Санкт-Петербургский государственный политехнический университет Институт компьютерных наук и кибербезопасности

**Кафедра «Программная инженерия»**

**ЛАБОРАТОРНАЯ РАБОТА 3**

По дисциплине «Базы данных»

Выполнил студент гр. 5130904/10106 Сафронов. А.А.

Руководитель

проф. д.т.н. О.В. Александрова

Санкт-Петербург

2023

**Отчет по работе с базой данных PostgreSQL**

### Техническое задание.

### Представления

1. Создать представление, отображающее все услуги, по которым за все время сумма превысила некоторую границу
2. Создать представление, отображающее общий доход мастеров за последний год, включая мастеров, которые ничего не получили

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

### Хранимые процедуры

* без параметров

1. Создать хранимую процедуру, выводящую все автомобили и среднюю стоимость услуги для них, включая автомобили, по которым не производились работы

* с входными параметрами

1. Создать хранимую процедуру, имеющую два параметра «услуга» и «машина». Она должна возвращать общую стоимость этой услуги для машины за все время существования автосервиса и количество проводимых по этой услуге работ.

* с выходными параметрами

1. Создать хранимую процедуру с входными параметрами «мастер1» и «мастер2» и выводящую количество услуг, которые оказывали оба мастера.

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

### Триггера

* Триггера на вставку

1. Создать триггер, который не позволяет добавить автомобиль с уже существующим номером

* Триггера на модификацию

1. Создать триггер, который не позволяет изменить дату работы более чем на один день

* Триггера на удаление

1. Создать триггер, который при удалении автомобиля, в случае если по нему были какие-то работы, откатывает транзакцию

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

### Курсоры

* Хранимая процедура для распределения премии мастерам за период

Необходимо реализовать хранимую процедуру, рассчитывающую зарплату+премию, полученную мастером за текущий месяц. Хранимая процедура должна иметь два входных параметра: мастера, для которого производим расчет и его зарплата и один выходной, в котором возвращать рассчитанную сумму.

Предлагаемый алгоритм: создаем курсор, который пробегает по работам для данного мастера.

Для каждой строки проверяем сколько раз для данной машины в заданном периоде оказывались услуги. Если меньше 3-х, то премия составляет 5% от з/п, если больше, то 7% (по каждой машине). Суммируем полученный результат в некоторой переменной, значение которой по окончании работы курсора будет выдано в качестве выходного параметра.

**Документация**

**Представления**

1. **Представление услуг с суммой выше границы**

Я создал представление, которое отображает все услуги, сумма по которым превышает определенную границу. Это полезно для быстрого просмотра наиболее доходных услуг.

**SQL код**:

sql

 CREATE VIEW high\_cost\_services AS

SELECT service\_id, SUM(cost) AS total\_cost

FROM works

GROUP BY service\_id

HAVING SUM(cost) > 1000;

 **Представление доходов мастеров за последний год**

Это представление позволяет увидеть общий доход каждого мастера за последний год, включая тех, кто не имеет дохода. Это полезно для анализа эффективности работы мастеров.

**SQL код**:

sql

1. CREATE VIEW master\_income\_last\_year AS
2. SELECT master\_id, COALESCE(SUM(cost), 0) AS total\_income
3. FROM works
4. WHERE date\_work BETWEEN (CURRENT\_DATE - INTERVAL '1 year') AND CURRENT\_DATE
5. GROUP BY master\_id;

**Хранимые процедуры**

1. **Средняя стоимость услуг по автомобилям**

Я создал хранимую процедуру, которая выводит среднюю стоимость услуг для каждого автомобиля. Это полезно для анализа стоимости обслуживания разных типов автомобилей.

**SQL код**:

sql

 CREATE OR REPLACE PROCEDURE avg\_service\_cost\_per\_car()

LANGUAGE SQL

AS $$

SELECT car\_id, AVG(cost) AS avg\_cost

FROM works

GROUP BY car\_id;

$$;

 **Стоимость услуги для машины**

Эта хранимая процедура принимает два параметра: ID услуги и ID машины. Она возвращает общую стоимость этой услуги для данной машины и количество выполненных работ.

**SQL код**:

sql

 CREATE OR REPLACE FUNCTION calculate\_service\_cost\_for\_car(p\_service\_id INT, p\_car\_id INT)

RETURNS TABLE (total\_cost NUMERIC, total\_works BIGINT)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY SELECT SUM(cost), COUNT(\*)

FROM works

WHERE service\_id = p\_service\_id AND car\_id = p\_car\_id;

END;

$$;

 **Общее количество услуг между двумя мастерами**

Эта хранимая процедура принимает два параметра: ID первого мастера и ID второго мастера. Она возвращает количество услуг, которые оба мастера предоставляли.

**SQL код**:

sql

1. CREATE OR REPLACE FUNCTION get\_common\_services\_count(p\_master1\_id INT, p\_master2\_id INT, OUT common\_services\_count INT)
2. LANGUAGE plpgsql
3. AS $$
4. BEGIN
5. SELECT COUNT(DISTINCT service\_id)
6. INTO common\_services\_count
7. FROM works
8. WHERE master\_id IN (p\_master1\_id, p\_master2\_id)
9. GROUP BY master\_id
10. HAVING COUNT(DISTINCT service\_id) > 1;
11. END;
12. $$;

**Триггеры**

1. **Проверка уникальности номера автомобиля**

Я создал триггер, который не позволяет добавить автомобиль с уже существующим номером. Это предотвращает дублирование данных.

**SQL код**:

sql

 CREATE TRIGGER check\_unique\_car\_number

BEFORE INSERT ON cars

FOR EACH ROW EXECUTE FUNCTION check\_unique\_car\_number();

 **Ограничение на изменение даты**

Этот триггер не позволяет изменить дату работы более чем на один день. Это помогает поддерживать целостность данных.

**SQL код**:

sql

 CREATE TRIGGER check\_date\_change

BEFORE UPDATE ON works

FOR EACH ROW EXECUTE FUNCTION check\_date\_change();

 **Откат транзакции при наличии работ**

Я создал триггер, который откатывает транзакцию удаления автомобиля, если по этому автомобилю были выполнены какие-либо работы. Это предотвращает потерю важных данных.

**SQL код**:

sql

1. CREATE TRIGGER prevent\_delete\_if\_works\_exist
2. BEFORE DELETE ON cars
3. FOR EACH ROW EXECUTE FUNCTION prevent\_delete\_if\_works\_exist();

**Курсоры**

1. **Распределение премии мастерам**

Я создал хранимую процедуру, которая распределяет премии мастерам на основе количества выполненных ими работ. Это полезно для мотивации персонала.

**SQL код**:

sql

CREATE OR REPLACE FUNCTION calculate\_bonus(p\_master\_id INT, p\_base\_salary NUMERIC)

RETURNS NUMERIC

LANGUAGE plpgsql

AS $$

DECLARE

v\_bonus NUMERIC := 0;

v\_service\_count INT;

cur CURSOR FOR SELECT car\_id FROM works WHERE master\_id = p\_master\_id;

BEGIN

FOR rec IN cur LOOP

SELECT COUNT(\*) INTO v\_service\_count FROM works WHERE car\_id = rec.car\_id AND master\_id = p\_master\_id;

IF v\_service\_count < 3 THEN

v\_bonus := v\_bonus + (p\_base\_salary \* 0.05);

ELSE

v\_bonus := v\_bonus + (p\_base\_salary \* 0.07);

END IF;

END LOOP;

RETURN p\_base\_salary + v\_bonus;

END;

$$;