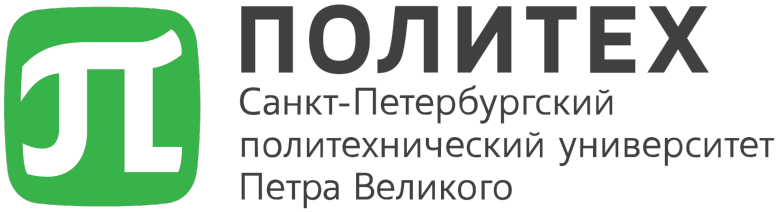
Федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»
Институт компьютерных наук и технологий
Программная инженерия



Отчёт по лабораторным работам

по дисциплине

«Базы данных»

Студент гр. 5130904/30108 Ребдев П.А
Проверил: Юркин В.А

Санкт-Петербург
2025

Оглавление

Лабораторная работа №3 (Вариант №3)

Представления

1. Создать представление, отображающее все товары, число которых на складе1 менее некоторого числа

```
first=# CREATE VIEW firstView As
SELECT goods.id, goods.name, goods.priority, warehouse1.good_count FROM goods
LEFT join warehouse1 ON warehouse1.good_id = goods.id
where warehouse1.good_count > 10;
CREATE VIEW
first=# SELECT * FROM firstview;
 id |                  name                  | priority | good_count
----+----------------------------------------+----------+------------
  1 | Процессор Bintel                       |        1 |        100
  2 | Процессор AMV                          |        1 |        100
  3 | Видеокарта NVIDIO                      |        1 |         50
  4 | Видеокарта RAZEN                       |        1 |         40
  5 | Материнская плата ASPAPER              |        2 |         30
  6 | Материнская плата TERABYTE             |        2 |         30
  7 | Оперативная память ARDOR WORKING       |        2 |         60
  8 | Оперативная память QUEENSTON           |        2 |         67
 20 | Диски с операционной системой TempleOS |        5 |        400
 28 | TEST_DATA                              |      100 |        100
 29 | TEST_DATA_2                            |      100 |        100
(11 строк)
```

2. Создать представление, отображающее 5 самых популярных товаров за заданный месяц.

```
first=# CREATE VIEW secondView AS
SELECT goods.id, goods.name, SUM(sales.good_count) as total_sold FROM goods
JOIN sales ON goods.id = sales.good_id
WHERE( extract(month FROM sales.create_date) = '9')
GROUP BY goods.id, goods.name
LIMIT 5;
CREATE VIEW
first=# SELECT * FROM s
sales         sales_id_seq  secondview
first=# SELECT * FROM secondview;
 id |           name            | total_sold
----+---------------------------+------------
  1 | Процессор Bintel          |         15
  2 | Процессор AMV             |         33
  3 | Видеокарта NVIDIO         |          6
  4 | Видеокарта RAZEN          |         16
  5 | Материнская плата ASPAPER |          3
(5 строк)
```

Hints: select, where, count, max, group by, having, like, create view, drop view

Хранимые процедуры

Перед созданием хранимых процедур изучите разницу между хранимыми функциями и процедурами в PostgreSQL. Обоснуйте на основе материалов лекций почему требуется создание именно хранимых процедур.

