ГУАП

КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель М. В. Величко

должность, уч. степень, звание подпись, дата инициалы, фамилия

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 М.Д..Быстров

подпись, дата инициалы, фамилия

Санкт-Петербург 2024

# ЗАДАНИЕ

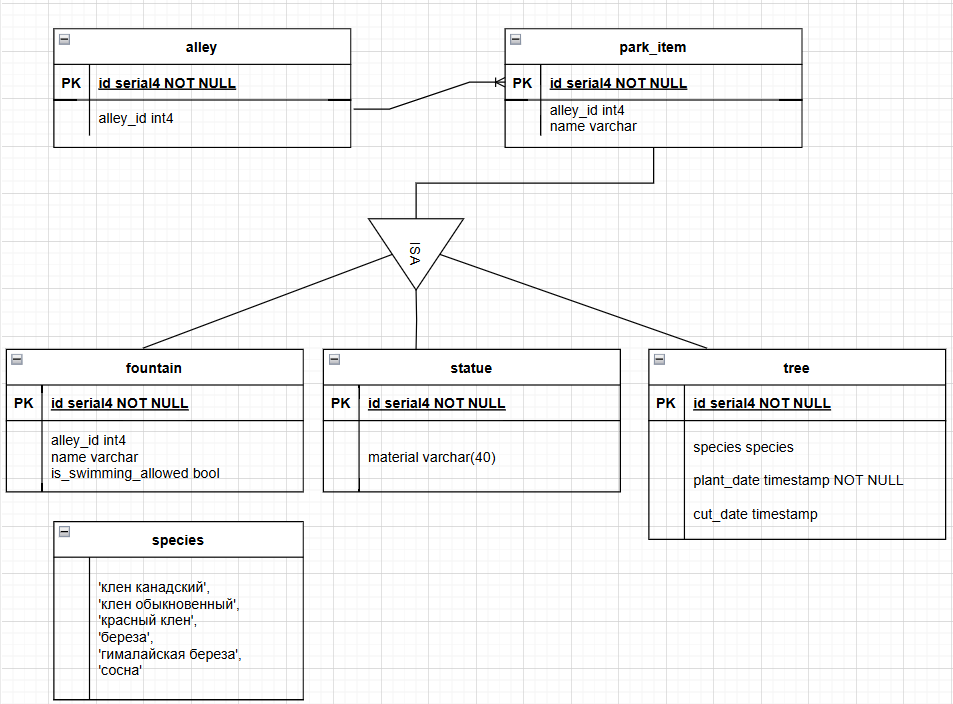
1. Выполнить вставку тестовых данных в таблицы, созданные в ходе выполнения лабораторной работы 9.
2. Сделать запросы выборки с условием к таблицам предку и потомку, только потомку и только предку.
3. Придумать и создать пользовательский оператор для своей предметной области
4. Придумать и создать пользовательскую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

**Вариант** 4:

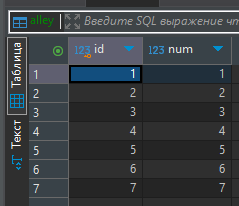
парк: статуи, фонтаны, деревья ,породы, дата высадки, дата обрезки, расположение, аллеи

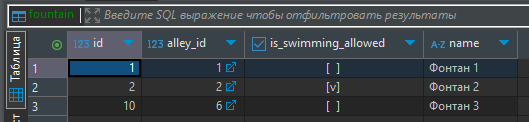
1. аллеи, на которых встречаются разные виды кленов (клен в названии)
2. аллеи, на которых есть и статуи и фонтаны
3. дерево, которое было посажено позже всех
4. порода, деревьев которой больше всего
5. аллея, на которой нет фонтанов

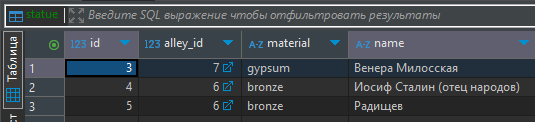
**Физическая модель БД для СУБД Postgresql**

