## REPEATABLE READ)

```
-- Создаем тестовый товар
INSERT INTO products (id, name, category id, description, price, rating, image url, stock)
VALUES
('550e8400-e29b-41d4-a716-446655440000', 'Test',
'550e8400-e29b-41d4-a716-446655440010', 'Test product', 0.00, 0, ", 1);
-- Транзакция 1
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
- Текущий остаток
SELECT stock FROM products
WHERE id = '550e8400-e29b-41d4-a716-446655440000'; -- 1
UPDATE products
SET stock = stock - 1
WHERE id = '550e8400-e29b-41d4-a716-446655440000';
-- Ждём
- Транзакция 2
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
- Текущий остаток
SELECT stock FROM products
WHERE id = '550e8400-e29b-41d4-a716-446655440000'; -- тоже 1
UPDATE products
SET stock = stock - 1
WHERE id = '550e8400-e29b-41d4-a716-446655440000';
-- Ждём
- Транзакция 1
СОММІТ; -- успешно
- Транзакция 2
СОММІТ: -- успешно
SELECT stock FROM products WHERE id = '550e8400-e29b-41d4-a716-446655440000';
-- stock = -1 | КОЛИЧЕСТВО НЕ МОЖЕТ БЫТЬ ОТРИЦАТЕЛЬНЫМ
```

## SERIALIZABLE) -- Сбрасываем состояние тестового товара **UPDATE** products SET stock = 1 WHERE id = '550e8400-e29b-41d4-a716-446655440000'; - Транзакция 1 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE; - Текущий остаток SELECT stock FROM products WHERE id = '550e8400-e29b-41d4-a716-446655440000'; -- 1 **UPDATE** products SET stock = stock - 1 **WHERE** id = '550e8400-e29b-41d4-a716-446655440000'; - Ждём - Транзакция 2 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE: -- Текущий остаток SELECT stock FROM products WHERE id = '550e8400-e29b-41d4-a716-446655440000'; -- тоже 1 **UPDATE** products SET stock = stock - 1 WHERE id = '550e8400-e29b-41d4-a716-446655440000'; -- Ждём - Транзакция 1

**COMMIT**; -- ERROR: could not serialize access due to concurrent update

**SELECT** stock **FROM** products **WHERE** id = '550e8400-e29b-41d4-a716-446655440000';

СОММІТ; -- успешно

- stock = 0 | все в порядке

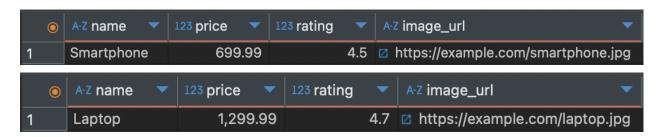
- Транзакция 2

## Задача 2

1) Получение товаров с пагинацией

```
SELECT name, price, rating, image_url
FROM products
WHERE category_id = '550e8400-e29b-41d4-a716-446655440010'
ORDER BY id
OFFSET 0 LIMIT 1;

SELECT name, price, rating, image_url
FROM products
WHERE category_id = '550e8400-e29b-41d4-a716-446655440010'
ORDER BY id
OFFSET 1 LIMIT 1;
```



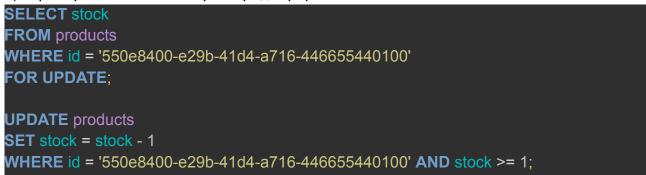
- а) Да. При уровне READ COMMITTED каждая выборка видит текущее состояние, поэтому при добавлении/удалении товаров между запросами страницы содержимое может "прыгать".
- б) Уровень REPEATABLE READ гарантирует один непрерывный снапшот данных на протяжении транзакции, что обеспечивает стабильную пагинацию.
- 2) Получение содержимого корзины и его итоговой суммы

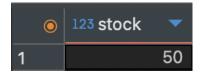
```
SELECT p.name product_name, c.quantity, p.price, ROUND(c.quantity * p.price, 2) price
FROM cart_items c
JOIN products p ON c.product_id = p.id
WHERE c.user_id = '550e8400-e29b-41d4-a716-446655440000';

SELECT SUM(c.quantity * p.price) AS total_price
FROM cart_items c
JOIN products p ON c.product_id = p.id
WHERE c.user_id = '550e8400-e29b-41d4-a716-446655440000';
```



- б) Минимально необходимый уровень изоляции REPEATABLE READ, так как он гарантирует, что все строки, прочитанные в рамках одной транзакции, не будут изменены или удалены другими транзакциями до ее завершения.
- 3) Проверка наличия товара перед оформлением заказа

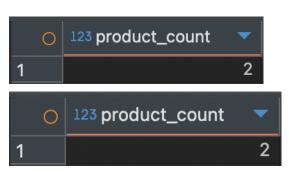




б) READ COMMITTED предотвратит overselling, т.к. FOR UPDATE ставит блокировку на строку товара до конца транзакции. Пока первая транзакция не завершилась, вторая будет ждать окончания COMMIT или ROLLBACK и только потом получит обновленное значение stock. Таким образом одновременно списать одну и ту же единицу товара в двух транзакциях невозможно.

## 4) Агрегирование данных





б) SERIALIZABLE даст точный, воспроизводимый результат, так как полностью исключает влияние других транзакций, не позволяя им вносить изменения в данные, используемые в текущей транзакции, до ее завершения.