- без параметров

1. Создать хранимую процедуру, выводящую список товаров для перевоза и его количество согласно текущему состоянию приоритетов.

```
CREATE OR REPLACE PROCEDURE get_goods_for_transfer()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
BEGIN
  RAISE NOTICE 'Список товаров для перевозки:';
  RAISE NOTICE '--------------------------------';

  FOR rec IN
    SELECT
      goods.id,
      goods.name,
      goods.priority,
      GREATEST(
        LEAST(
          (goods.priority * 10) - COALESCE(warehouse1.good_count, 0),
          COALESCE(warehouse2.good_count, 0)
        ),
        0
      ) AS amount_to_transfer
    FROM
      goods
    LEFT JOIN warehouse1 ON goods.id = warehouse1.good_id
    LEFT JOIN warehouse2 ON goods.id = warehouse2.good_id
    WHERE
      COALESCE(warehouse2.good_count, 0) > 0
      AND (COALESCE(warehouse1.good_count, 0) < (goods.priority * 10))
      AND GREATEST(
        LEAST(
          (goods.priority * 10) - COALESCE(warehouse1.good_count, 0),
          COALESCE(warehouse2.good_count, 0)
        ),
        0
      ) > 0
    ORDER BY
      goods.priority DESC
  LOOP
    RAISE NOTICE 'Товар: %, Приоритет: %, Количество для перевозки: %',
      rec.name, rec.priority, rec.amount_to_transfer;
  END LOOP;
```

```
  IF NOT FOUND THEN
    RAISE NOTICE 'Нет товаров для перевозки';
  END IF;
END;
$$;
```

```
first=# call get_goods_for_transfer();
ЗАМЕЧАНИЕ:  Список товаров для перевозки:
ЗАМЕЧАНИЕ:  --------------------------------
ЗАМЕЧАНИЕ:  Товар: TEST_DATA, Приоритет: 100, Количество для перевозки: 100
ЗАМЕЧАНИЕ:  Товар: TEST_DATA_2, Приоритет: 100, Количество для перевозки: 100
ЗАМЕЧАНИЕ:  Товар: Коврик для мыши ARDOR WORKING, Приоритет: 6, Количество для перевозки: 50
ЗАМЕЧАНИЕ:  Товар: Диски с операционной системой Шиндовс, Приоритет: 5, Количество для перевозки: 50
ЗАМЕЧАНИЕ:  Товар: Наушники ARDOR WORKING, Приоритет: 4, Количество для перевозки: 40
ЗАМЕЧАНИЕ:  Товар: Наушники EARTHPODS, Приоритет: 4, Количество для перевозки: 40
ЗАМЕЧАНИЕ:  Товар: Монитор ARDOR WORKING, Приоритет: 3, Количество для перевозки: 30
ЗАМЕЧАНИЕ:  Товар: Мышь GOGITECH, Приоритет: 3, Количество для перевозки: 30
ЗАМЕЧАНИЕ:  Товар: Клавиатура ARDOR WORKING, Приоритет: 3, Количество для перевозки: 30
ЗАМЕЧАНИЕ:  Товар: Клавиатура BEXP, Приоритет: 3, Количество для перевозки: 30
ЗАМЕЧАНИЕ:  Товар: Мышь ARDOR WORKING, Приоритет: 3, Количество для перевозки: 30
ЗАМЕЧАНИЕ:  Товар: Монитор MSA, Приоритет: 3, Количество для перевозки: 30
CALL
```

