Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 9

Программирование на стороне сервера в среде СУБД PostgreSQL
Тема

Преподаватель _____ Е. П. Моргунов
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Ход выполнения

1.1 Вопрос 1

```
edu=# SELECT * FROM (SELECT * FROM generate_students_data()) subquery;
 mark_book |                name                | psp_ser | psp_num
-----------+------------------------------------+---------+---------
     10000 | Логинов Андрей Анатольевич         |    1000 |  100000
     10001 | Логиновых Андрей Анатольевич       |    1001 |  100001
     10002 | Логиновский Андрей Анатольевич     |    1002 |  100002
     10003 | Логиновичев Андрей Анатольевич     |    1003 |  100003
     10004 | Логиненко Андрей Анатольевич       |    1004 |  100004
     10005 | Логинов Андрей Кириллович          |    1005 |  100005
     10006 | Логиновых Андрей Кириллович        |    1006 |  100006
     10007 | Логиновский Андрей Кириллович      |    1007 |  100007
     10008 | Логиновичев Андрей Кириллович      |    1008 |  100008
     10009 | Логиненко Андрей Кириллович        |    1009 |  100009
     10010 | Логинов Андрей Павлович            |    1010 |  100010
     10011 | Логиновых Андрей Павлович          |    1011 |  100011
     10012 | Логиновский Андрей Павлович        |    1012 |  100012
```

Рисунок 1 – Использование функции в подзапросе

1.2 Вопрос 3

IN используется, чтобы передать в функцию параметр, OUT используется, чтобы возвратить параметр из функции.

1.3 Вопрос 5

Чтобы функция возвращала табличное значение, нужно написать RETURNS TABLE перед телом функции.

1.4 Вопрос 7

Триггер – это механизм, заставляющий СУБД выполнить конкретную функцию, когда выполняется определенный тип операций.

1.5 Вопрос 9

Row-level триггеры вызываются для каждой строки, на которую влияет команда, вызвавшая срабатывание триггера.

Statement-level триггеры вызываются лишь один раз после срабатывания.

1.6 Вопрос 11

```
postgres@brain:~$ psql -d ais -f adj_list.sql
psql:adj_list.sql:6: ЗАМЕЧАНИЕ:  таблица "personnel" не существует, пропускается
DROP TABLE
CREATE TABLE
INSERT 0 9
psql:adj_list.sql:29: ЗАМЕЧАНИЕ:  таблица "org_chart" не существует, пропускается
DROP TABLE
CREATE TABLE
INSERT 0 8
CREATE FUNCTION
psql:adj_list.sql:126: ЗАМЕЧАНИЕ:  триггер "check_org_chart" для отношения "org_chart" не существует, пропускается
DROP TRIGGER
CREATE TRIGGER
CREATE FUNCTION
CREATE FUNCTION
psql:adj_list.sql:319: ОШИБКА:  ошибка синтаксиса (примерное положение: "82")
СТРОКА 9: 82
          ^
psql:adj_list.sql:325: ЗАМЕЧАНИЕ:  представление "personnel_org_chart" не существует, пропускается
DROP VIEW
CREATE VIEW
psql:adj_list.sql:341: ЗАМЕЧАНИЕ:  представление "create_paths" не существует, пропускается
DROP VIEW
CREATE VIEW
CREATE FUNCTION
postgres@brain:~$
```

Рисунок 2 – Создание базы данных ais с методами хранения иерархий

1.7 Вопрос 13

```
ais=# SELECT * FROM tree_test();
 tree_test
-----------
 Tree
(1 строка)

ais=#
```

Рисунок 3 – Работы функции без циклов

```
ais=# SELECT * FROM Org_chart;
      job_title       | emp_nbr | boss_emp_nbr |  salary
----------------------+---------+--------------+-----------
 Президент            |       1 |              | 1000.0000
 Вице-президент 1     |       2 |            1 |  900.0000
 Вице-президент 2     |       3 |            1 |  800.0000
 Архитектор           |       4 |            3 |  700.0000
 Программист C        |       6 |            3 |  500.0000
 Программист Perl     |       7 |            5 |  450.0000
 Оператор             |       8 |            5 |  400.0000
 Ведущий программист  |       5 |            8 |  600.0000
(8 строк)

ais=# SELECT * FROM tree_test();
 tree_test
-----------
 Cycles
(1 строка)
```

Рисунок 4 – Создание простого цикла

```
ais=# SELECT * FROM Org_chart;
      job_title       | emp_nbr | boss_emp_nbr |  salary
----------------------+---------+--------------+-----------
 Президент            |       1 |              | 1000.0000
 Вице-президент 1     |       2 |            1 |  900.0000
 Вице-президент 2     |       3 |            1 |  800.0000
 Архитектор           |       4 |            3 |  700.0000
 Программист C        |       6 |            3 |  500.0000
 Ведущий программист  |       5 |            8 |  600.0000
 Программист Perl     |       7 |            5 |  450.0000
 Оператор             |       8 |            7 |  400.0000
(8 строк)

ais=# SELECT * FROM tree_test();
 tree_test
-----------
 Cycles
(1 строка)
```

