

Лабораторна робота 10

«Ієрархічні запити»

Для завдань/розділів відмічених « » – скріншот – обов'язково. Для інших – за домовленістю із викладачем.

Завдання лабораторної роботи виконую всі

Завдання роботи

1. Напишіть запит, який повертає би з таблиці emp інформацію:
 - про ім'я співробітника;
 - про рівень підпорядкованості (найвищий рівень – головний начальник, який нікому не підпорядковується – рівень 1);
 - про шлях підзвітності у форматі: /керівник 1/керівник 2/рядовий співробітник.

Звіт має бути відсортований за рівнем підпорядкованості.

```
WITH RECURSIVE cte_query AS (  
    SELECT e.*, e.ename::varchar as "path", 1 as "level"  
    FROM emp e  
    WHERE e.mgr is null  
    UNION  
    SELECT e.*, "path" || '/' || e.ename, level+1  
    FROM cte_query  
    INNER JOIN emp e ON cte_query.empno = e.mgr  
)  
SELECT RPAD(' ', "level"*3, ' ') || ename "name", "level", "path"  
FROM cte_query  
ORDER BY "path" ASC;
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.*, e.ename::varchar as "path", 1 as "level"  
3     FROM emp e  
4     WHERE e.mgr is null  
5     UNION  
6     SELECT e.*, "path" || '/' || e.ename, level+1  
7     FROM cte_query  
8     INNER JOIN emp e ON cte_query.empno = e.mgr  
9 )  
10 SELECT RPAD(' ', "level"*3, ' ') || ename "name", "level", "path"  
11 FROM cte_query  
12 ORDER BY "path" ASC;
```

name	level	path
KING	1	KING
Aristotle Kristatos	2	KING/Aristotle Kristatos
Auric Goldfinger	2	KING/Auric Goldfinger
BLAKE	2	KING/BLAKE
ALLEN	3	KING/BLAKE/ALLEN
JAMES	3	KING/BLAKE/JAMES
MARTIN	3	KING/BLAKE/MARTIN

2. Виведіть усі імена та зарплати начальників співробітника на ім'я «JET LI»

```
WITH RECURSIVE cte_query AS (  
    SELECT e.* FROM emp e  
    WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET  
LI')  
    UNION  
    SELECT e.*  
    FROM cte_query  
    INNER JOIN emp e ON cte_query.mgr = e.empno  
)  
SELECT ename, sal FROM cte_query;
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.* FROM emp e  
3     WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET LI')  
4     UNION  
5     SELECT e.*  
6     FROM cte_query  
7     INNER JOIN emp e ON cte_query.mgr = e.empno  
8 )  
9 SELECT ename, sal FROM cte_query;  
10  
11  
12  
13  
14
```

ename	sal
JACKIE CHAN	2250
KING	5000

3. Хто з начальників співробітника «JET LI» отримує найвищу зарплату?

```
WITH RECURSIVE cte_query AS (  
    SELECT e.* FROM emp e  
    WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET  
LI')  
    UNION  
    SELECT e.*  
    FROM cte_query  
    INNER JOIN emp e ON cte_query.mgr = e.empno  
)  
SELECT ename, sal FROM cte_query  
ORDER BY sal DESC  
LIMIT 1;
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.* FROM emp e  
3     WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET LI')  
4     UNION  
5     SELECT e.*  
6     FROM cte_query  
7     INNER JOIN emp e ON cte_query.mgr = e.empno  
8 )  
9 SELECT ename, sal FROM cte_query  
10 ORDER BY sal DESC  
11 LIMIT 1;  
12  
13  
14
```

ename	sal
-------	-----

KING	5000
------	------

4. *Виведіть клерків, які є підлеглими (чи підлеглими підлеглих, ...) Блейка.

```
WITH RECURSIVE cte_query AS (  
    SELECT e.* FROM emp e WHERE ename = 'BLAKE'  
    UNION  
    SELECT e.*  
    FROM cte_query  
    INNER JOIN emp e ON cte_query.empno = e.mgr AND e.job = 'CLERK'  
)  
SELECT ename, sal FROM cte_query  
WHERE ename ≠ 'BLAKE';
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.* FROM emp e WHERE ename = 'BLAKE'  
3     UNION  
4     SELECT e.*  
5     FROM cte_query  
6     INNER JOIN emp e ON cte_query.empno = e.mgr AND e.job = 'CLERK'  
7 )  
8 SELECT ename, sal FROM cte_query  
9 WHERE ename != 'BLAKE';  
10  
11  
12  
13  
14
```

