МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 11

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

Козик И. Д.

Брест – 2024

**Ход работы**

Измените структуру любой из таблиц вашей базы данных, добавив числовое поле «Примечание» и заполнив его произвольными дробными различными значениями в диапазоне от 50 до 300. Выберите две такие таблицы, которые связаны между собой еще через две другие (есть цепочка из четырех таблиц), одна из которых содержит столбец «Примечание».

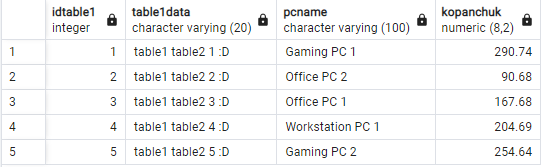
SELECT table1.idtable1, table1.table1Data, pc.pcname, cpu.kopanchuk

FROM table1

JOIN table2 ON table1.table2id = table2.idTable2

JOIN cpu ON table2.cpuid = cpu.cpuID

JOIN pc ON cpu.cpuID = pc.cpuID;



*Процедуры*

Используйте цепочку из четырех связанных таблиц. Создайте процедуры, которые по заданному пользователем диапазону для значений первичного ключа из первой в цепочке таблицы определят все соответствующие данные из последней таблицы в цепочке:

* процедура 1 – без входных параметров;
* процедура 2 – с входными параметрами;
* процедура 3 – с выходным параметром. Выведите сообщение о количестве строк в искомом результате.

Осуществите вызов каждой из процедур.

CREATE FUNCTION find\_data\_from\_first\_to\_last()

RETURNS TABLE (

pcID INT,

pcName VARCHAR(100),

cpuID INT,

gpuID INT,

ramID INT,

motherboardID INT

) AS $$

DECLARE

row\_table1 RECORD;

BEGIN

FOR row\_table1 IN SELECT table2id FROM table1 LOOP

RETURN QUERY

SELECT pc.\*

FROM pc

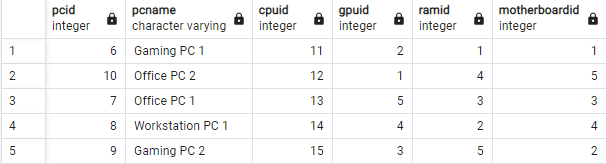
WHERE pc.cpuID = (SELECT t.cpuID FROM (SELECT \* FROM table2 WHERE table2.idTable2 = row\_table1.table2id) AS t);

END LOOP;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM find\_data\_from\_first\_to\_last();



CREATE OR REPLACE FUNCTION find\_data\_from\_first\_to\_last\_by\_table1id(idTable1\_param INT)

RETURNS TABLE (

pcID INT,

pcName VARCHAR(100),

cpuID INT,

gpuID INT,

ramID INT,

motherboardID INT

) AS $$

BEGIN

RETURN QUERY

SELECT pc.\*

FROM pc

WHERE pc.cpuID = (SELECT table2.cpuID FROM table2 WHERE table2.idtable2 = (SELECT table1.table2id FROM table1 WHERE table1.idtable1 = idTable1\_param));

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM find\_data\_from\_first\_to\_last\_by\_table1id(1);



CREATE OR REPLACE FUNCTION find\_data\_from\_first\_to\_last\_count()

RETURNS INT AS $$

DECLARE

row\_count INT := 0;

row\_table1 RECORD;

BEGIN

FOR row\_table1 IN SELECT table2id FROM table1 LOOP

row\_count := row\_count + 1;

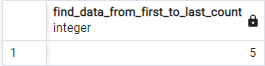
END LOOP;

RETURN row\_count;

END;

$$ LANGUAGE plpgsql;

SELECT find\_data\_from\_first\_to\_last\_count();



*Триггеры*

Создайте два триггера к любой из таблиц вашей базы данных для следующих операций:

1. вставка, изменение;
2. удаление.

Для каждого триггера продемонстрируйте выполнение (с помощью скриншотов), вывод сообщений об ошибке, поведение системы при отключении одного триггера.