- с входными параметрами
    1. Создать хранимую процедуру с параметром количество перевозимого товара за ближайший рейс и выводящую все товары, которые необходимо привезти, и их количество.

```
CREATE OR REPLACE PROCEDURE get_goods_for_transfer_with_capacity(
  IN max_transport_capacity INTEGER
)
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  remaining_capacity INTEGER;
  transfer_amount INTEGER;
BEGIN
  RAISE NOTICE 'Список товаров для перевозки (лимит: %):', max_transport_capacity;
  RAISE NOTICE '--------------------------------';

  remaining_capacity := max_transport_capacity;

  FOR rec IN
    SELECT
      goods.id,
      goods.name,
      goods.priority,
      COALESCE(warehouse1.good_count, 0) as current_warehouse1_count,
      COALESCE(warehouse2.good_count, 0) as current_warehouse2_count,
      (goods.priority * 10) - COALESCE(warehouse1.good_count, 0) as needed_amount
    FROM
      goods
    LEFT JOIN warehouse1 ON goods.id = warehouse1.good_id
    LEFT JOIN warehouse2 ON goods.id = warehouse2.good_id
    WHERE
      COALESCE(warehouse2.good_count, 0) > 0
      AND (COALESCE(warehouse1.good_count, 0) < (goods.priority * 10))
    ORDER BY
```

```
        goods.priority DESC
    LOOP
        EXIT WHEN remaining_capacity <= 0;

        transfer_amount := LEAST(
            rec.needed_amount,
            rec.current_warehouse2_count,
            remaining_capacity
        );

        IF transfer_amount > 0 THEN
            RAISE NOTICE 'Товар: %, Приоритет: %, Количество для перевозки: %',
                rec.name, rec.priority, transfer_amount;

            remaining_capacity := remaining_capacity - transfer_amount;
        END IF;
    END LOOP;

    IF remaining_capacity = max_transport_capacity THEN
        RAISE NOTICE 'Нет товаров для перевозки';
    ELSIF remaining_capacity > 0 THEN
        RAISE NOTICE 'Остаток свободной емкости: %', remaining_capacity;
    ELSE
        RAISE NOTICE 'Лимит перевозки полностью использован';
    END IF;
END;
$$;
first=# CALL get_goods_for_transfer_with_capacity(50);
ЗАМЕЧАНИЕ:  Список товаров для перевозки (лимит: 50):
ЗАМЕЧАНИЕ:  --------------------------------
ЗАМЕЧАНИЕ:  Товар: TEST_DATA, Приоритет: 100, Количество для перевозки: 50
ЗАМЕЧАНИЕ:  Лимит перевозки полностью использован
CALL
first=# CALL get_goods_for_transfer_with_capacity(400);
ЗАМЕЧАНИЕ:  Список товаров для перевозки (лимит: 400):
ЗАМЕЧАНИЕ:  --------------------------------
ЗАМЕЧАНИЕ:  Товар: TEST_DATA, Приоритет: 100, Количество для перевозки: 100
ЗАМЕЧАНИЕ:  Товар: TEST_DATA_2, Приоритет: 100, Количество для перевозки: 100
ЗАМЕЧАНИЕ:  Товар: Коврик для мыши ARDOR WORKING, Приоритет: 6, Количество для перевозки: 50
ЗАМЕЧАНИЕ:  Товар: Диски с операционной системой Шиндовс, Приоритет: 5, Количество для перевозки: 50
ЗАМЕЧАНИЕ:  Товар: Наушники ARDOR WORKING, Приоритет: 4, Количество для перевозки: 40
ЗАМЕЧАНИЕ:  Товар: Наушники EARTHPODS, Приоритет: 4, Количество для перевозки: 40
ЗАМЕЧАНИЕ:  Товар: Монитор ARDOR WORKING, Приоритет: 3, Количество для перевозки: 20
ЗАМЕЧАНИЕ:  Лимит перевозки полностью использован
CALL
```

2. Создать хранимую процедуру, имеющую два параметра «товар1» и «товар2». Она должна возвращать даты, спрос в которые на «товар1» был больше чем на «товар2». Если в какой-либо день один из товаров не продавался, такой день не рассматривается.

```
CREATE OR REPLACE PROCEDURE get_dates_with_higher_demand(
    IN product1_id INTEGER,
    IN product2_id INTEGER
)
LANGUAGE plpgsql
AS $$
```

```
DECLARE
  date_record RECORD;
BEGIN
  RAISE NOTICE 'Даты, когда спрос на товар % был выше, чем на товар %:', product1_id, product2_id;
  RAISE NOTICE '-------------------------------------------------';

  FOR date_record IN
    SELECT
      sales1.create_date
    FROM
      (SELECT create_date, SUM(good_count) as total_count
       FROM sales
       WHERE good_id = product1_id
       GROUP BY create_date) as sales1
    JOIN
      (SELECT create_date, SUM(good_count) as total_count
       FROM sales
       WHERE good_id = product2_id
       GROUP BY create_date) as sales2
    ON sales1.create_date = sales2.create_date
    WHERE
      sales1.total_count > sales2.total_count
    ORDER BY
      sales1.create_date
  LOOP
    RAISE NOTICE 'Дата: %', date_record.create_date;
  END LOOP;

  IF NOT FOUND THEN
    RAISE NOTICE 'Нет дат, когда спрос на товар % был выше, чем на товар %', product1_id, product2_id;
  END IF;
END;
$$;
```

```
first=# CALL get_dates_with_higher_demand(8, 5);
ЗАМЕЧАНИЕ:  Даты, когда спрос на товар 8 был выше, чем на товар 5:
ЗАМЕЧАНИЕ:  -------------------------------------------------
ЗАМЕЧАНИЕ:  Дата: 2025-09-03
CALL
```