ename ▲	sal ▲
---------	-------

JAMES	950
-------	-----

5. ** Виведіть клерків, які НЕ є підлеглими (чи підлеглими підлеглих, ...) Блейка.

```
WITH RECURSIVE cte_query AS (  
    SELECT e.* FROM emp e WHERE mgr is null  
    UNION  
    SELECT e.*  
    FROM cte_query  
    INNER JOIN emp e ON (cte_query.empno = e.mgr AND e.ename ≠  
'BLAKE')  
)  
SELECT * FROM cte_query  
WHERE job = 'CLERK';
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.* FROM emp e WHERE mgr is null  
3     UNION  
4     SELECT e.*  
5     FROM cte_query  
6     INNER JOIN emp e ON (cte_query.empno = e.mgr AND e.ename != 'BLAKE')  
7 )  
8 SELECT * FROM cte_query  
9 WHERE job = 'CLERK';
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7934	MILLER	CLERK	7782	2012-01-23	1200	(NULL)	10
7876	ADAMS	CLERK	7788	2013-01-12	400	(NULL)	20
7369	SMITH	CLERK	7902	2010-12-17	800	(NULL)	20

6. ** У кого зі співробітників 2-ї ланки (підлеглих Кінга) у підпорядкуванні найбільше клерків?

```
WITH RECURSIVE cte_query AS (  
    SELECT e.*, NULL::varchar as "manager", 1 as "level"  
    FROM emp e  
    WHERE e.mgr is null  
    UNION  
    SELECT e.*, (  
        CASE WHEN level = 2 THEN cte_query.ename  
        ELSE cte_query.manager  
        END  
    ) as "manager", level+1  
    FROM cte_query  
    INNER JOIN emp e ON cte_query.empno = e.mgr  
)  
SELECT manager, COUNT(*) FROM cte_query  
WHERE NOT manager is NULL AND job = 'CLERK'  
GROUP by manager  
ORDER by count DESC  
LIMIT 1;
```

```
1 WITH RECURSIVE cte_query AS (  
2     SELECT e.*, NULL::varchar as "manager", 1 as "level"  
3     FROM emp e  
4     WHERE e.mgr is null  
5     UNION  
6     SELECT e.*, (  
7         CASE WHEN level = 2 THEN cte_query.ename  
8         ELSE cte_query.manager  
9         END  
10    ) as "manager", level+1  
11    FROM cte_query  
12    INNER JOIN emp e ON cte_query.empno = e.mgr  
13 )  
14 SELECT manager, COUNT(*) FROM cte_query  
15 WHERE NOT manager is NULL AND job = 'CLERK'  
16 GROUP by manager  
17 ORDER by count DESC  
18 LIMIT 1;
```

manager ▲	count ▲
-----------	---------

JONES	2
-------	---

Пропуски

7. Завдання

- Створіть таблицю T заповнену числами -3..10

```
CREATE TABLE "T" AS  
WITH RECURSIVE t_table AS (  
  SELECT -3 as "num"  
  UNION  
  SELECT num+1 FROM t_table WHERE num < 10  
)  
SELECT * FROM t_table;
```

```
1 CREATE TABLE "T" AS  
2 WITH RECURSIVE t_table AS (  
3   SELECT -3 as "num"  
4   UNION  
5   SELECT num+1 FROM t_table WHERE num < 10  
6 )  
7 SELECT * FROM t_table;  
8  
9 SELECT * FROM "T";
```

num ▲

-3

-2

-1

0

1

2

3

4

5

6

- Видаліть з таблиці числа 4 і 6. Уявімо, що ми не знаємо які саме числа були видалені, але точно НЕ перше чи останнє.



num
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10

- Напишіть запит, який виведе видалені числа.

```
WITH RECURSIVE t_table AS (  
  SELECT -3 as "num"  
  UNION  
  SELECT num+1 FROM t_table WHERE num < 10  
)  
SELECT t_table.* FROM t_table  
LEFT JOIN "T" USING (num)  
WHERE "T".num is NULL;
```

```
1 WITH RECURSIVE t_table AS (  
2   SELECT -3 as "num"  
3   UNION  
4   SELECT num+1 FROM t_table WHERE num < 10  
5 )  
6 SELECT t_table.* FROM t_table  
7 LEFT JOIN "T" USING (num)  
8 WHERE "T".num is NULL;
```

Save

Run

num

4

6

- * Як буде виглядати запит, якщо максимальний елемент наперед не відомий

```
WITH RECURSIVE t_table AS (  
  SELECT -3 as "num"  
  UNION  
  SELECT num+1 FROM t_table WHERE num < (SELECT MAX(num) FROM "T")  
)  
SELECT t_table.* FROM t_table  
LEFT JOIN "T" USING (num)  
WHERE "T".num is NULL;
```

```
1 WITH RECURSIVE t_table AS (  
2   SELECT -3 as "num"  
3   UNION  
4   SELECT num+1 FROM t_table WHERE num < (SELECT MAX(num) FROM "T")  
5 )  
6 SELECT t_table.* FROM t_table  
7 LEFT JOIN "T" USING (num)  
8 WHERE "T".num is NULL;
```

num ▲

4

6

- * Як буде виглядати запит, якщо максимальний і мінімальний елемент наперед невідомі?

