

ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

старший преподаватель

должность, уч. степень, звание

М. В. Величко

инициалы, фамилия

подпись, дата

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №10
Объектно-реляционные базы
данных. Манипуляция данными
и пользовательские операторы

по курсу: «Проектирование баз данных»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №

Z1431

подпись, дата
фамилия

М.Д..Быстров

инициалы,

Санкт-Петербург 2024

ЗАДАНИЕ

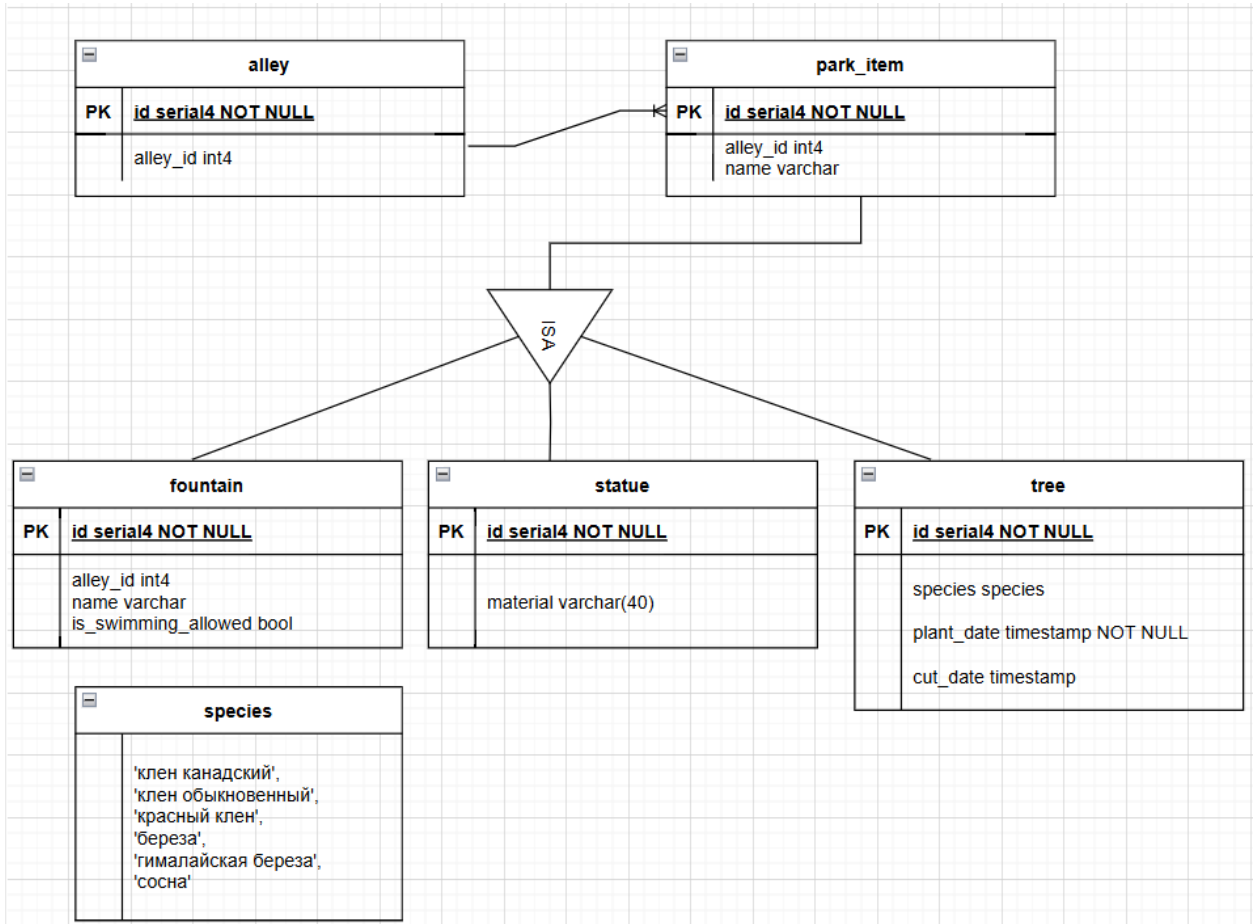
1. Выполнить вставку тестовых данных в таблицы, созданные в ходе выполнения лабораторной работы 9.
2. Сделать запросы выборки с условием к таблицам предку и потомку, только потомку и только предку.
3. Придумать и создать пользовательский оператор для своей предметной области
4. Придумать и создать пользовательскую агрегатную функцию для своей предметной области

Вариант 4:

парк: статуи, фонтаны, деревья ,породы, дата высадки, дата обрезки, расположение, аллеи