CREATE OR REPLACE FUNCTION prevent\_insert\_update\_table1()

RETURNS TRIGGER AS $$

BEGIN

RAISE EXCEPTION 'Inserts and updates are not allowed on table1';

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent\_insert\_update\_trigger

BEFORE INSERT OR UPDATE ON table1

FOR EACH ROW

EXECUTE PROCEDURE prevent\_insert\_update\_table1();

CREATE OR REPLACE FUNCTION prevent\_delete\_table1()

RETURNS TRIGGER AS $$

BEGIN

RAISE EXCEPTION 'Deletes are not allowed on table1';

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

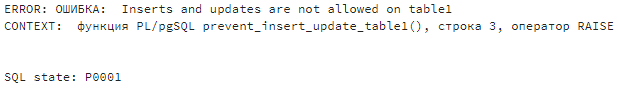
CREATE TRIGGER prevent\_delete\_trigger

BEFORE DELETE ON table1

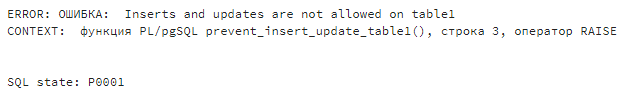
FOR EACH ROW

EXECUTE PROCEDURE prevent\_delete\_table1();

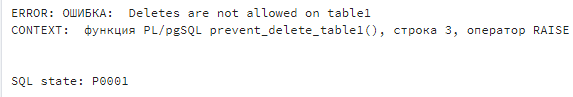
INSERT INTO table1 (table1Data, table2id) VALUES ('table1 triggers 0\_0', 1);



UPDATE table1 SET table1Data = 'table1 update x\_x' WHERE idTable1 = 1;



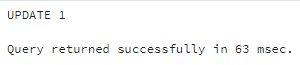
DELETE FROM table1 WHERE idTable1 = 1;



ALTER TABLE table1 DISABLE TRIGGER prevent\_insert\_update\_trigger;

INSERT INTO table1 (table1Data, table2id) VALUES ('table1 triggers 0\_0', 1);

UPDATE table1 SET table1Data = 'table1 update x\_x' WHERE idTable1 = 1;



*Функции*

Создайте функции, которые возвращают результаты, соответствующие запросам 1, 2, 3 лабораторной работы №10 (пункт «Агрегатные функции»).

Осуществите вызов каждой из них

CREATE OR REPLACE FUNCTION get\_kopanchuk\_stats()

RETURNS TABLE (min\_kopanchuk DECIMAL, max\_kopanchuk DECIMAL, avg\_kopanchuk DECIMAL, sum\_kopanchuk DECIMAL) AS $$

BEGIN

RETURN QUERY

SELECT

MIN(kopanchuk) AS min\_kopanchuk,

MAX(kopanchuk) AS max\_kopanchuk,

ROUND(AVG(kopanchuk), 2) AS avg\_kopanchuk,

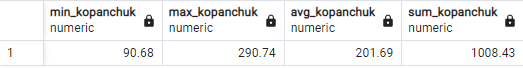
SUM(kopanchuk) AS sum\_kopanchuk

FROM cpu;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM get\_kopanchuk\_stats();



CREATE OR REPLACE FUNCTION count\_kopanchuk\_below\_threshold()

RETURNS INT AS $$

DECLARE

min\_threshold DECIMAL;

count\_rows INT;

BEGIN

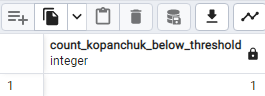
SELECT MIN(kopanchuk) + 50 INTO min\_threshold FROM cpu;

SELECT COUNT(\*) INTO count\_rows FROM cpu WHERE kopanchuk <= min\_threshold;

RETURN count\_rows;

END;

$$ LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION get\_rows\_below\_avg\_div\_100()

RETURNS TABLE (cpuID INT, cpuModel VARCHAR(100), ClockSpeed DECIMAL, Cores INT, kopanchuk DECIMAL) AS $$

BEGIN

RETURN QUERY

SELECT cpu.\*

FROM cpu

WHERE cpu.cpuid <= (SELECT AVG(cpu.cpuid)/100 FROM cpu);

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM get\_rows\_below\_avg\_div\_100();