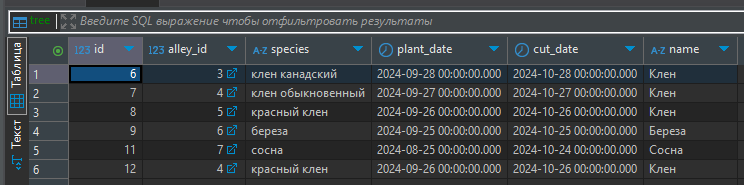


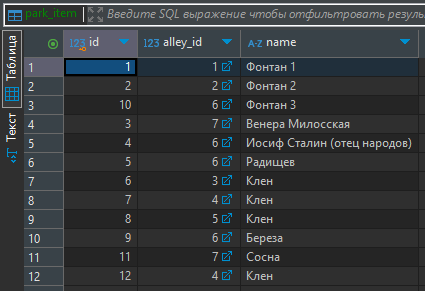
**Наборы данных**



****

****

****

****

**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**;

**return** group\_count;

**END**;

**$function$**

;

-- создание агрегатной функции

**CREATE** **AGGREGATE** unique\_by\_similar(**species**) (

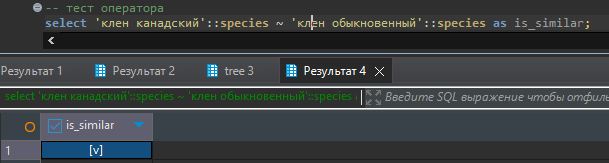
SFUNC = species\_acc, -- функция, собирающая массив

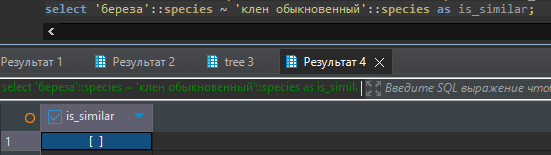
STYPE = **species**[], -- тип данных состояния

FINALFUNC = count\_unique\_similar, -- финализируюшая функция

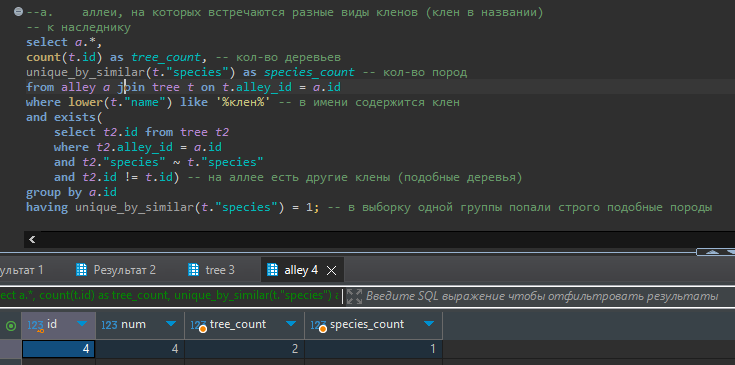
INITCOND = **"{}"**); -- начальный пустой массив