- с выходными параметрами
    1. Создать хранимую процедуру с входными параметрами, задающими интервал времени, и выходным – идентификатором товара. Процедура должна возвращать товар с максимальным приростом спроса.

```
CREATE OR REPLACE PROCEDURE get_product_with_max_demand_increase(
  IN start_date DATE,
  IN end_date DATE,
  OUT product_id INTEGER
)
LANGUAGE plpgsql
AS $$
DECLARE
```

```
  max_increase_product INTEGER;
BEGIN
  SELECT goods.id INTO product_id
  FROM goods
  JOIN (
    SELECT
      good_id,
      SUM(CASE WHEN create_date >= start_date AND create_date <= end_date THEN good_count ELSE 0 END) -
      SUM(CASE WHEN create_date < start_date THEN good_count ELSE 0 END) as demand_increase
    FROM sales
    GROUP BY good_id
    ORDER BY demand_increase DESC
    LIMIT 1
  ) sales_stats ON goods.id = sales_stats.good_id;

  IF product_id IS NULL THEN
    RAISE NOTICE 'Не удалось определить товар с максимальным приростом спроса';
  ELSE
    RAISE NOTICE 'Товар с максимальным приростом спроса: %', product_id;
  END IF;
END;
$$;
first=# CALL get_product_with_max_demand_increase('2025-09-01', '2025-09-25', NULL);
ЗАМЕЧАНИЕ:  Товар с максимальным приростом спроса: 17
 product_id
------------
         17
(1 строка)
```

2. Создать хранимую процедуру с входными параметрами, задающими интервал времени, и выходным параметром – товар, с минимальным числом продаж за заданный период времени.

```
CREATE OR REPLACE PROCEDURE get_product_with_min_sales(
  IN start_date DATE,
  IN end_date DATE,
  OUT product_id INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
  SELECT sales.good_id INTO product_id
  FROM sales
  WHERE sales.create_date BETWEEN start_date AND end_date
  GROUP BY sales.good_id
  ORDER BY SUM(sales.good_count) ASC
  LIMIT 1;

  IF product_id IS NULL THEN
    RAISE NOTICE 'В указанный период не было продаж';
  ELSE
    RAISE NOTICE 'Товар с минимальным числом продаж: %', product_id;
  END IF;
END;
```

```
$$;
first=# CALL get_product_with_min_sales('2025-09-01', '2025-09-25', NULL);
ЗАМЕЧАНИЕ:  Товар с минимальным числом продаж: 5
 product_id
------------
          5
(1 строка)
```

Hints: select, where, count, max, group by, having, create procedure, drop procedure
Разберите возможность выбора языка sql и plpgsql. На чем основывается Ваш выбор?

Триггера

- Триггера на вставку
    1. Создать триггер, который не позволяет добавить заявку на товар, число которого на обоих складах меньше указанного в заявке

```
CREATE OR REPLACE FUNCTION check_warehouse_stock()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
   warehouse1_count INTEGER;
   warehouse2_count INTEGER;
   total_stock INTEGER;
BEGIN
   SELECT COALESCE(good_count, 0) INTO warehouse1_count
   FROM warehouse1
   WHERE good_id = NEW.good_id;

   SELECT COALESCE(good_count, 0) INTO warehouse2_count
   FROM warehouse2
   WHERE good_id = NEW.good_id;

   total_stock := warehouse1_count + warehouse2_count;

   IF total_stock < NEW.good_count THEN
      RAISE EXCEPTION 'Недостаточно товара на складах. Требуется: %, доступно: %', NEW.good_count, total_stock;
   END IF;

   RETURN NEW;
END;
$$;

CREATE TRIGGER prevent_insufficient_stock_order
   BEFORE INSERT ON sales
   FOR EACH ROW
   EXECUTE FUNCTION check_warehouse_stock();
first=# INSERT INTO sales VALUES (DEFAULT, 5, 200, CURRENT_DATE);
ОШИБКА:  Недостаточно товара на складах. Требуется: 200, доступно: 98
КОНТЕКСТ:  функция PL/pgSQL check_warehouse_stock(), строка 18, оператор RAISE
```

