SELECT … FOR UPDATE SKIP LOCKED Optimization

목적: 이 강의에서는 MySQL의 `SELECT FOR UPDATE` 문에 `SKIP LOCKED`를 사용하여 성능을 최적화하는 방법을 배웁니다.

포인트: `SKIP LOCKED`의 개념과 이를 사용하여 동시성을 향상시키는 방법을 이해하는게 중요합니다.

▼ Understanding SELECT FOR UPDATE and SKIP LOCKED

SELECT FOR UPDATE: 조회하는 레코드에 잠금을 거는 기능, 다른 트랜잭션이 해당 레코드에 쓰거나 변경할 수 없게 함.

SKIP LOCKED: 잠금이 걸린 레코드를 건너뛰고, 잠금이 없는 레코드만 조회하고 잠금 가능.

▼ Optimization Strategies

동시성이 높은 환경에서 성능 개선에 유리함 (예: 선착순 쿠폰 배포 시스템).

- 잠금 대기 시간 감소 및 처리량 증가.

▼ Practice

Task 1: Try Applying SKIP LOCKED

목적: 선착순 쿠폰 시스템 구현을 통한 `SELECT FOR UPDATE SKIP LOCKED`의 효과 실습.

시나리오: 사용자가 쿠폰 발급을 요청하면 할당되지 않은 쿠폰을 조회하고 잠금을 건 후 사용자에게 전달

쿠폰 테이블 `coupon`:

- `id` : 기본 키
- `status` : 쿠폰 상태 (`available', assigned, used`)
- `user_id` : 쿠폰을 가진 사용자
- `created_at` : 쿠폰 생성 날짜
- `expires_at` : 쿠폰 만료 날짜

1. **쿠폰 테이블 설계:**

```
CREATE TABLE coupons (
    id INT AUTO_INCREMENT PRIMARY KEY,
    status VARCHAR(20) NOT NULL,
    user_id INT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
```

```
    expires_at DATETIME
);
```

2. **쿠폰 테스트 데이터 삽입:**

3. **쿠폰 발급 요청을 처리하는 SQL 문 작성:**

```
SELECT id FROM coupons
WHERE status = 'available'
ORDER BY id
LIMIT 1
FOR UPDATE
```

```
UPDATE coupons
SET status = 'assigned', user_id = [사용자 ID]
WHERE id = [선택된 쿠폰 ID];
```

3. **Python 코드를 이용해서 쿠폰 발급 요청을 병렬로 요청**

```
import threading
import pymysql
import queue
import time

def request_coupon(user_id, connection_pool):
    try:
        connection = connection_pool.get(True, 10)  # 연결 풀에서 연결을 가져옴

        with connection.cursor() as cursor:
            # 사용 가능한 쿠폰 선택
            cursor.execute("SELECT id FROM coupons WHERE status = 'available' ORDER BY id LIMIT 1 FOR UPDATE;")
            result = cursor.fetchone()

            if result:
                coupon_id = result[0]
                # 쿠폰 상태 업데이트
                update_query = "UPDATE coupons SET status = 'assigned', user_id = %s WHERE id = %s;"
                cursor.execute(update_query, (user_id, coupon_id))
                connection.commit()
                print(f"User {user_id} received coupon {coupon_id}")
            else:
                print(f"No available coupons for user {user_id}")

    except pymysql.MySQLError as e:
        print(f"Error while connecting to MySQL: {e}")
    finally:
        connection_pool.put(connection)  # 사용한 연결을 다시 연결 풀에 반환

def create_connection():
    return pymysql.connect(
        host='localhost',
        database='test',
        user='root',
        password='1234')

# 연결 풀 생성
connection_pool = queue.Queue()
for _ in range(10):  # 예를 들어 10개의 연결을 생성
```

```
    connection_pool.put(create_connection())


start_time = time.perf_counter()
threads = []
for _ in range(100):
    # 스레드 생성 및 실행
    for user_id in range(100):
        thread = threading.Thread(target=request_coupon, args=(user_id, connection_pool))
        thread.start()
        threads.append(thread)

    # 모든 스레드가 완료될 때까지 기다림
    for thread in threads:
        thread.join()

    threads.clear()
end_time = time.perf_counter()
elapsed_time = end_time - start_time
print(f"Execution time: {elapsed_time} seconds")

# 연결 풀 내의 연결들을 종료
while not connection_pool.empty():
    connection = connection_pool.get()
    connection.close()
```

4. **User 에게 발급된 쿠폰 테스트 데이터 롤백:**

```
UPDATE coupons SET user_id = NULL WHERE user_id IS NOT NULL;

UPDATE coupons SET status = 'available'
```

5. **이번엔 `SELECT ... FOR UPDATE SKIP LOCKED` 를 통한 쿠폰 발급 요청 후 성능 비교**