



**«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

О т ч е т
по лабораторной работе № 3

Название лабораторной работы: Пользовательские функции и процедуры

Дисциплина: Базы данных

Вариант 18

Студент гр. ИУ6-34Б _____ А. И. Мокшина
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____
(Подпись, дата) (И.О. Фамилия)

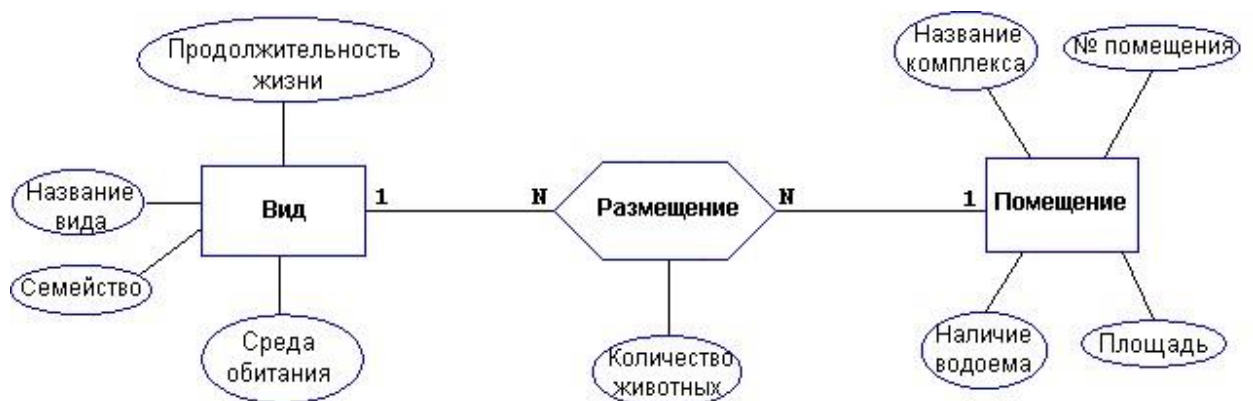
Цель:

Данная лабораторная работа призвана сформировать у студента понимание назначения пользовательских функции и процедур, их написание и использование.

Задачи:

- Получить теоретические знания о назначении функций и процедур БД.
- Изучить синтаксис функций и процедур.
- Научиться добавлять функции и процедуры в БД.
- Научиться удалять и изменять функции и процедуры.
- Научится использовать функции и процедуры.

Предметная область для практических заданий: **Зоопарк**



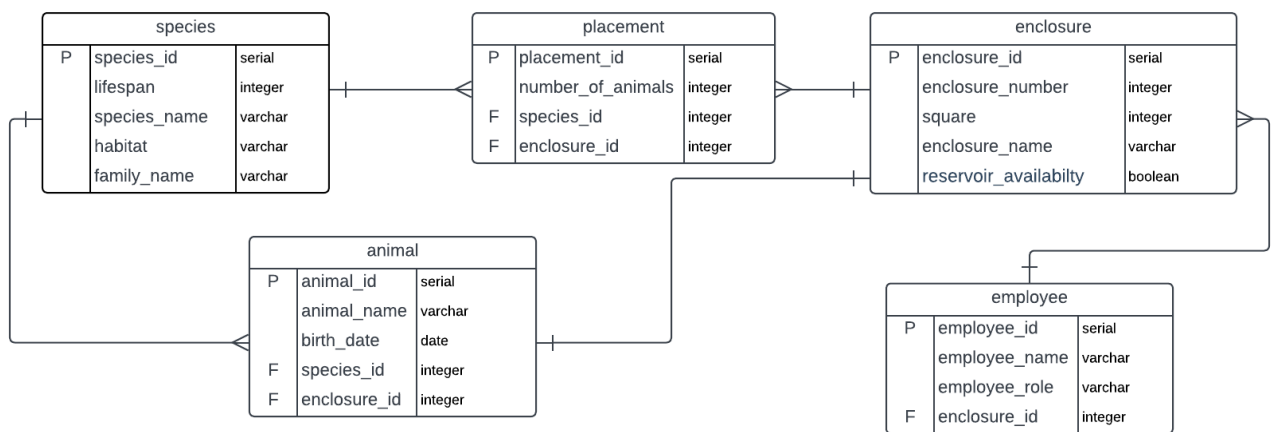


Рис. 1 – спроектированная БД

Написать хранимые функции и процедуры, выполняющие следующие операции для своей предметной области.