2. Создать триггер, который не позволяет добавить заявку с числом товара меньше 1.

```
CREATE OR REPLACE FUNCTION check_order_quantity()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  IF NEW.good_count < 1 THEN
    RAISE EXCEPTION 'Количество товара в заявке не может быть меньше 1';
  END IF;

  RETURN NEW;
END;
$$;

CREATE TRIGGER prevent_invalid_order_quantity
  BEFORE INSERT ON sales
  FOR EACH ROW
  EXECUTE FUNCTION check_order_quantity();
```

```
first=# INSERT INTO sales VALUES (DEFAULT, 5, -14, CURRENT_DATE);
ОШИБКА:  Количество товара в заявке не может быть меньше 1
КОНТЕКСТ:  функция PL/pgSQL check_order_quantity(), строка 4, оператор RAISE
```

- Триггера на модификацию
    1. Создать триггер, который не позволяет уменьшить число товара на складе 2 при наличии этого товара на складе 1.

```
CREATE OR REPLACE FUNCTION prevent_warehouse2_decrease()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
  warehouse1_count INTEGER;
BEGIN
  SELECT COALESCE(good_count, 0) INTO warehouse1_count
  FROM warehouse1
  WHERE good_id = NEW.good_id;

  IF OLD.good_count > NEW.good_count AND warehouse1_count > 0 THEN
    RAISE EXCEPTION 'Нельзя уменьшить количество товара на складе 2 при наличии товара на складе 1';
  END IF;

  RETURN NEW;
END;
$$;

CREATE TRIGGER prevent_warehouse2_decrease_trigger
  BEFORE UPDATE ON warehouse2
  FOR EACH ROW
  EXECUTE FUNCTION prevent_warehouse2_decrease();
first=# UPDATE warehouse2 SET good_count = 40 WHERE good_id = 1;
ОШИБКА:  Нельзя уменьшить количество товара на складе 2 при наличии товара на складе 1
КОНТЕКСТ:  функция PL/pgSQL prevent_warehouse2_decrease(), строка 10, оператор RAISE
```

- Триггера на удаление
    1. Создать триггер, который при удалении товара в случае наличия на него ссылок откатывает транзакцию

```
CREATE OR REPLACE FUNCTION prevent_good_deletion()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
   sales_count INTEGER;
   warehouse1_count INTEGER;
   warehouse2_count INTEGER;
BEGIN
   SELECT COUNT(*) INTO sales_count FROM sales WHERE good_id = OLD.id;
   SELECT COUNT(*) INTO warehouse1_count FROM warehouse1 WHERE good_id = OLD.id;
   SELECT COUNT(*) INTO warehouse2_count FROM warehouse2 WHERE good_id = OLD.id;

   IF sales_count > 0 OR warehouse1_count > 0 OR warehouse2_count > 0 THEN
      RAISE EXCEPTION 'Невозможно удалить товар с ID %, так как на него есть ссылки в других таблицах', OLD.id;
   END IF;

   RETURN OLD;
END;
$$;

CREATE TRIGGER prevent_good_deletion_trigger
   BEFORE DELETE ON goods
   FOR EACH ROW
   EXECUTE FUNCTION prevent_good_deletion();
```

```
first=# DELETE FROM goods WHERE id = 1;
ОШИБКА:  Невозможно удалить товар с ID 1, так как на него есть ссылки в других таблицах
КОНТЕКСТ:  функция PL/pgSQL prevent_good_deletion(), строка 12, оператор RAISE
```

Hints: select, where, in, exists, join, commit, rollback, create trigger, drop trigger

Разберите особенности работы триггера FOR EACH STATEMENT/ FOR EACH ROW

Курсоры

- Хранимая процедура для прогноза спроса на заданный товар

Необходимо реализовать хранимую процедуру, рассчитывающую прогнозируемый спрос на 7 дней на некоторый товар. Хранимая процедура должна иметь два входных параметра задающие интервал времени для анализа изменения спроса и параметр, задающий анализируемый товар.

Предлагаемый алгоритм: с помощью курсора формируем временную таблицу, содержащую номер дня в рассматриваемом интервале и число хранящегося товара. Максимальный номер сохраняем в переменной.

Организуем курсор, перебирающий последовательно строки временной таблицы, упорядоченные по номеру точки, и усредняющий попарно соседние точки (за каждую итерацию точек становится на 1 меньше). Полученное среднее значение каждой пары, заменяет значение минимальной по номеру точки. Последняя точка удаляется.