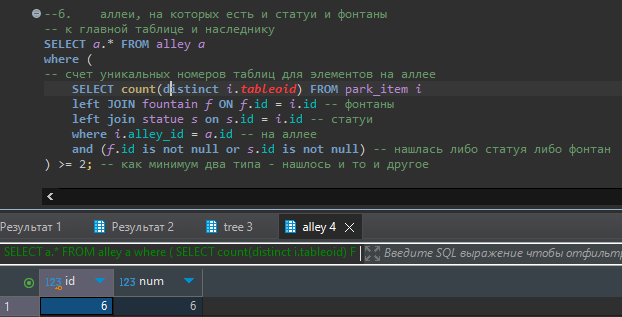
**Пример выполнения операторов**

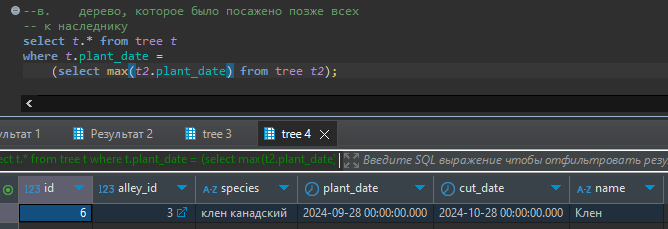


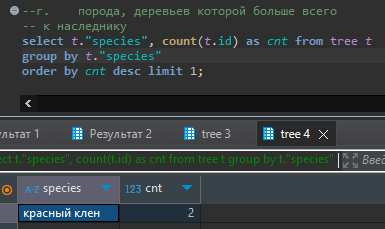


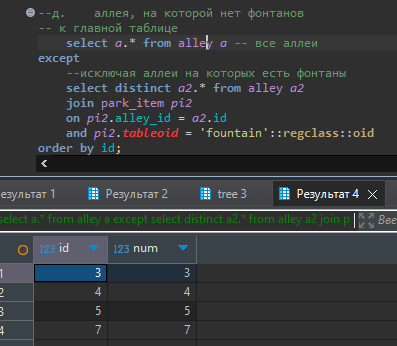
**Наборы данных, возвращаемых запросами**

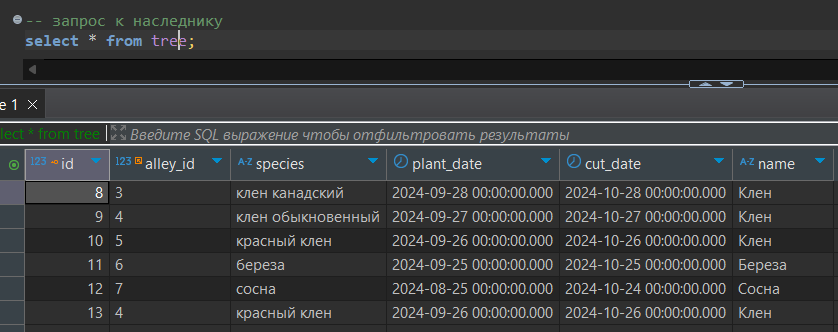
****

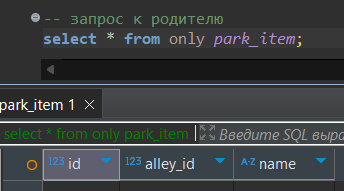
****

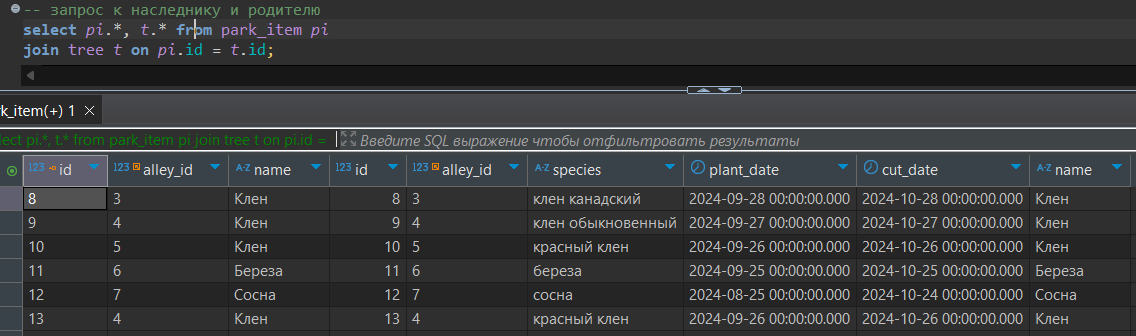
****

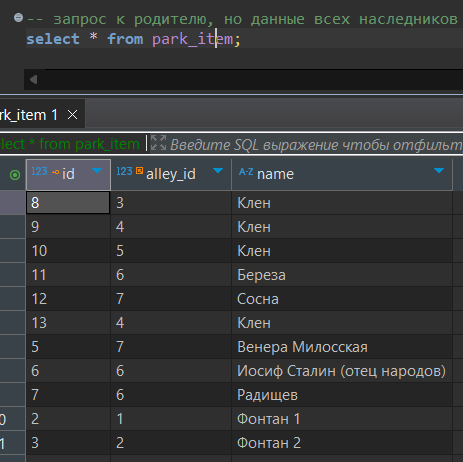
****

****

****

****

****

****

**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.