- Добавление одной записи, только через вызов процедуры, для таблиц Вид, Размещение, Помещение. Проверять там, где надо, на дублирование записей с выдачей сообщения об ошибке. Проверять на корректность ввода данных (номер помещения уникальный, название вида уникальное, в размещении существует такой id вида и помещения)

```

CREATE OR REPLACE PROCEDURE add_to_species(l INT, sn VARCHAR, h VARCHAR, fn VARCHAR)
    LANGUAGE plpgsql
AS $$
BEGIN
    IF l IS NULL OR sn IS NULL OR h IS NULL OR fn IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;
    -- Проверка на уникальность имени вида
    IF EXISTS (
        SELECT 1
        FROM species
        WHERE species_name = sn
    ) THEN
        RAISE EXCEPTION 'Имя вида уже существует: %', sn;
    END IF;

    INSERT INTO species (lifespan, species_name, habitat, family_name)
    VALUES (l, sn, h, fn)
    ON CONFLICT DO NOTHING;
END;
$$;
  
```

Создание процедуры добавления в таблицу Вид

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL add_to_species(23, 'seal', 'ocean', 'kitty');
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO		
Query returned successfully in 33 msec.		

Рис. 1 – Добавление в таблицу Вид

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL add_to_species(null, 'ioesjo', 'ocean', 'kitty');
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Введено недопустимое значение null CONTEXT: функция PL/pgSQL add_to_species(integer,character varying,character SQL-оператор: "CALL add_to_species(null, 'ioesjo', 'ocean', 'kitty')" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Введено недопустимое значение null SQL state: P0001		

Рис. 2 – Добавление в таблицу Вид и контроль корректности ввода данных

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL add_to_species(23, 'Miller', 'ocean', 'kitty');
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Имя вида уже существует: Miller CONTEXT: функция PL/pgSQL add_to_species(integer,character varying,charact SQL-оператор: "CALL add_to_species(23, 'Miller', 'ocean', 'kitty')" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Имя вида уже существует: Miller SQL state: P0001		

Рис. 3 – Добавление в таблицу Вид и контроль корректности ввода данных

```

CREATE OR REPLACE PROCEDURE add_to_enclosure(en INT, s INT, ena VARCHAR, ra BOOLEAN)
    LANGUAGE plpgsql
AS $$
BEGIN
    IF en IS NULL OR s IS NULL OR ena IS NULL OR ra IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    -- Проверка на уникальность имени вида
    IF EXISTS (
        SELECT 1
        FROM enclosure
        WHERE enclosure_number = en
    ) THEN
        RAISE EXCEPTION 'Номер помещения уже занят: %', en;
    END IF;

    INSERT INTO enclosure (enclosure_number, square, enclosure_name,
reservoir_availability)
        VALUES (en, s, ena, ra)
        ON CONFLICT DO NOTHING;
END;
$$;

```

Создание процедуры добавления в таблицу Помещение

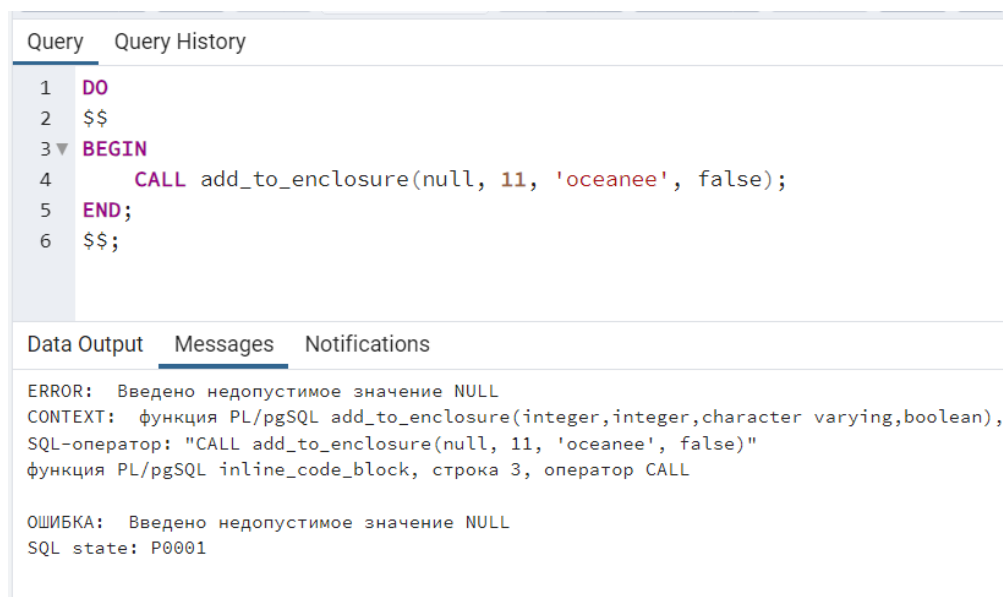


Рис. 4 – Добавление в таблицу помещение

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL add_to_enclosure(34, 11, 'oceanee', false);
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Номер помещения уже занят: 34 CONTEXT: функция PL/pgSQL add_to_enclosure(integer,integer,character v SQL-оператор: "CALL add_to_enclosure(34, 11, 'oceanee', false)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Номер помещения уже занят: 34 SQL state: P0001		

