비관적 락 동시성 제어

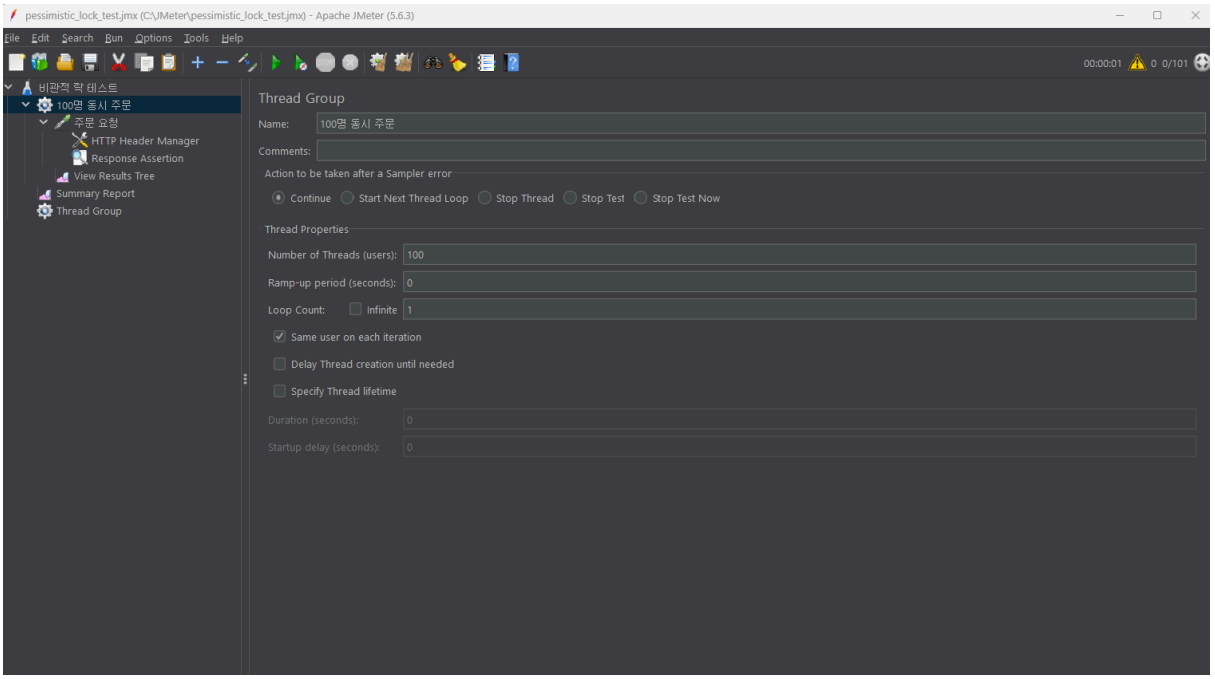
비관적 락 동시성 제어 테스트 결과

프로젝트: LapPick
목적: SELECT ... FOR UPDATE의 동시성 제어 효과 검증

---

1. 테스트 환경

도구: Apache JMeter 5.6.3
동시 사용자: 100명 (Ramp-up: 0초)
테스트 상품: JMETER_LOCK_TEST (초기 재고 1개)
서버: Spring Boot 3.4.0, Oracle 21c
Connection Pool: HikariCP (max-pool-size: 50)



JMeter Thread Group 설정 - 100명 동시 요청 (Ramp-up 0초)

---

2. 테스트 케이스

케이스 A: 비관적 락 미적용

```
SELECT G.*,
       (SELECT SUM(IPGO_QTY) FROM GOODS_IPGO WHERE GOODS_NUM = G.GOO
DS_NUM) AS STOCK_QTY
FROM GOODS G
WHERE G.GOODS_NUM = #{goodsNum}
-- FOR UPDATE 없음
```