- а. аллеи, на которых встречаются разные виды кленов (клен в названии)
- б. аллеи, на которых есть и статуи и фонтаны
- в. дерево, которое было посажено позже всех
- г. порода, деревьев которой больше всего
- д. аллея, на которой нет фонтанов

Физическая модель БД для СУБД Postgresql



Наборы данных

alley Введите SQL выражение чтобы отфильтровать результаты		
Таблица	123 id	123 num
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7

fountain Введите SQL выражение чтобы отфильтровать результаты				
Таблица	123 id	123 alley_id	<input checked="" type="checkbox"/> is_swimming_allowed	A-Z name
1	1	1	[]	Фонтан 1
2	2	2	[v]	Фонтан 2
3	10	6	[]	Фонтан 3

statue Введите SQL выражение чтобы отфильтровать результаты				
Таблица	123 id	123 alley_id	A-Z material	A-Z name
1	3	7	gypsum	Венера Милосская
2	4	6	bronze	Иосиф Сталин (отец народов)
3	5	6	bronze	Радищев

tree Введите SQL выражение чтобы отфильтровать результаты						
Таблица	123 id	123 alley_id	A-Z species	🕒 plant_date	🕒 cut_date	A-Z name
1	6	3	клен канадский	2024-09-28 00:00:00.000	2024-10-28 00:00:00.000	Клен
2	7	4	клен обыкновенный	2024-09-27 00:00:00.000	2024-10-27 00:00:00.000	Клен
3	8	5	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен
4	9	6	береза	2024-09-25 00:00:00.000	2024-10-25 00:00:00.000	Береза
5	11	7	сосна	2024-08-25 00:00:00.000	2024-10-24 00:00:00.000	Сосна
6	12	4	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен

park_item Введите SQL выражение чтобы отфильтровать резуль			
Таблица	123 id	123 alley_id	A-Z name
1	1	1	Фонтан 1
2	2	2	Фонтан 2
3	10	6	Фонтан 3
4	3	7	Венера Милосская
5	4	6	Иосиф Сталин (отец народов)
6	5	6	Радищев
7	6	3	Клен
8	7	4	Клен
9	8	5	Клен
10	9	6	Береза
11	11	7	Сосна
12	12	4	Клен

Текст запросов

```
--а. аллеи, на которых встречаются разные виды кленов (клен в названии)
-- к наследнику
select a.*,
count(t.id) as tree_count, -- кол-во деревьев
unique_by_similar(t."species") as species_count -- кол-во пород
from alley a join tree t on t.alley_id = a.id
where lower(t."name") like '%клен%' -- в имени содержится клен
and exists(
    select t2.id from tree t2
    where t2.alley_id = a.id
    and t2."species" ~ t."species"
    and t2.id != t.id) -- на аллее есть другие клены (подобные деревья)
group by a.id
having unique_by_similar(t."species") = 1; -- в выборку одной группы попали строго
подобные породы

--б. аллеи, на которых есть и статуи и фонтаны
-- к главной таблице и наследнику
SELECT a.* FROM alley a
where (
-- счет уникальных номеров таблиц для элементов на аллее
SELECT count(distinct i.tableoid) FROM park_item i
left JOIN fountain f ON f.id = i.id -- фонтаны
left join statue s on s.id = i.id -- статуи
where i.alley_id = a.id -- на аллее
```

```

        and (f.id is not null or s.id is not null) -- нашлась либо статуя либо фонтан
    ) >= 2; -- как минимум два типа - нашлось и то и другое

--в.   дерево, которое было посажено позже всех
-- к наследнику
select t.* from tree t
where t.plant_date =
    (select max(t2.plant_date) from tree t2);

--г.   порода, деревьев которой больше всего
-- к наследнику
select t."species", count(t.id) as cnt from tree t
group by t."species"
having count(t.id) = (select count(t.id) as cnt from tree t
    group by t."species"
    order by cnt desc limit 1)
order by cnt desc;

--д.   аллея, на которой нет фонтанов
-- к главной таблице
select a.* from alley a -- все аллеи
except
    --исключая аллеи на которых есть фонтаны
    select distinct a2.* from alley a2
    join park_item pi2
    on pi2.alley_id = a2.id
    and pi2.tableoid = 'fountain'::regclass::oid
order by id;

-- запрос к наследнику
select * from tree;

-- запрос к родителю
select * from only park_item;

-- запрос к наследнику и родителю
select pi.*, t.* from park_item pi
join tree t on pi.id = t.id;

-- запрос к родителю, но данные всех наследников
select * from park_item;