Рисунок 5 – Создание сложного цикла

1.8 Вопрос 15

```
ais=# SELECT * FROM org_chart;
      job_title      | emp_nbr | boss_emp_nbr |  salary
---------------------+---------+--------------+-----------
 Президент           |       1 |              | 1000.0000
 Вице-президент 1    |       2 |            1 |  900.0000
 Вице-президент 2    |       3 |            1 |  800.0000
 Архитектор          |       4 |            3 |  700.0000
 Ведущий программист |       5 |            8 |  600.0000
 Программист Perl    |       7 |            5 |  450.0000
 Оператор            |       8 |            7 |  400.0000
(7 строк)

ais=# SELECT * FROM delete_subtree( 8 );
 delete_subtree
----------------

(1 строка)

ais=# SELECT * FROM org_chart;
    job_title     | emp_nbr | boss_emp_nbr |  salary
------------------+---------+--------------+-----------
 Президент        |       1 |              | 1000.0000
 Вице-президент 1 |       2 |            1 |  900.0000
 Вице-президент 2 |       3 |            1 |  800.0000
 Архитектор       |       4 |            3 |  700.0000
(4 строки)

ais=# SELECT * FROM personnel_org_chart;
 emp_nbr |  emp   | boss_emp_nbr | boss
---------+--------+--------------+-------
       1 | Иван   |              |
       2 | Петр   |            1 | Иван
       3 | Антон  |            1 | Иван
       4 | Захар  |            3 | Антон
(4 строки)
```

Рисунок 6 – Удаление члена в иерархическом дереве

1.9 Вопрос 17

```
ais=# DROP VIEW IF EXISTS Create_paths;
ЗАМЕЧАНИЕ:  представление "create_paths" не существует, пропускается
DROP VIEW
ais=# CREATE VIEW Create_paths ( level1, level2, level3, level4, level5 ) AS
ais-#  SELECT O1.emp AS e1, O2.emp AS e2, O3.emp AS e3,
ais-#  O4.emp AS e4, O5.emp AS e5
ais-#  FROM Personnel_org_chart AS O1
ais-#  LEFT OUTER JOIN Personnel_org_chart AS O2
ais-#  ON O1.emp = O2.boss
ais-#  LEFT OUTER JOIN Personnel_org_chart AS O3
ais-#  ON O2.emp = O3.boss
ais-#  LEFT OUTER JOIN Personnel_org_chart AS O4
ais-#  ON O3.emp = O4.boss
ais-#  LEFT OUTER JOIN Personnel_org_chart AS O5
ais-#  ON O4.emp = O5.boss
ais-#  -- Если закомментировать условие WHERE, тогда будут
ais-#  -- построены цепочки, начинающиеся с каждого работника,
ais-#  -- а не только с главного руководителя.
ais-#  WHERE O1.emp = 'Иван';
CREATE VIEW
ais=# SELECT * FROM Create_paths;
 level1 | level2 | level3 | level4 | level5
--------+--------+--------+--------+--------
 Иван   | Антон  | Захар  |        |
 Иван   | Петр   |        |        |
(2 строки)
```

Рисунок 7 – Добавление нового уровня иерархии

1.10 Вопрос 19

```
edu=# CREATE TABLE log_students ( name text,
edited timestamp);
CREATE TABLE
```

Рисунок 8 – Добавление таблицы с логом

```
edu=# CREATE TABLE log_students ( name text,
edited timestamp);
CREATE TABLE
edu=# CREATE RULE log_students_rule AS ON INSERT TO students
edu-# DO INSERT INTO log_students VALUES (
edu(# NEW.name,
edu(# current_timestamp
edu(# );
CREATE RULE
edu=# CREATE RULE log_students_rule_update AS ON UPDATE TO students
edu-# DO INSERT INTO log_students VALUES (
edu(# NEW.name,
edu(# current_timestamp
edu(# );
CREATE RULE
edu=#
```

Рисунок 9 – Создание правил

```
edu=# UPDATE students SET psp_ser = 7777
edu-# WHERE name = 'Петров Петр Петрович';
UPDATE 1
edu=# SELECT * FROM students;
 record_book |            name             | psp_ser | psp_num |           edited
-------------+-----------------------------+---------+---------+---------------------------
       11111 | Иванов Иван Иванович        |    1111 |  111111 | 2021-05-19 15:02:12.893326
       22222 | Павлов Павел Павлович       |    2222 |  222222 | 2021-05-19 15:02:12.893326
       33333 | Андреев Андрей Андреевич    |    3333 |  333333 | 2021-05-19 15:02:12.893326
       44444 | Николаев Николай Николаевич |    4444 |  444444 | 2021-05-19 15:02:12.893326
       55555 | Федоров Федор Федорович     |    5555 |  555555 | 2021-05-19 15:02:12.893326
       66666 | Петров Петр Петрович        |    7777 |         | 2021-05-19 15:02:12.893326
(6 строк)

edu=# SELECT * FROM log_students;
         name         |           edited
----------------------+---------------------------
 Петров Петр Петрович | 2021-05-19 15:13:44.699835
(1 строка)
```

Рисунок 10 – Пример срабатывания правила