케이스 B: 비관적 락 적용

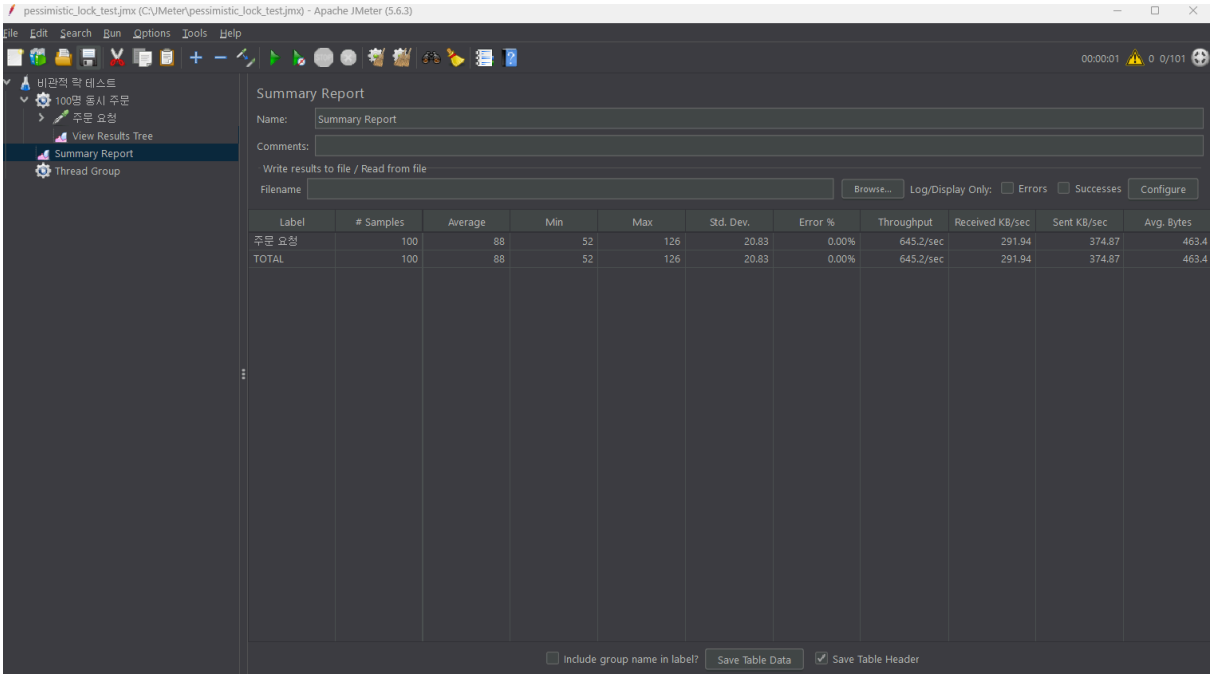
```
SELECT GOODS_NUM
FROM GOODS
WHERE GOODS_NUM = #{goodsNum}
FOR UPDATE WAIT 5
```