Рис. 5 – Добавление в таблицу помещение и контроль корректности ввода данных

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL add_to_enclosure(128, 50, 'Yummy Gummy', true);
5	END;
6	\$\$;
7	

Data Output	Messages	Notifications
DO Query returned successfully in 34 msec.		

Рис. 6 – Добавление в таблицу помещение и контроль корректности ввода данных

```
CREATE OR REPLACE PROCEDURE add_to_placement(noa INT, si INT, ei INT)
    LANGUAGE plpgsql
AS $$
BEGIN
    IF noa IS NULL OR si IS NULL OR ei IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    -- Проверка на существование вида и помещения
    IF NOT EXISTS (
        SELECT 1
```

```

        FROM species
        WHERE species_id = si
    ) THEN
        RAISE EXCEPTION 'Не существует вида: %', si;
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM enclosure
        WHERE enclosure_id = ei
    ) THEN
        RAISE EXCEPTION 'Не существует помещения: %', ei;
    END IF;

    INSERT INTO placement (number_of_animals, species_id, enclosure_id)
    VALUES (noa, si, ei)
    ON CONFLICT DO NOTHING;
END;
$$;

```

Создание процедуры добавления в таблицу Размещения

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL add_to_placement(10, 13, 48);
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO		
	Query returned successfully in 37 msec.	

Рис. 7 – Добавление в таблицу размещение

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL add_to_placement(null, 13, 48);
5	END;
6	\$\$;

Data Output	Messages	Notifications
	ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL add_to_placement(integer,integer,integ SQL-оператор: "CALL add_to_placement(null, 13, 48)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Введено недопустимое значение NULL SQL state: P0001	

Рис. 8– Добавление в таблицу размещение и контроль корректности ввода данных

Query	Query History
1 DO	
2 \$\$	
3 ▼ BEGIN	
4 CALL add_to_placement(5, 133, 428);	
5 END;	
6 \$\$;	

Data Output	Messages	Notifications
	ERROR: Не существует вида: 133 CONTEXT: функция PL/pgSQL add_to_placement(integer,integer,in SQL-оператор: "CALL add_to_placement(5, 133, 428)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL	
	ОШИБКА: Не существует вида: 133 SQL state: P0001	

Рис. 9 – Добавление в таблицу размещение и контроль корректности ввода данных

- Изменение одной записи, только через вызов процедуры, для таблиц Вид, Размещение, Помещение. Проверять там, где надо, на дублирование записей с выдачей сообщения об ошибке.

```
CREATE OR REPLACE PROCEDURE update_placement_by_id(pid INT, noa INT, si INT, ei INT)
LANGUAGE plpgsql
AS $$
BEGIN
    IF noa IS NULL OR si IS NULL OR ei IS NULL or pid is NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;
    -- Проверка на существование вида и помещения
    IF NOT EXISTS (
        SELECT 1
        FROM species
        WHERE species_id = si
    ) THEN
        RAISE EXCEPTION 'Не существует вида: %', si;
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM enclosure
        WHERE enclosure_id = ei
    ) THEN
        RAISE EXCEPTION 'Не существует помещения: %', ei;
    END IF;

    UPDATE placement
    SET number_of_animals = noa, species_id = si, enclosure_id = ei
    WHERE placement_id = pid;
END;
$;
```

Создание процедуры изменения записи в таблице Размещения

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL update_placement_by_id(23, 20, 10, 3);
5	END;
6	\$\$;

Data Output	Messages	Notifications
D0		
Query returned successfully in 40 msec.		