```
WITH RECURSIVE t_table AS (  
  SELECT (SELECT MIN(num) FROM "T") as "num"  
  UNION  
  SELECT num+1 FROM t_table WHERE num < (SELECT MAX(num) FROM "T")  
)  
SELECT t_table.* FROM t_table  
LEFT JOIN "T" USING (num)  
WHERE "T".num is NULL;
```

```
1 WITH RECURSIVE t_table AS (  
2   SELECT (SELECT MIN(num) FROM "T") as "num"  
3   UNION  
4   SELECT num+1 FROM t_table WHERE num < (SELECT MAX(num) FROM "T")  
5 )  
6 SELECT t_table.* FROM t_table  
7 LEFT JOIN "T" USING (num)  
8 WHERE "T".num is NULL;
```

num ▲

4

6

- Видаліть таблицю T.

```
DROP TABLE "T";
```

```
1 DROP TABLE "T";
```

Save


Run

Query 1/1: No Results. 0 rows affected. See the select box in the bottom left ↙ for more query results.

Втекти звідси

Створіть таблицю відстаней між містами (не менше 8 рядків),

8. Виведіть данні з Вашої таблиці.

 id int4 ▲	A text ▲	B text ▲	d int4 ▲
1	Dnipro	Odesa	468
2	Kyiv	Dnipro	533
3	Vinnytsia	Kyiv	256
4	Vinnytsia	Zshytomyr	125
5	Dnipro	Donetsk	252
6	Vinnytsia	Dnipro	645
7	Kyiv	Sumy	346
8	Sumy	Donetsk	706

9. Куди можна доїхати за 2 пересадки з обраного вами міста?

```
WITH RECURSIVE t_path AS (  
    SELECT  
        "A" as "path",  
        0 as "level",  
        "A" as "prev",  
        0 as "dist"  
    FROM path WHERE "A" = 'Vinnytsia'  
    UNION  
    SELECT  
        t_path."path" || ' → ' || path."B",  
        "level"+1,  
        path."B" as "prev",  
        dist + path.d  
    FROM t_path  
    JOIN path ON (t_path.prev = path."A")  
)  
SELECT path, level, dist FROM t_path  
WHERE level = 2;
```

```
1 WITH RECURSIVE t_path AS (  
2     SELECT  
3         "A" as "path",  
4         0 as "level",  
5         "A" as "prev",  
6         0 as "dist"  
7     FROM path WHERE "A" = 'Vinnytsia'  
8     UNION  
9     SELECT  
10        t_path."path" || ' → ' || path."B",  
11        "level"+1,  
12        path."B" as "prev",  
13        dist + path.d  
14    FROM t_path  
15    JOIN path ON (t_path.prev = path."A")  
16 )  
17 SELECT path, level, dist FROM t_path  
18 WHERE level = 2;
```

path	level	dist
Vinnytsia → Kyiv → Dnipro	2	789
Vinnytsia → Dnipro → Donetsk	2	897
Vinnytsia → Kyiv → Sumy	2	602

10. * Знайдіть маршрут з мінімальною кількістю пересадок між двома вибраними вами містами.

```
WITH RECURSIVE t_path AS (  
    SELECT  
        "A" as "path",  
        0 as "level",  
        "A" as "last",  
        0 as "dist"  
    FROM path WHERE "A" = 'Vinnytsia'  
    UNION  
    SELECT  
        t_path."path" || ' → ' || path."B",  
        "level"+1,  
        path."B" as "last",  
        dist + path.d  
    FROM t_path  
    JOIN path ON (t_path.last = path."A")  
)  
SELECT last, min(level) FROM t_path  
WHERE last = 'Kyiv'  
GROUP BY last;
```

```
1 WITH RECURSIVE t_path AS (  
2     SELECT  
3         "A" as "path",  
4         0 as "level",  
5         "A" as "last",  
6         0 as "dist"  
7     FROM path WHERE "A" = 'Vinnytsia'  
8     UNION  
9     SELECT  
10        t_path."path" || ' → ' || path."B",  
11        "level"+1,  
12        path."B" as "last",  
13        dist + path.d  
14    FROM t_path  
15    JOIN path ON (t_path.last = path."A")  
16 )  
17 SELECT last, min(level) FROM t_path  
18 WHERE last = 'Kyiv'  
19 GROUP BY last;
```

last	m...
------	------

Kyiv	1
------	---

11. ** Знайдіть найкоротший маршрут між вибраними вами містами.