---

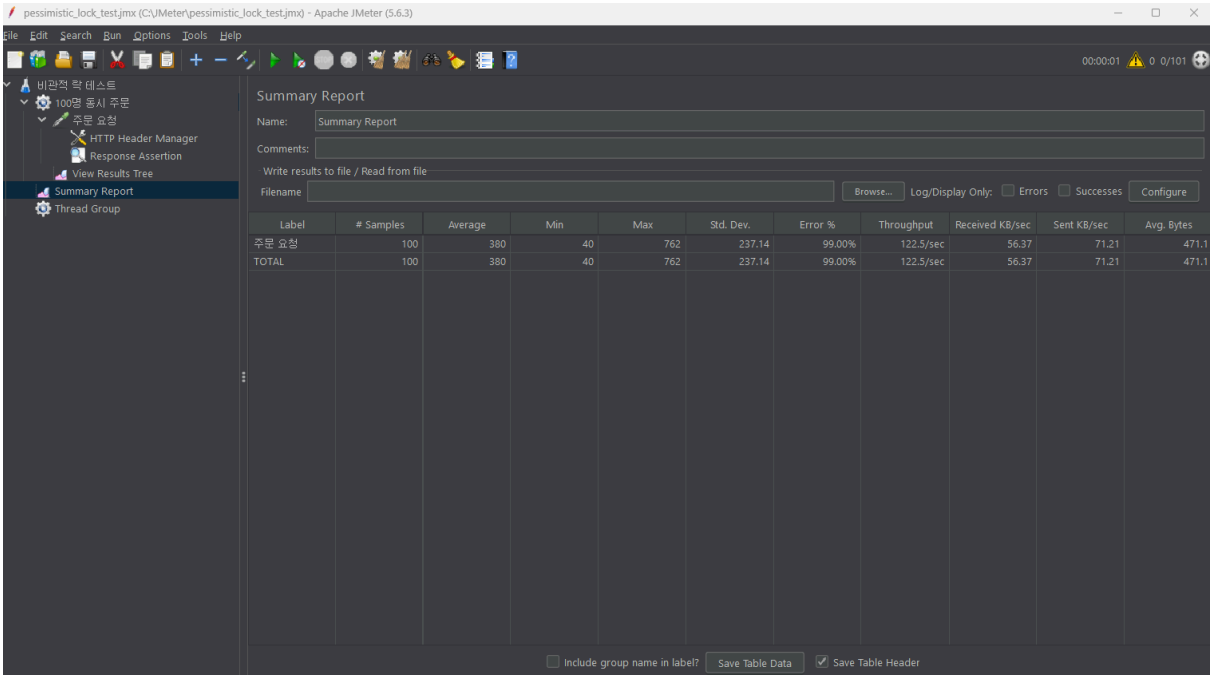
3. 측정 결과

3.1 JMeter 결과

항목	비관적 락 X	비관적 락 O	차이
# Samples	100	100	-
Average (ms)	88	380	+292ms
Error %	0.00%	99.00%	+99%p
Throughput	645/sec	122/sec	-523/sec



