

# Язык SQL, домашнее задание 10

Бурмашев Григорий, БПМИ-208

22 декабря 2022 г.

## Номер 12

12. Сделайте выборки данных из таблиц «Персонал» и «Организационная структура», а также реконструируйте организационную структуру с помощью двух представлений (view). Команды можно выполнять не только в среде интерактивного терминала psql, но также и из командной строки операционной системы. Выполните эти команды в командной строке операционной системы:

```
psql -d ais -c "SELECT * FROM Personnel"  
psql -d ais -c "SELECT * FROM Org_chart"  
psql -d ais -c "SELECT * FROM Personnel_org_chart"  
psql -d ais -c "SELECT * FROM Create_paths"
```

```

~ > psql -d ais -c 'SELECT * FROM personnel'
 emp_nbr | emp_name | address | birth_date
-----+-----+-----+-----
      0 | вакансия |         | 2014-05-19
      1 | Иван     | ул. Любителей языка С | 1962-12-01
      2 | Петр     | ул. UNIX гуру         | 1965-10-21
      3 | Антон    | ул. Ассемблерная     | 1964-04-17
      4 | Захар    | ул. им. СУБД PostgreSQL | 1963-09-27
      5 | Ирина    | просп. Программистов   | 1968-05-12
      6 | Анна     | пер. Перловый          | 1969-03-20
      7 | Андрей   | пл. Баз данных        | 1945-11-07
      8 | Николай  | наб. ОС Linux          | 1944-12-01
(9 rows)

```

```

~ > psql -d ais -c '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 |              3 | 600.0000
Программист С |      6 |              3 | 500.0000
Программист Perl |      7 |              5 | 450.0000
Оператор     |      8 |              5 | 400.0000
(8 rows)

```

```
~ > psql -d ais -c '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
5	Ирина	3	Антон
6	Анна	3	Антон
7	Андрей	5	Ирина
8	Николай	5	Ирина

(8 rows)

```
~ > psql -d ais -c 'SELECT * FROM create_paths'
```

level1	level2	level3	level4
Иван	Антон	Ирина	Андрей
Иван	Антон	Ирина	Николай
Иван	Петр		
Иван	Антон	Захар	
Иван	Антон	Анна	

(5 rows)

## Номер 13

В базе данных не создана учетная запись такого пользователя.

13. Выполните проверку структуры дерева на предмет отсутствия циклов с помощью функции tree\_test().

```
SELECT * FROM tree_test();
```

Если вы еще не вносили изменения в таблицу «Организационная структура», то функция покажет отсутствие нарушения структуры дерева. Теперь создайте в таблице «Организационная структура» сначала короткий цикл, а затем длинный цикл. Для каждого из указанных циклов выполните проверку с помощью функции tree\_test().

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

Первый раз:

```
ais=# UPDATE org_chart SET boss_emp_nbr = 4 WHERE emp_nbr = 3;
UPDATE 1
ais=# SELECT * FROM tree_test();
tree_test
-----
Cycles
(1 row)
```

Второй раз:

```
ais=# UPDATE org_chart SET boss_emp_nbr = 8 WHERE emp_nbr = 3;
UPDATE 1
ais=# SELECT * FROM tree_test();
tree_test
-----
Cycles
(1 row)
```

## Номер 14

14. Выполните обход дерева организационной структуры снизу вверх, начиная с конкретного узла, можно с помощью функции `up_tree_traversal()` либо функции `up_tree_traversal2()`. Сначала сде-

```
ais=# SELECT * FROM up_tree_traversal( 6 );
 emp_nbr | boss_emp_nbr
```

```
-----+-----
      6 |          3
      3 |          1
      1 |
(3 rows)
```

```
ais=# SELECT * FROM up_tree_traversal( 3 );
 emp_nbr | boss_emp_nbr
```

```
-----+-----
      3 |          1
      1 |
(2 rows)
```

```
ais=# SELECT * FROM up_tree_traversal2( 6 ) AS (emp int, boss int);
 emp | boss
```

```
-----+-----
      6 |      3
      3 |      1
      1 |
(3 rows)
```

```
ais=# SELECT * FROM up_tree_traversal( ( SELECT emp_nbr FROM personnel
WHERE emp_name = 'Ирина' ) );
 emp_nbr | boss_emp_nbr
```

```
-----+-----
      5 |          3
      3 |          1
      1 |
(3 rows)
```

```
ais=# SELECT * FROM up_tree_traversal( ( SELECT emp_nbr FROM personnel
WHERE emp_name = 'Андрей' ) );
 emp_nbr | boss_emp_nbr
```

```
-----+-----
      7 |          5
      5 |          3
      3 |          1
      1 |
(4 rows)
```

## Номер 15

15. Выполните операцию удаления поддерева с помощью функции `delete_subtree()`. Параметром функции является код работника.