Рис. 10 – Обновление данных в таблице размещение

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL update_placement_by_id(null, 20, 10, 3);
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL update_placement_by_id(integer,integer,integer, SQL-оператор: "CALL update_placement_by_id(null, 20, 10, 3)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Введено недопустимое значение NULL SQL state: P0001		

Рис. 11 – Обновление данных в таблице размещение с проверкой на корректность

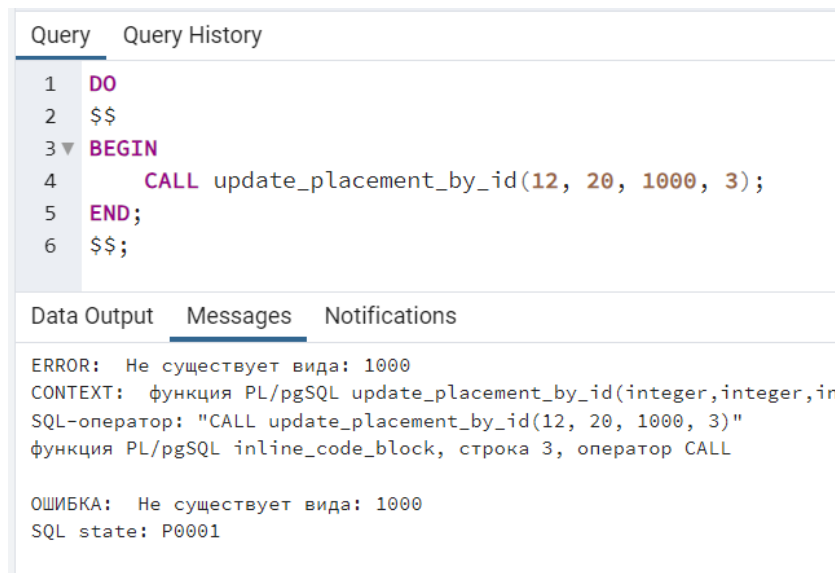


Рис. 12 – Обновление данных в таблице размещение с проверкой на корректность

```
CREATE OR REPLACE PROCEDURE update_species_by_family_name(fn VARCHAR, h VARCHAR)
    LANGUAGE plpgsql
AS $$
BEGIN
    IF fn IS NULL OR h IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM species
        WHERE family_name = fn
    ) THEN
        RAISE EXCEPTION 'Не существует семейства: %', fn;
    END IF;

    UPDATE species
    SET habitat = h
    WHERE family_name = fn;
END;
$;
```

Создание процедуры изменения записи в таблице Виды

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL update_species_by_family_name('Michael', 'desert');
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO		
Query returned successfully in 41 msec.		

Рис. 13 – Обновление данных в таблице Вид

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL update_species_by_family_name('Michael', null);
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL update_species_by_family_name(character varying, SQL-оператор: "CALL update_species_by_family_name('Michael', null)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL		
ОШИБКА: Введено недопустимое значение NULL SQL state: P0001		

Рис. 14 – Обновление данных в таблице Вид с проверкой на корректность

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL update_species_by_family_name('pupupup', 'desert');
5	END;
6	\$\$;

Data Output	Messages	Notifications
<p>ERROR: Не существует семейства: pupupup CONTEXT: функция PL/pgSQL update_species_by_family_name(character varying,character varying) SQL-оператор: "CALL update_species_by_family_name('pupupup', 'desert')" функция PL/pgSQL inline_code_block, строка 3, оператор CALL</p> <p>ОШИБКА: Не существует семейства: pupupup SQL state: P0001</p>		

Рис. 15 – Обновление данных в таблице Вид с проверкой на корректность

```
CREATE OR REPLACE PROCEDURE update_enclosure_reservoir_by_square(s INT, ra BOOLEAN)
LANGUAGE plpgsql
AS $$
BEGIN
    IF s IS NULL OR ra IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM enclosure
        WHERE square > s
    ) THEN
        RAISE EXCEPTION 'Не существует площади больше чем: %', s;
    END IF;

    UPDATE enclosure
    SET reservoir_availability = ra
    WHERE square > s;
END;
$;
```

Создание процедуры изменения записи в таблице Помещение

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL update_enclosure_reservoir_by_square(100, true) ;
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Не существует площади больше чем: 100 CONTEXT: функция PL/pgSQL update_enclosure_reservoir_by_square(integer,boolean) SQL-оператор: "CALL update_enclosure_reservoir_by_square(100, true)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Не существует площади больше чем: 100 SQL state: P0001		

Рис. 16 – Обновление данных в таблице Помещение с проверкой на корректность

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL update_enclosure_reservoir_by_square(70, true) ;
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO Query returned successfully in 33 msec.		