```
WITH RECURSIVE t_path AS (  
    SELECT  
        "A" as "path",  
        0 as "level",  
        "A" as "last",  
        0 as "dist"  
    FROM path WHERE "A" = 'Vinnytsia'  
    UNION  
    SELECT  
        t_path."path" || ' → ' || path."B",  
        "level"+1,  
        path."B" as "last",  
        dist + path.d  
    FROM t_path  
    JOIN path ON (t_path.last = path."A")  
)  
SELECT last, min(dist) FROM t_path  
WHERE last = 'Kyiv'  
GROUP BY last;
```

```
1 WITH RECURSIVE t_path AS (  
2     SELECT  
3         "A" as "path",  
4         0 as "level",  
5         "A" as "last",  
6         0 as "dist"  
7     FROM path WHERE "A" = 'Vinnytsia'  
8     UNION  
9     SELECT  
10        t_path."path" || ' → ' || path."B",  
11        "level"+1,  
12        path."B" as "last",  
13        dist + path.d  
14    FROM t_path  
15    JOIN path ON (t_path.last = path."A")  
16 )  
17 SELECT last, min(dist) FROM t_path  
18 WHERE last = 'Kyiv'  
19 GROUP BY last;
```

last	m...
------	------

Kyiv	256
------	-----

Завдання за варіантами

По завданню 2. Для «А» та «В» їх «спільним найближчим керівником» є «Г»

Варіант 1

1. * У співробітниках додайте поле «керівник» - хто з співробітників є керівником для даного. У випадку, якщо клієнт незадоволений швидкістю проведення операції він може звернутись до керівника. Значи операцію виведіть весь перелік керівників – до кого можна звернутись із скаргою.

```
WITH t_transactions AS (  
    SELECT * FROM transactions WHERE transactionid = 6  
)  
SELECT managers.workerid, people.* FROM t_transactions  
JOIN workers worker ON (t_transactions.confirmedby = worker.workerid)  
JOIN workers managers ON (worker.mgr = managers.workerid)  
JOIN people ON (managers.pid = people.id)
```

```
1 WITH t_transactions AS (  
2     SELECT * FROM transactions WHERE transactionid = 6  
3 )  
4 SELECT managers.workerid, people.* FROM t_transactions  
5 JOIN workers worker ON (t_transactions.confirmedby = worker.workerid)  
6 JOIN workers managers ON (worker.mgr = managers.workerid)  
7 JOIN people ON (managers.pid = people.id)
```

workerid ▲	id ▲	firstname ▲	secondname ▲	sanctions ▲
3	5	DMYTRO	BORSHCH	true

2. ** Для двох співробітників визначить, хто є їх спільним найближчим керівником

```
WITH RECURSIVE first_worker AS (  
    SELECT *, 0 as level FROM workers WHERE workers.workerid = 4  
    UNION  
    SELECT workers.*, level + 1 FROM workers  
    JOIN first_worker ON (workers.workerid = first_worker.mgr)  
) ,  
second_worker AS (  
    SELECT *, 0 as level FROM workers WHERE workers.workerid = 5  
    UNION  
    SELECT workers.*, level + 1 FROM workers  
    JOIN second_worker ON (workers.workerid = second_worker.mgr)  
)  
SELECT 'Nearest mgr is ' || people.firstname || ' ' || people.secondname || ' ID=' || workerid as "Result"  
FROM first_worker JOIN second_worker USING (workerid)  
JOIN people ON (first_worker.pid = people.id)  
ORDER BY first_worker.level  
LIMIT 1;
```

```
1 WITH RECURSIVE first_worker AS (  
2     SELECT *, 0 as level FROM workers WHERE workers.workerid = 4  
3     UNION  
4     SELECT workers.*, level + 1 FROM workers  
5     JOIN first_worker ON (workers.workerid = first_worker.mgr)  
6 ),  
7 second_worker AS (  
8     SELECT *, 0 as level FROM workers WHERE workers.workerid = 5  
9     UNION  
10    SELECT workers.*, level + 1 FROM workers  
11    JOIN second_worker ON (workers.workerid = second_worker.mgr)  
12 )  
13 SELECT 'Nearest mgr is ' || people.firstname || ' ' || people.secondname || ' ID=' || workerid as "Result"  
14 FROM first_worker JOIN second_worker USING (workerid)  
15 JOIN people ON (first_worker.pid = people.id)  
16 ORDER BY first_worker.level  
17 LIMIT 1;
```