

Eclipse Blocking 검증

Eclipse Breakpoint 비관적 락 Blocking 테스트

프로젝트: LapPick

목적: SELECT ... FOR UPDATE의 Row-Level Lock 동작 확인

1. 테스트 환경

- 서버: Spring Boot 3.4.0 + Oracle 21c
- 클라이언트: User A (일반 브라우저), User B (시크릿 모드)
- 테스트 상품: goods_999999 (Eclipse 테스트용노트북)
- 초기 재고: 1개

2. Breakpoint 설정

PurchaseService.java 53번 라인에 Breakpoint 설정.

```
goodsMapper.selectGoodsForUpdate(goodsNum);
log.debug("락 획득 성공: 상품번호={}", goodsNum); // ← 이 줄에 Breakpoint
```

User A가 FOR UPDATE 락을 획득한 직후 실행을 멈춰서, User B의 대기 상태를 확인하는 방식.

```
44     Log.info("주문 시작: 회원번호={}, 상품번호={}", memberNum, command.getGoodsNums().length);
45     for (int i = 0; i < command.getGoodsNums().length; i++) {
46         String goodsNum = command.getGoodsNums()[i];
47         int quantity = Integer.parseInt(command.getGoodsQtys()[i]);
48
49         // 1-1. 비관적 락 획득 (FOR UPDATE WAIT)
50         // → 다른 트랜잭션이 이 상품을 주문 중이면 최대 5초 대기
51         try {
52             goodsMapper.selectGoodsForUpdate(goodsNum);
53             Log.debug("락 획득 성공: 상품번호={}", goodsNum);
54         } catch (Exception e) {
55             Log.error("락 획득 실패: 상품번호={}, goodsNum, e");
56             throw new CannotAcquireLockException("현재 잔여 재고가 부족하여 주문이 처리되고 있습니다. 잠시 후 다시 시도해주세요.", e);
57         }
58     }
59
60     // 1-2. 재고 확인
61     GoodsStockResponse goodsStock = goodsService.getGoodsDetailWithStock(goodsNum);
62     if (goodsStock == null) {
63         throw new IllegalStateException("상품 정보를 찾을 수 없습니다. (상품번호: " + goodsNum + ")");
64     }
65 }
66 
```

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer GoodsMapper.xml GoodsService.java PurchaseController.java

Console Debug Shell

LapPick - LapPickApplication [Spring Boot App] C:\WProgram Files\Eclipse Adoptium\jdk-17.0.16.8-hotspot\bin\javaw.exe (2025, 11, 29, 오전 12:33:12) [pid: 9]

2025-11-29T08:35:08.956+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. ResultSet.next() returned false
2025-11-29T08:35:08.956+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. ResultSet.close() returned void
2025-11-29T08:35:08.956+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. Connection.getMetaData() returned ConnectionMetaData
2025-11-29T08:35:08.956+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. PreparedStatement.close() returned void
2025-11-29T08:35:08.956+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. Connection.clearWarnings() returned void
2025-11-29T08:35:08.957+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. Connection.getTransactionIsolation() returned int
2025-11-29T08:35:08.957+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. Connection.getAutoCommit() returned boolean
2025-11-29T08:35:08.957+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] log4jdbc.log4j2 : 1. Connection.setAutoCommit(boolean)
2025-11-29T08:35:08.957+09:00 INFO 9956 --- [lappick] [nio-8080-exec-7] 1.purchase.service.PurchaseService : 주문 시작: 회원번호=mem_100041, 상품번호=mem_100041

Eclipse에서 53번 라인에 Breakpoint가 설정된 화면

3. 테스트 진행 과정

3-1. User A 주문 시작

User A가 주문/결제 버튼 클릭.

상태:

- Eclipse: 53번 라인에서 정지
- User A 브라우저: 로딩 중 (응답 대기)
- DB: User A 트랜잭션 활성, goods_999999 Row Lock 보유

The screenshot shows a web browser window for 'LapPick' with the URL `localhost:8080/purchases/order-direct?goodsNum=goods_999999&qty=1`. The page displays a form for entering shipping information (Address 4799, Seoul, Dongdaemun-gu, 143, 104동 506호) and a delivery message (비관적 락 동시성 테스트 중 (User A)). Below this is a payment section with payment method selection (Credit Card selected), bank information (국민은행 123-456-789123 (주)LapPick), and a recipient name (김희원). At the bottom, it shows a summary: 총 상품: 1종, 총 수량: 1개, 최종 결제 금액: 1,000,000원, and a large '결제하기' button. A loading icon is visible on the right side of the page.

User A 주문 화면 - 왼쪽 상단 무한 로딩 아이콘 표시

3-2. User B 주문 시도

User A가 멈춘 상태에서 User B가 동일 상품 주문/결제 버튼 클릭.