Работа курсора повторяется до тех пор, пока в таблице не останется 2 точки. Будем считать, что разница спроса в этих двух точках равна разнице спроса между прогнозируемым на следующий день спросом и спросом на заданный товар последнего дня. Таким образом, мы можем получить величину прогнозируемого спроса. (экстраполяция методом скользящего среднего)

```
CREATE OR REPLACE PROCEDURE forecast_demand_7_days(
    p_start_date DATE,
    p_end_date DATE,
    p_good_id INTEGER
)
LANGUAGE plpgsql
AS $$
DECLARE
    temp_table_rec RECORD;
    point1_value NUMERIC;
    point2_value NUMERIC;
    point1_day INTEGER;
    point2_day INTEGER;
    last_day_num INTEGER;
    current_count INTEGER;
    avg_value NUMERIC;
    slope NUMERIC;
    forecast_day INTEGER;
    forecast_demand NUMERIC;

    data_cursor CURSOR FOR
        SELECT
            ROW_NUMBER() OVER (ORDER BY create_date) as day_num,
            SUM(good_count) as daily_demand
        FROM sales
        WHERE good_id = p_good_id
            AND create_date BETWEEN p_start_date AND p_end_date
        GROUP BY create_date
        ORDER BY create_date;

    smooth_cursor REFCURSOR;
BEGIN
    CREATE TEMP TABLE temp_demand_data (
        day_num INTEGER,
        demand NUMERIC
    );

    INSERT INTO temp_demand_data (day_num, demand)
    SELECT
        ROW_NUMBER() OVER (ORDER BY create_date),
        SUM(good_count)
    FROM sales
```

```
WHERE good_id = p_good_id
  AND create_date BETWEEN p_start_date AND p_end_date
GROUP BY create_date
ORDER BY create_date;

SELECT MAX(day_num) INTO last_day_num FROM temp_demand_data;

WHILE last_day_num > 2 LOOP
  OPEN smooth_cursor FOR
    SELECT day_num, demand
    FROM temp_demand_data
    ORDER BY day_num;

  DELETE FROM temp_demand_data;

  FETCH smooth_cursor INTO point1_day, point1_value;
  WHILE FOUND LOOP
    FETCH smooth_cursor INTO point2_day, point2_value;
    IF FOUND THEN
      avg_value := (point1_value + point2_value) / 2;

      INSERT INTO temp_demand_data (day_num, demand)
      VALUES (point1_day, avg_value);

      point1_day := point2_day;
      point1_value := point2_value;
    END IF;
  END LOOP;

  CLOSE smooth_cursor;

  SELECT MAX(day_num) INTO last_day_num FROM temp_demand_data;
END LOOP;

SELECT demand INTO point1_value
FROM temp_demand_data
ORDER BY day_num ASC
LIMIT 1;

SELECT demand INTO point2_value
FROM temp_demand_data
ORDER BY day_num ASC
LIMIT 1 OFFSET 1;

slope := point2_value - point1_value;

RAISE NOTICE 'Прогноз спроса на товар % на 7 дней:', p_good_id;
RAISE NOTICE 'День | Прогнозируемый спрос';
RAISE NOTICE '-----|-------------------';

FOR forecast_day IN 1..7 LOOP
  forecast_demand := point2_value + (slope * forecast_day);
  RAISE NOTICE '%    | %', forecast_day, ROUND(forecast_demand, 2);
END LOOP;

DROP TABLE temp_demand_data;
```

```
END;
$$;
```

```
CALL forecast_demand_7_days('2025-09-01', '2025-09-30', 2);
ЗАМЕЧАНИЕ:  Прогноз спроса на товар 2 на 7 дней:
ЗАМЕЧАНИЕ:  День | Прогнозируемый спрос
ЗАМЕЧАНИЕ:  -----|-------------------
ЗАМЕЧАНИЕ:  1    | 24.00
ЗАМЕЧАНИЕ:  2    | 29.00
ЗАМЕЧАНИЕ:  3    | 34.00
ЗАМЕЧАНИЕ:  4    | 39.00
ЗАМЕЧАНИЕ:  5    | 44.00
ЗАМЕЧАНИЕ:  6    | 49.00
ЗАМЕЧАНИЕ:  7    | 54.00
```

Hints: CURSOR, %NOTFOUND, FETCH