비관적 락 미적용 - Average 88ms, Error 0%, Throughput 645/sec

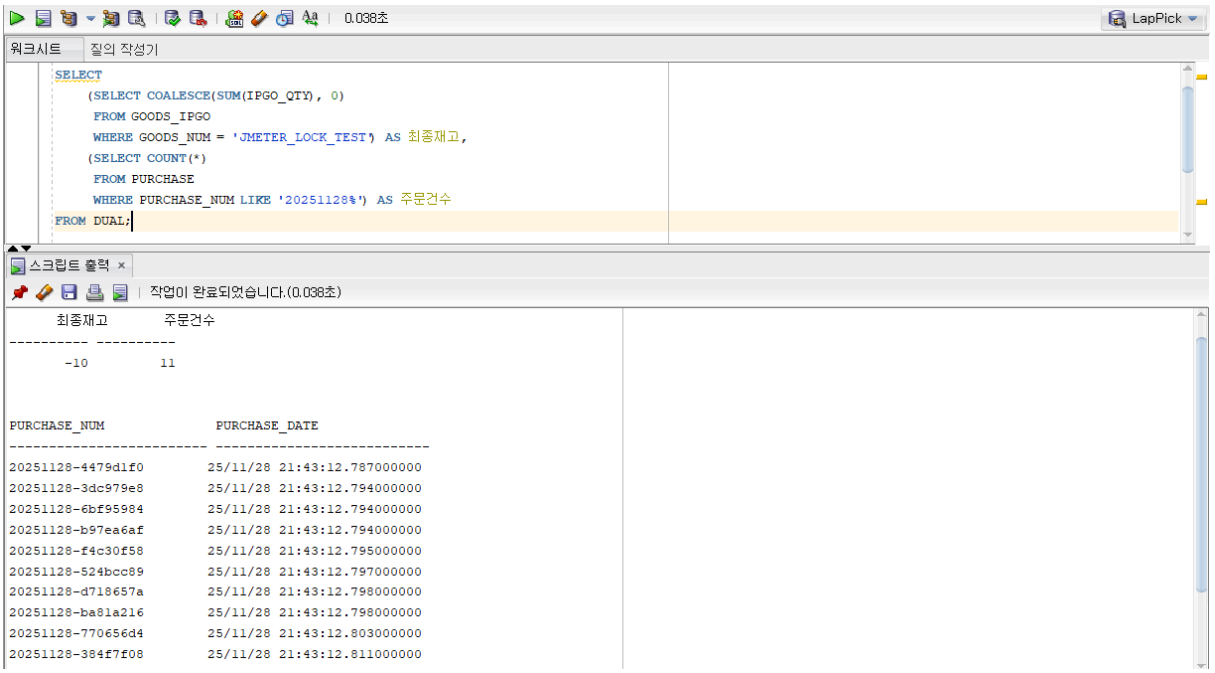


비관적 락 적용 - Average 380ms, Error 99%, Throughput 122/sec

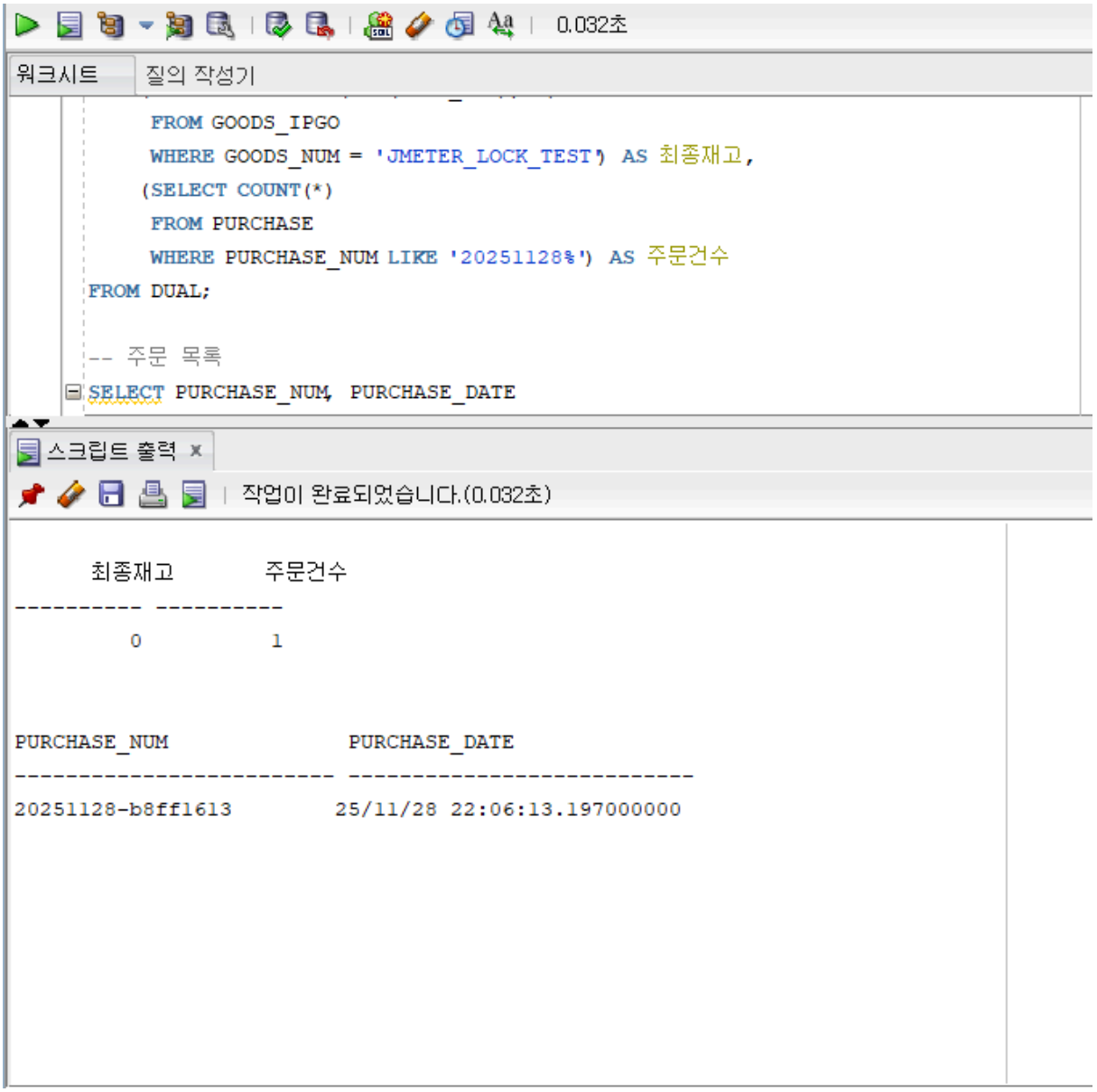
참고: Error 99%는 99명이 재고 부족으로 주문 실패한 것. HTTP 응답은 모두 200 OK.

