할 때까지 Session2 트랜잭션은 기다려야 한다. 두 트랜잭션이 서로 상대가 commit할 때까지 자신도 commit할 수 없는 상태, 즉 deadlock에 빠진 상황이다. 이때 위 실행 예시에서는 DBMS가 deadlock을 감지하여 Session2 트랜잭션을 rollback시킴으로써 해결하였다. Session2 트랜잭션이 rollback되었으므로 Session2에서 업데이트한 '아메리카호' 행의 class도 다시 이전의 '베이징함급'으로 돌아오고, 해당 행에 대한 X-lock도 해제되었다. X-lock이 해제되었으므로 Session1 트랜잭션은 '아메리카호' 행에 대한 S-lock을 얻을 수 있으며, '베이징함급'을 리턴한다.