결과:

- User B 브라우저: 로딩 중 (응답 없음)
- Eclipse: User B 요청이 Breakpoint에 도달하지 않음
- 해석: User B의 SELECT ... FOR UPDATE가 DB Connection 레벨에서 Blocking, 애플리케이션 로직 진입 불가

주문/결제

localhost:8080/purchases/order-direct?goodsNum=goods_999999&qty=1

LapPick BRAND PRODUCTS NOTICE FAQ LOGOUT

주소

13522 우편번호 찾기

경기 성남시 분당구 판교로 421

105동 1007호

배송 메시지

비관적 락 동시성 테스트 중 (User B)

결제 수단

결제 수단 선택

신용카드 무통장입금

입금 정보

신한은행 123-456-789123 (주)LapPick 박희원

총 상품: 1종 총 수량: 1개

최종 결제 금액: 1,000,000원

결제하기

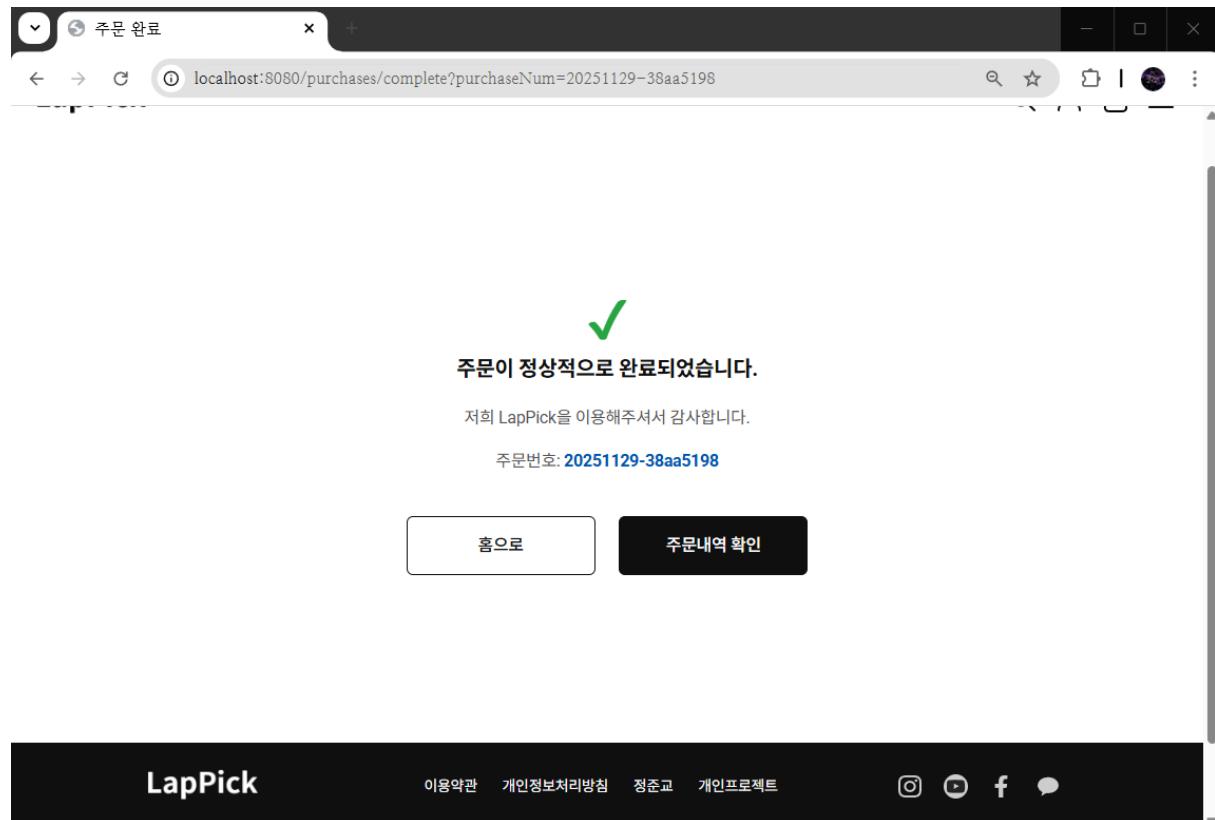
User B 주문 화면 - 왼쪽 상단 무한 로딩 아이콘 표시 (대기 중)

3-3. User A 주문 완료

Eclipse에서 F8(Resume) 눌러 User A 로직 계속 진행.

결과:

- User A: 주문 완료
- 주문번호: 20251129-38aa5198
- DB: 재고 1 → 0, 트랜잭션 커밋, Lock 해제



User A 주문 완료 화면 - "주문이 정상적으로 완료되었습니다"

3-4. User B 주문 실패

User A의 Lock이 해제되자 User B의 쿼리 실행됨.