```

Код операторов

```

CREATE OR REPLACE FUNCTION public.specia_is_similar(first species, second species)
  RETURNS boolean
  LANGUAGE plpgsql
AS $function$
  BEGIN

    if first = second then
      return true;
    end if;

    if (first in ('клен канадский', 'клен обыкновенный', 'красный клен')
    and second in ('клен канадский', 'клен обыкновенный', 'красный клен'))
    or (first in ('береза', 'гималайская береза')
    and second in ('береза', 'гималайская береза')) then

      return true;
    end if;

    return false;

  END;
$function$
;

```

```

-- создание оператора
create OPERATOR ~ (
  leftarg = species,
  rightarg = species,
  procedure = specia_is_similar
);

```

```

CREATE OR REPLACE FUNCTION public.species_acc(arr species[], next species)
  RETURNS species[]
  LANGUAGE plpgsql
AS $function$
  declare
    ret species[];
  BEGIN

    if (next is not NULL) then
      ret = array_append(arr, next);
    else
      ret = arr;
    end if;

    return ret;

  END;
$function$
;

```

```

CREATE OR REPLACE FUNCTION public.count_unique_similar("values" species[])

```

```

RETURNS integer
LANGUAGE plpgsql
AS $function$
    declare
        distinct_values species[];
        current_specia species;
        inner_current_specia species;

        specs RECORD;
        spec RECORD;

        similar_count int4;
        group_count int4;
        i int4;
    BEGIN

        group_count = 0;

        -- уникальные значения в массиве
        for spec in select distinct * from unnest("values") as val loop

            distinct_values = array_append(
                distinct_values,
                spec.val::species);

        end loop;

        group_count = 0;

        -- удаляем поэлементно с расчетом кол-ва подобных
        while array_length(distinct_values, 1) > 0 loop

            current_specia = distinct_values[1];

            group_count = group_count + 1;

            i = 1;

            while i <= array_length(distinct_values, 1) loop

                inner_current_specia = distinct_values[i];

                -- если вид подобен текущему - удаляем
                if inner_current_specia ~ current_specia then

                    distinct_values = array_remove(distinct_values,
inner_current_specia);

                else

                    i = i + 1;

                end if;

            end loop;

        end loop;
    END
$function$

```



```

        end loop;

    end loop;

    return group_count;

END;
$function$
;

-- создание агрегатной функции
CREATE AGGREGATE unique_by_similar(species) (
    SFUNC = species_acc, -- функция, собирающая массив
    STYPE = species[], -- тип данных состояния
    FINALFUNC = count_unique_similar, -- финализирующая функция
    INITCOND = "{}"); -- начальный пустой массив

```

Пример выполнения операторов

-- тест оператора

```
select 'клен канадский'::species ~ 'клен обыкновенный'::species as is_similar;
```

Результат 1 Результат 2 tree 3 Результат 4 ✕

select 'клен канадский'::species ~ 'клен обыкновенный'::species as is_similar; | Введите SQL выражение чтобы отфильт

	is_similar
1	[v]

```
select 'береза'::species ~ 'клен обыкновенный'::species as is_similar;
```

Результат 1 Результат 2 tree 3 Результат 4 ✕

select 'береза'::species ~ 'клен обыкновенный'::species as is_similar; | Введите SQL выражение что

	is_similar
1	[]

Наборы данных, возвращаемых запросами

```
--а. аллеи, на которых встречаются разные виды кленов (клен в названии)
-- к наследнику
select a.*,
count(t.id) as tree_count, -- кол-во деревьев
unique_by_similar(t."species") as species_count -- кол-во пород
from alley a join tree t on t.alley_id = a.id
where lower(t."name") like '%клен%' -- в имени содержится клен
and exists(
select t2.id from tree t2
where t2.alley_id = a.id
and t2."species" ~ t."species"
and t2.id != t.id) -- на аллее есть другие клены (подобные деревья)
group by a.id
having unique_by_similar(t."species") = 1; -- в выборку одной группы попали строго подобные породы
```

Результат 1 Результат 2 tree 3 alley 4 X

select a.*, count(t.id) as tree_count, unique_by_similar(t."species")

123 id	123 num	123 tree_count	123 species_count
4	4	2	1

```
--б. аллеи, на которых есть и статуи и фонтаны
-- к главной таблице и наследнику
SELECT a.* FROM alley a
where (
-- счет уникальных номеров таблиц для элементов на аллее
SELECT count(distinct i.tableoid) FROM park_item i
left JOIN fountain f ON f.id = i.id -- фонтаны
left join statue s on s.id = i.id -- статуи
where i.alley_id = a.id -- на аллее
and (f.id is not null or s.id is not null) -- нашлась либо статуя либо фонтан
) >= 2; -- как минимум два типа - нашлось и то и другое
```

