Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут”
Кафедра АСОІУ

Контрольная робота № 3

з дисципліни

“Бази даних - 2. Програмні додатки з використанням баз даних”

Тема: “Ієрархічні запити”

Прийняв:
Клименко О. М.

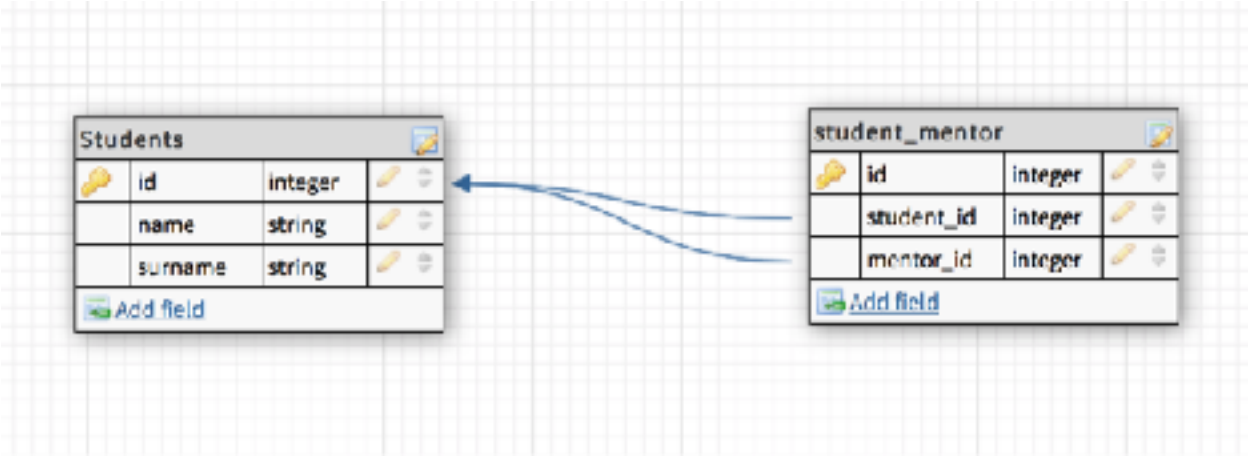
Виконав:
студент 3-го курсу
гр. ІП-52 ФІОТ
Онбиш Олександр
Олегович

Київ 2017

Постановка задачі:

1) Вивести список всіх « наща дків » вказан ого « предка » .
2) Вивести список всіх « пре дків » вказан ого « нащадка » .
3) Вивести список, другий полем якого є « рівень » (аналог псевдо стовпчика level в connect by).
4) (2 запити) Змінити дані в доданій таблиці так, щоб утворився цикл. Написати запит, що видає помилку при зациклюванні. Змінити цей запит так, що б помилки не було.
5) Для всіх « нащадків » (це перше поле : Іванов) вивести список « пре дків » через « / » , де останнім в ланцюгу є цей « нащадок » (це друге поле: Іванен ко /Іванченко/Іванчук /Іванов)

Схема БД:



Запити:

```sql
-- Створення таблиць та вставка даних
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name CHAR(20),
    surname CHAR(20)
    );

CREATE TABLE student_mentor (
    id SERIAL PRIMARY KEY,
    student_id INT REFERENCES students(id),
    mentor_id INT REFERENCES students(id));

INSERT INTO students (name, surname) VALUES
('Daenerys', 'Targaryen');
INSERT INTO students (name, surname) VALUES ('John',
'Snow');
INSERT INTO students (name, surname) VALUES ('Cersei',
'Lannister');
INSERT INTO students (name, surname) VALUES ('Petyr',
'Baelish');
INSERT INTO students (name, surname) VALUES ('Tyrion',
'Lannister');
INSERT INTO students (name, surname) VALUES ('Sandor',
'Clegane');
INSERT INTO students (name, surname) VALUES ('Eddard',
'Stark');
INSERT INTO students (name, surname) VALUES ('Sansa',
'Stark');
INSERT INTO students (name, surname) VALUES ('Joffrey',
'Baratheon');
INSERT INTO students (name, surname) VALUES ('Lord',
'Varys');
INSERT INTO students (name, surname) VALUES ('Grey',
'Worm');
INSERT INTO students (name, surname) VALUES ('Samwell',
'Tarly');

INSERT INTO student_mentor (student_id, mentor_id)
VALUES (1, 2);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (2, 3);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (3, 4);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (3, 11);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (4, 11);
```

```sql
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (4, 12);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (5, 11);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (5, 6);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (6, 7);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (7, 8);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (6, 9);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (6, 10);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (5, 11);
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (5, 11);



-- TASK 1
-- Рекурсивний запит вибирає нащадка ментора
-- та рекурсивно для цього нащадка шукає його нащадка
-- Ці результати об'єднуються за допомогою UNION
WITH RECURSIVE sub(student_id) AS
    (SELECT student_id FROM student_mentor WHERE
mentor_id = (SELECT id FROM students WHERE
name='Samwell')
    UNION ALL
    SELECT student_mentor.student_id FROM sub
    INNER JOIN student_mentor ON
student_mentor.mentor_id = sub.student_id
    )

    SELECT students.name, students.surname FROM sub
    JOIN students ON Students.id = sub.student_id;

-- TASK 2
-- Рекурсивний запит вибирає предка учня
-- та рекурсивно для цього предка шукає його предка
-- Ці результати об'єднуються за допомогою UNION
WITH RECURSIVE sub(mentor_id) AS
    (SELECT mentor_id FROM student_mentor WHERE
student_id = (SELECT id FROM students WHERE
name='Daenerys')
     UNION ALL
     SELECT student_mentor.mentor_id FROM sub
```

```
     INNER JOIN student_mentor ON sub.mentor_id =
student_mentor.student_id
    )

    SELECT students.name, students.surname FROM sub
    JOIN students ON Students.id = sub.mentor_id;


-- TASK 3
-- Рекурсивний запит вибирає предка учня
-- та рекурсивно для цього предка шукає його предка
-- Ці результати об'єднуються за допомогою UNION
-- Поле level рекурсивно передається зі збільшиним
значенням на 1
WITH RECURSIVE sub(mentor_id, level) AS
    (SELECT mentor_id, 1 FROM student_mentor WHERE
student_id = (SELECT id FROM students WHERE
name='Daenerys')
     UNION ALL
     SELECT student_mentor.mentor_id, level + 1 FROM
sub
     INNER JOIN student_mentor ON sub.mentor_id =
student_mentor.student_id
    )

    SELECT students.name, students.surname, level FROM
sub
    JOIN students ON Students.id = sub.mentor_id
    ORDER BY level;

-- TASK 4
-- Додаємо цикл у граф
INSERT INTO student_mentor (student_id, mentor_id)
VALUES (12, 1);
-- Рекурсивний запит вибирає предка учня
-- та рекурсивно для цього предка шукає його предка
-- Ці результати об'єднуються за допомогою UNION
-- Для запобігання нескінченного циклу створюємо масив
-- з ребрами графу по яким вже проходився запит
-- якщо ребро знаходиться у масиві, то припеняємо
рекурсивне занурення
WITH RECURSIVE sub(mentor_id, level, path, cycle) AS
    (SELECT mentor_id, 1, ARRAY[mentor_id], false FROM
student_mentor WHERE student_id = (SELECT id FROM
students WHERE name='Daenerys')
     UNION ALL
     SELECT student_mentor.mentor_id, level + 1, path
|| sub.mentor_id, sub.mentor_id = ANY(path) FROM sub
```

```
     INNER JOIN student_mentor ON sub.mentor_id =
student_mentor.student_id
     WHERE NOT cycle
    )

    SELECT students.name, students.surname, level FROM
sub
    JOIN students ON Students.id = sub.mentor_id
    ORDER BY level;


-- TASK 5
-- Аналогічно TASK 4
-- Для побудови шляху передаємо як параметер попередні
-- ребра по яким занурявся запит
WITH RECURSIVE sub(mentor_id, route, level, path,
cycle) AS
    (SELECT mentor_id, '/' || (SELECT name || ' ' ||
surname FROM students where student_id = students.id)
|| '/' || (SELECT name || ' ' || surname FROM students
where mentor_id = students.id), 1, ARRAY[mentor_id],
false FROM student_mentor WHERE student_id = (SELECT id
FROM students WHERE name='Sandor')
     UNION ALL
     SELECT student_mentor.mentor_id, sub.route || '/'
|| (SELECT name || ' ' || surname FROM students limit
1), level + 1, path || sub.mentor_id, sub.mentor_id =
ANY(path) FROM sub
     INNER JOIN student_mentor ON sub.mentor_id =
student_mentor.student_id
     WHERE NOT cycle
    )

    SELECT students.name || ' ' || students.surname AS
NAME, route FROM sub
    JOIN students ON Students.id = sub.mentor_id
    ORDER BY level;
```

Відеокопія результату роботи програми:

name	surname
Petyr	Baelish
Cersei	Lannister
John	Snow
Daenerys	Targaryen

name	surname
John	Snow
Cersei	Lannister
Petyr	Baelish
Grey	Worm
Grey	Worm
Samwell	Tarly

name	surname	level
John	Snow	1
Cersei	Lannister	2
Petyr	Baelish	3
Grey	Worm	3
Grey	Worm	4
Samwell	Tarly	4

name	surname	level
John	Snow	1
Cersei	Lannister	2

name	route
Eddard Stark	/Sandor Clegane/Eddard Stark
Joffrey Baratheon	/Sandor Clegane/Joffrey Baratheon
Lord Varys	/Sandor Clegane/Lord Varys
Sansa Stark	/Sandor Clegane/Eddard Stark/Daenerys Targaryen