결과:

- User B: 재고 0 확인 → 주문 실패
- 오류 메시지: "재고 부족: 'Eclipse테스트용노트북' 상품의 재고가 부족합니다. (요청: 1개, 현재: 0개)"

The screenshot shows a web browser window for 'LapPick' with the URL 'localhost:8080/cart/cartList'. The page displays a shopping cart interface. At the top, there are navigation links: BRAND, PRODUCTS, NOTICE, FAQ, and LOGOUT. On the right side, there are icons for search, user profile, and notifications (1). Below the header, a message says '선택하신 상품들을 한눈에 확인하고 수량을 조정하거나 구매할 수 있습니다.' (You can check the selected products at a glance, adjust the quantity, or purchase them). A pink warning box at the top left states '⚠ 주문 실패!' (Order Failed!) and '재고 부족: 'Eclipse테스트용노트북' 상품의 재고가 부족합니다. (요청: 1개, 현재: 0개)' (Stock不足: The requested product 'Eclipsetesztongnotebook' has insufficient stock. Requested: 1, Current: 0). The main content area shows a table of items in the cart. The first item listed is 'ASUS TUF Gaming A14' (ASUS TUF Gaming A14 FA401UV-RG025). The table columns include 이미지 (Image), 상품 이름 (Product Name), 수량 (Quantity), 상품금액 (Product Price), and 관리 (Management). The quantity for this item is set to 1. At the bottom of the page, there are summary statistics: '총 선택 상품: 1개' (Total selected products: 1), '총 수량: 1개' (Total quantity: 1), '총 상품 금액: 1,654,050 원' (Total product price: 1,654,050 won), and a large '구매하기' (Buy) button.

User B 주문 실패 화면 - 재고 부족 오류 메시지

4. DB 검증

테스트 완료 후 DB에서 최종 재고 및 주문 내역 확인.

-- 최종 재고 확인

```
SELECT SUM(IPGO_QTY) AS 최종재고  
FROM GOODS_IPGO  
WHERE GOODS_NUM = 'goods_999999';
```

결과: 0

-- 주문 건수 확인

```
SELECT COUNT(*) AS 주문건수  
FROM PURCHASE P  
JOIN PURCHASE_LIST PL ON P.PURCHASE_NUM = PL.PURCHASE_NUM  
WHERE PL.GOODS_NUM = 'goods_999999';
```

결과: 1

-- 주문 상세

```
SELECT P.PURCHASE_NUM, P.PURCHASE_DATE, P.MEMBER_NUM  
FROM PURCHASE P  
JOIN PURCHASE_LIST PL ON P.PURCHASE_NUM = PL.PURCHASE_NUM  
WHERE PL.GOODS_NUM = 'goods_999999';
```

PURCHASE_NUM	PURCHASE_DATE	MEMBER_NUM
20251129-38aa5198	25/11/29 00:36:48	mem_100041

The screenshot shows the Oracle SQL Developer environment. The top window is the '워크시트' (Worksheet) containing the SQL scripts for checking inventory and purchase details. The bottom window is the '스크립트 출력' (Script Output) showing the execution results:

- 최종재고:** 0
- 주문건수:** 1
- 주문 상세:**

PURCHASE_NUM	PURCHASE_DATE	MEMBER_NUM
20251129-38aa5198	25/11/29 00:36:48	mem_100041

DB 쿼리 실행 결과 - 최종 재고 0, 주문 1건

검증 결과:

- User A만 주문 성공
- User B는 재고 부족으로 실패
- 최종 재고 정확히 0

5. 기술 분석

Row-Level Lock 동작

[User A]

```
SELECT ... FOR UPDATE → Lock 획득  
주문 처리 (Breakpoint에서 대기)  
COMMIT → Lock 해제
```

[User B]

```
SELECT ... FOR UPDATE → Lock 대기 (Connection Blocking)  
(User A 완료 시까지 대기)  
Lock 획득 → 재고 0 확인 → 실패
```

Blocking 근거

1. User B 브라우저가 무한 로딩 상태 → 서버 응답 대기
2. Eclipse에 User B 요청이 Breakpoint 도달 안 함 → DB Connection 레벨에서 Blocking
3. User A 완료 직후 User B 진행 → Lock이 원인

6. WAIT 5 동작

쿼리:

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

동작:

- Lock 대기 최대 시간: 5초
- 5초 이내 Lock 획득 → 쿼리 실행
- 5초 초과 → ORA-30006 (resource busy) 예외 발생

본 테스트 결과:

- User A가 수동으로 Resume했으므로 처리 시간 < 5초
 - User B는 5초 이내 Lock 획득 → 재고 0 확인 → 비즈니스 로직 실패
 - Timeout 예외 발생하지 않음
-

7. 결론

Eclipse Breakpoint로 비관적 락의 순차 처리를 확인했다. User A가 Lock을 보유한 상태에서 User B는 DB Connection 레벨에서 대기했으며, User A 완료 후 User B가 재고 0을 확인하고 실패하여 오버셀링이 방지되었다.

트랜잭션 격리와 Row-Level Lock이 정상 동작함을 검증했다.