```
SELECT * FROM delete_subtree( 6 );
```

Аналогично работе с функцией `up_tree_traversal()` используйте подзапрос для получения кода работника по его имени. После удаления поддерева посмотрите, что стало с организационной структурой, с помощью двух представлений `Personnel_org_chart` и `Create_paths`.

Изначальное состояние:

1 **SELECT \* FROM create\_paths**

	level1 character varying (10)	level2 character varying (10)	level3 character varying (10)	level4 character varying (10)
1	Иван	Антон	Ирина	Андрей
2	Иван	Антон	Ирина	Николай
3	Иван	Петр	[null]	[null]
4	Иван	Антон	Захар	[null]
5	Иван	Антон	Анна	[null]

1 **SELECT \* FROM personnel\_org\_chart;**

	emp_nbr integer	emp character varying (10)	boss_emp_nbr integer	boss character varying (10)
1	1	Иван	[null]	[null]
2	2	Петр	1	Иван
3	3	Антон	1	Иван
4	4	Захар	3	Антон
5	5	Ирина	3	Антон
6	6	Анна	3	Антон
7	7	Андрей	5	Ирина
8	8	Николай	5	Ирина

Проводим запрос:



```

1 SELECT * FROM delete_subtree( ( SELECT emp_nbr FROM personnel
2 WHERE emp_name = 'Андрей' ) );

```

Data Output Messages Notifications

delete\_subtree  
void

1	
---	--

Новое состояние:

```

1 SELECT * FROM create_paths;

```

Data Output Messages Notifications

	level1 character varying (10)	level2 character varying (10)	level3 character varying (10)	level4 character varying (10)
1	Иван	Антон	Ирина	Николай
2	Иван	Петр	[null]	[null]
3	Иван	Антон	Захар	[null]
4	Иван	Антон	Анна	[null]

```

1 SELECT * FROM personnel_org_chart;

```

Data Output Messages Notifications

	emp_nbr integer	emp character varying (10)	boss_emp_nbr integer	boss character varying (10)
1	1	Иван	[null]	[null]
2	2	Петр	1	Иван
3	3	Антон	1	Иван
4	4	Захар	3	Антон
5	5	Ирина	3	Антон
6	6	Анна	3	Антон
7	8	Николай	5	Ирина

## Номер 16

16. Если в таблице «Организационная структура» осталось мало данных, то дополните ее данными и выполните удаление элемента иерархии и продвижение дочерних элементов на один уровень вверх (т. е. к «бабушке»).

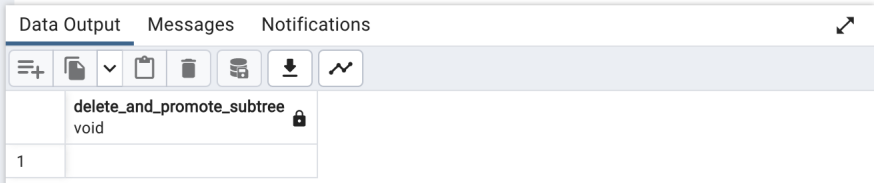
```
SELECT * FROM delete_and_promote_subtree( 5 );
```

Аналогично работе с функцией `up_tree_traversal()` используйте подзапрос для получения кода работника по его имени.

После удаления элемента иерархии посмотрите, что стало с организационной структурой, с помощью двух представлений `Personnel_org_chart` и `Create_paths`.

Провожу запрос:

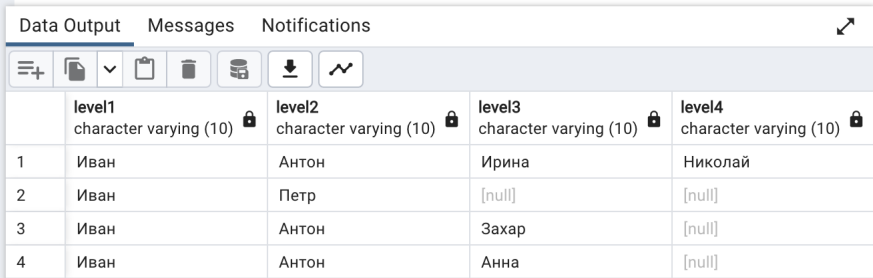
```
1 SELECT * FROM delete_and_promote_subtree(( SELECT emp_nbr FROM personnel WHERE emp_name = 'Андрей' ));
```



	delete_and_promote_subtree
1	void

Смотрю состояния:

```
1 SELECT * FROM create_paths;
```



	level1 character varying (10)	level2 character varying (10)	level3 character varying (10)	level4 character varying (10)
1	Иван	Антон	Ирина	Николай
2	Иван	Петр	[null]	[null]
3	Иван	Антон	Захар	[null]
4	Иван	Антон	Анна	[null]

