

## Задача 1

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
COMMIT; -- успешно

-- Транзакция 2
COMMIT; -- 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; -- успешно

-- Транзакция 2
COMMIT; -- ERROR: could not serialize access due to concurrent update

SELECT stock FROM products WHERE id = '550e8400-e29b-41d4-a716-446655440000';
-- stock = 0 | все в порядке
```

## 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;
```

	A-Z name	123 price	123 rating	A-Z image_url
1	Smartphone	699.99	4.5	<a href="https://example.com/smartphone.jpg">https://example.com/smartphone.jpg</a>

	A-Z name	123 price	123 rating	A-Z image_url
1	Laptop	1,299.99	4.7	<a href="https://example.com/laptop.jpg">https://example.com/laptop.jpg</a>

- 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 снимок 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';
```

	A-Z product_name	123 quantity	123 price	123 price
1	Laptop	2	1,299.99	2,599.98

	123 total_price
1	2,599.98

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

```
SELECT stock
FROM products
WHERE id = '550e8400-e29b-41d4-a716-446655440100'
FOR UPDATE;

UPDATE products
SET stock = stock - 1
WHERE id = '550e8400-e29b-41d4-a716-446655440100' AND stock >= 1;
```

	123 stock ▼
1	50

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

```
SELECT COUNT(*) AS product_count
FROM products
WHERE category_id = '550e8400-e29b-41d4-a716-446655440010';

SELECT COUNT(*) AS product_count
FROM products
WHERE category_id = '550e8400-e29b-41d4-a716-446655440010';
```

	123 product_count ▼
1	2

	123 product_count ▼
1	2

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.