Результат 1 Результат 2 tree 3 alley 4 X

SELECT a.* FROM alley a where (SELECT count(distinct i.tableoid) F

123 id	123 num
6	6

```
--в.    дерево, которое было посажено позже всех
-- к наследнику
select t.* from tree t
where t.plant_date =
    (select max(t2.plant_date) from tree t2);
```

Результат 1 Результат 2 tree 3 tree 4 ✕

select t.* from tree t where t.plant_date = (select max(t2.plant_date) from tree t2) | Введите SQL выражение чтобы отфильтровать резу

123 id	123 alley_id	A-Z species	plant_date	cut_date	A-Z name
6	3	клен канадский	2024-09-28 00:00:00.000	2024-10-28 00:00:00.000	Клен

```
--г.    порода, деревьев которой больше всего
-- к наследнику
select t."species", count(t.id) as cnt from tree t
group by t."species"
order by cnt desc limit 1;
```

Результат 1 Результат 2 tree 3 tree 4 ✕

select t."species", count(t.id) as cnt from tree t group by t."species" | Введ

A-Z species	123 cnt
красный клен	2

```

--д.    аллея, на которой нет фонтанов
-- к главной таблице
select a.* from alley a -- все аллеи
except
--исключая аллеи на которых есть фонтаны
select distinct a2.* from alley a2
join park_item pi2
on pi2.alley_id = a2.id
and pi2.tableoid = 'fountain'::regclass::oid
order by id;

```

результат 1 Результат 2 tree 3 Результат 4 ×

select a.* from alley a except select distinct a2.* from alley a2 join p | Все

	123 id	123 num
1	3	3
2	4	4
3	5	5
4	7	7

```

-- запрос к наследнику
select * from tree;

```

е 1 ×

select * from tree | Введите SQL выражение чтобы отфильтровать результаты

	123 id	123 alley_id	A-Z species	plant_date	cut_date	A-Z name
	8	3	клен канадский	2024-09-28 00:00:00.000	2024-10-28 00:00:00.000	Клен
	9	4	клен обыкновенный	2024-09-27 00:00:00.000	2024-10-27 00:00:00.000	Клен
	10	5	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен
	11	6	береза	2024-09-25 00:00:00.000	2024-10-25 00:00:00.000	Береза
	12	7	сосна	2024-08-25 00:00:00.000	2024-10-24 00:00:00.000	Сосна
	13	4	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен

-- запрос к родителю

```
select * from only park_item;
```

park_item 1 ×

select * from only park_item | Введите SQL выражение

123 id	123 alley_id	A-Z name
--------	--------------	----------

-- запрос к наследнику и родителю

```
select pi.*, t.* from park_item pi
join tree t on pi.id = t.id;
```

k_item(+) 1 ×

select pi.*, t.* from park_item pi join tree t on pi.id = t.id | Введите SQL выражение чтобы отфильтровать результаты

123 id	123 alley_id	A-Z name	123 id	123 alley_id	A-Z species	plant_date	cut_date	A-Z name
8	3	Клен	8	3	клен канадский	2024-09-28 00:00:00.000	2024-10-28 00:00:00.000	Клен
9	4	Клен	9	4	клен обыкновенный	2024-09-27 00:00:00.000	2024-10-27 00:00:00.000	Клен
10	5	Клен	10	5	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен
11	6	Береза	11	6	береза	2024-09-25 00:00:00.000	2024-10-25 00:00:00.000	Береза
12	7	Сосна	12	7	сосна	2024-08-25 00:00:00.000	2024-10-24 00:00:00.000	Сосна
13	4	Клен	13	4	красный клен	2024-09-26 00:00:00.000	2024-10-26 00:00:00.000	Клен

```
-- запрос к родителю, но данные всех наследников  
select * from park_item;
```

park_item 1 ×

select * from park_item | Введите SQL выражение чтобы отфильт

	123 id	123 alley_id	A-Z name
	8	3	Клен
	9	4	Клен
	10	5	Клен
	11	6	Береза
	12	7	Сосна
	13	4	Клен
	5	7	Венера Милосская
	6	6	Иосиф Сталин (отец народов)
	7	6	Радищев
0	2	1	Фонтан 1
1	3	2	Фонтан 2

Вывод

В ходе выполнения 10 лабораторной работы были созданы:

1. Пользовательский оператор
2. Агрегатная функция

В СУБД PostgreSQL.

Написаны запросы с использованием созданных оператора и функции.

Среди запросов есть запросы к наследнику, к наследнику и родителю, и только к родителю в иерархии наследования реляционных таблиц.