```
1 SELECT * FROM personnel_org_chart;
```

Data Output Messages Notifications



	emp_nbr integer	emp character varying (10)	boss_emp_nbr integer	boss character varying (10)
1	1	Иван	[null]	[null]
2	2	Петр	1	Иван
3	3	Антон	1	Иван
4	4	Захар	3	Антон
5	5	Ирина	3	Антон
6	6	Анна	3	Антон
7	8	Николай	5	Ирина

## Номер 17

17. Представление Create\_paths позволяет отобразить только четыре уровня иерархии. Модифицируйте его так, чтобы оно могло работать с пятью уровнями иерархии.

Создаю представление

```
1 CREATE VIEW create_paths5 ( level1, level2, level3, level4, level5 ) AS
2 SELECT 01.emp AS e1, 02.emp AS e2, 03.emp AS e3,
3         04.emp AS e4, 05.emp AS e5
4 FROM Personnel_org_chart AS 01
5 LEFT OUTER JOIN Personnel_org_chart AS 02
6     ON 01.emp = 02.boss
7 LEFT OUTER JOIN Personnel_org_chart AS 03
8     ON 02.emp = 03.boss
9 LEFT OUTER JOIN Personnel_org_chart AS 04
10    ON 03.emp = 04.boss
11 LEFT OUTER JOIN Personnel_org_chart AS 05
12    ON 04.emp = 05.boss
13 -- Если закомментировать условие WHERE, тогда будут
14 -- построены цепочки, начинающиеся с каждого работника,
15 -- а не только с главного руководителя.
16 WHERE 01.emp = 'Иван';
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 72 msec.

Смотрю вывод:

```
1 SELECT * FROM create_paths5;
```

Data Output Messages Notifications



	level1 character varying (10) 🔒	level2 character varying (10) 🔒	level3 character varying (10) 🔒	level4 character varying (10) 🔒	level5 character varying (10) 🔒
1	Иван	Антон	Анна	[null]	[null]
2	Иван	Антон	Захар	[null]	[null]
3	Иван	Петр	[null]	[null]	[null]
4	Иван	Антон	Ирина	Николай	[null]

## Номер 18

18. Самостоятельно ознакомьтесь с таким средством работы с таблицами базы данных, как курсоры (cursors). Воспользуйтесь технической документацией на PostgreSQL, глава «PL/pgSQL – SQL Procedural Language». Напишите небольшую функцию с применением курсора.

В качестве примера решил создать функцию с курсором, которая показывает, родился ли человек в четный день месяца. Исходно создал новую колонку в таблице и заполнил ее False. Сама функция:

```
1 create or replace function is_even()
2     returns void as $$
3 declare
4     curs cursor
5     for SELECT *
6     FROM personnel
7     WHERE extract(day from birth_date) % 2 = 0;
8 begin
9     open curs;
10    move curs;
11    WHILE found loop
12        UPDATE personnel SET
13            even_day = 'True'
14        WHERE
15            current OF curs;
16        move curs;
17    END loop;
18    close curs;
19 end; $$
20
21 language plpgsql;
22
```

Исходное состояние до вызова функции:

```
1 SELECT * FROM personnel;
```

	emp_nbr [PK] integer	emp_name character varying (10)	address character varying (35)	birth_date date	even_day text
1	0	вакансия		2014-05-19	False
2	1	Иван	ул. Любителей языка С	1962-12-01	False
3	2	Петр	ул. UNIX гуру	1965-10-21	False
4	3	Антон	ул. Ассемблерная	1964-04-17	False
5	4	Захар	ул. им. СУБД PostgreSQL	1963-09-27	False
6	7	Андрей	пл. Баз данных	1945-11-07	False
7	8	Николай	наб. ОС Linux	1944-12-01	False
8	5	Ирина	просп. Программистов	1968-05-12	False
9	6	Анна	пер. Перловый	1969-03-20	False

Вызываю функцию:

```
1 SELECT * FROM is_even();
```

	is_even void
1	

Смотрю на результат:

```
1 SELECT * FROM personnel;
```

	emp_nbr [PK] integer	emp_name character varying (10)	address character varying (35)	birth_date date	even_day text
1	0	вакансия		2014-05-19	False
2	1	Иван	ул. Любителей языка С	1962-12-01	False
3	2	Петр	ул. UNIX гуру	1965-10-21	False
4	3	Антон	ул. Ассемблерная	1964-04-17	False
5	4	Захар	ул. им. СУБД PostgreSQL	1963-09-27	False
6	7	Андрей	пл. Баз данных	1945-11-07	False
7	8	Николай	наб. ОС Linux	1944-12-01	False
8	5	Ирина	просп. Программистов	1968-05-12	True
9	6	Анна	пер. Перловый	1969-03-20	True

у людей с четными днями рождения теперь графа равна True