3.2 DB 검증

항목	비관적 락 X	비관적 락 O
최종 재고	-10개	0개
주문 성공 건수	11건	1건
오버셀링 발생	O	X



비관적 락 미적용 - 최종 재고 -10개, 주문 11건 성공



비관적 락 적용 - 최종 재고 0개, 주문 1건 성공

3.3 주문 생성 시각 (비관적 락 X)

```
PURCHASE_NUM          PURCHASE_DATE
--------------------- --------------------------
20251128-4479d1f0      21:43:12.787
20251128-3dc979e8       21:43:12.794
20251128-6bf95984       21:43:12.794
20251128-b97ea6af      21:43:12.794
20251128-f4c30f58      21:43:12.795
20251128-524bcc89       21:43:12.797
20251128-d718657a      21:43:12.798
20251128-ba81a216      21:43:12.798
20251128-770656d4       21:43:12.803
20251128-384f7f08      21:43:12.811
20251128-94e07f44       21:43:12.815
```

11건의 주문이 28ms 이내에 동시 발생.

---

4. 문제 분석

4.1 Race Condition 발생 과정

```
Thread-1: [재고 조회: 1] → [주문 생성] → [재고 차감: 0] → COMMIT
Thread-2: [재고 조회: 1] → [주문 생성] → [재고 차감: -1] → COMMIT
Thread-3: [재고 조회: 1] → [주문 생성] → [재고 차감: -2] → COMMIT
...
Thread-11: [재고 조회: 1] → [주문 생성] → [재고 차감: -10] → COMMIT
```

여러 트랜잭션이 동시에 재고 1을 읽고, 각자 주문을 진행하여 오버셀링 발생.

4.2 비관적 락의 동작

```
Thread-1: [락 획득] → [재고 조회: 1] → [주문 생성] → [재고 차감: 0] → COMMIT [락 해제]
Thread-2:   [대기...] [락 획득] → [재고 조회: 0] → 주문 실패
Thread-3:   [대기...] [락 획득] → [재고 조회: 0] → 주문 실패
...
Thread-100: [대기...] [락 획득] → [재고 조회: 0] → 주문 실패
```

FOR UPDATE로 Row-Level Lock을 걸어 순차 처리함으로써 재고 정합성 보장.

---

5. 성능 vs 정합성

5.1 성능 비교

평균 응답 시간: 88ms → 380ms (약 4.3배 증가)
처리량: 645 req/sec → 122 req/sec (약 81% 감소)

비관적 락 사용 시 대기 시간으로 인한 성능 저하 발생.

5.2 데이터 정합성

비관적 락 X: 재고 1개 → 11명 주문 성공 (재고 -10)
비관적 락 O: 재고 1개 → 1명 주문 성공 (재고 0)

쇼핑몰에서 오버셀링은 고객 불만 및 신뢰도 하락으로 이어지므로, 성능 저하를 감수하고 비관적 락을 적용.

---

6. 결론

재고 1개 상품에 대해 100명이 동시 주문 시, 비관적 락 없이는 11건의 오버셀링이 발생했다. FOR UPDATE를 통한 비관적 락 적용으로 정확히 1건만 주문이 처리되어 재고 정합성을 확보했다.

성능은 약 4배 저하되었으나, 커머스 시스템에서 재고 정확도는 필수이므로 비관적 락 사용이 적절한 선택이다.

참고

JMeter Error %가 99%로 표시되는 이유:

- Response Assertion으로 "success":true 검증
- 99명은 재고 부족으로 "success":false 응답 받음
- HTTP 응답은 모두 200 OK, 비즈니스 로직 실패는 JSON 필드로 구분