Рис. 17 – Обновление данных в таблице Помещение

Query	Query History
1	DO
2	\$\$
3▼	BEGIN
4	CALL update_enclosure_reservoir_by_square(null, true) ;
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL update_enclosure_reservoir_by_square(integer,bool SQL-оператор: "CALL update_enclosure_reservoir_by_square(null, true)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Введено недопустимое значение NULL SQL state: P0001		

Рис. 18 – Обновление данных в таблице Помещение с проверкой на корректность

- Удаление одной записи только, через вызов процедуры, для таблиц Вид, Размещение, Помещение. Проверять на возможность удаления. Например, удаляем вид или помещение, которые задействованы в таблице Размещение или Животное. Удаляем помещение, в котором живут животные или работает сотрудник.

```
CREATE OR REPLACE PROCEDURE delete_enclosure_by_enclosure_number(en INT)
LANGUAGE plpgsql
AS $$
BEGIN
    IF en IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    -- Проверка на удаляем помещение, в котором живут животные или работает
    сотрудник.
    IF EXISTS (
        SELECT 1
        FROM enclosure e, animal an
        WHERE enclosure_number = en and e.enclosure_id = an.enclosure_id
    ) THEN
        RAISE EXCEPTION 'В помещении живут животные: %', en;
    END IF;

    IF EXISTS (
        SELECT 1
        FROM enclosure e, employee em
        WHERE enclosure_number = en and e.enclosure_id = em.enclosure_id
    ) THEN
        RAISE EXCEPTION 'В помещении работают сотрудники: %', en;
```

```

END IF;

DELETE FROM enclosure WHERE enclosure_number = en;
END;
$$;

```

Создание процедуры удаления записи в таблице Помещение

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL delete_enclosure_by_enclosure_number(23);
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: В помещении живут животные: 23 CONTEXT: функция PL/pgSQL delete_enclosure_by_enclosure_number(integer), строка 4, оператор SQL-оператор: "CALL delete_enclosure_by_enclosure_number(23)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: В помещении живут животные: 23 SQL state: P0001		

Рис. 19 – Удаление данных в таблице Помещение с проверкой на корректность

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL delete_enclosure_by_enclosure_number(939);
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO Query returned successfully in 36 msec.		

Рис. 20 – Удаление данных в таблице Помещение

```

CREATE OR REPLACE PROCEDURE delete_species_by_name(sn VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    IF sn IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    -- Проверка на задействованы в таблице Размещение или Животное

```

```

IF EXISTS (
    SELECT 1
    FROM species s, animal an
    WHERE species_name = sn and s.species_id = an.species_id
) THEN
    RAISE EXCEPTION 'Существуют животные этого вида: %', sn;
END IF;

IF EXISTS (
    SELECT 1
    FROM species s, placement pl
    WHERE species_name = sn and s.species_id = pl.species_id
) THEN
    RAISE EXCEPTION 'Существуют размещения с заданным видом: %', sn;
END IF;

DELETE FROM species WHERE species_name = sn;
END;
$$;

```

Создание процедуры удаления записи в таблице Вид

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL delete_species_by_name(null);
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL delete_species_by_name(character var: SQL-оператор: "CALL delete_species_by_name(null)" функция PL/pgSQL inline_code_block, строка 3, оператор CALL		
ОШИБКА: Введено недопустимое значение NULL SQL state: P0001		

Рис. 21 – Удаление данных в таблице Вид с проверкой на корректность

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL delete_species_by_name('Miller');
5	END;
6	\$\$;

Data Output	Messages	Notifications
ERROR: Существуют животные этого вида: Miller CONTEXT: функция PL/pgSQL delete_species_by_name(character varying) SQL-оператор: "CALL delete_species_by_name('Miller')" функция PL/pgSQL inline_code_block, строка 3, оператор CALL ОШИБКА: Существуют животные этого вида: Miller SQL state: P0001		

Рис. 22 – Удаление данных в таблице Вид с проверкой на корректность

Query	Query History
1	DO
2	\$\$
3	BEGIN
4	CALL delete_species_by_name('Fox');
5	END;
6	\$\$;

Data Output	Messages	Notifications
DO Query returned successfully in 40 msec.		

Рис. 23 – Удаление данных в таблице Вид

```
CREATE OR REPLACE PROCEDURE delete_placement_by_id(pi INT)
LANGUAGE plpgsql
AS $$
BEGIN
    IF pi IS NULL
    THEN
        RAISE EXCEPTION 'Введено недопустимое значение NULL';
    END IF;

    DELETE FROM placement WHERE placement_id = pi;
END;
$$;
```

Создание процедуры удаления записи в таблице Размещение

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL delete_placement_by_id(38);
5	END;
6	\$\$\$;

Data Output	Messages	Notifications
DO		
Query returned successfully in 38 msec.		

Рис. 24 – Удаление данных в таблице Размещение

Query	Query History
1	DO
2	\$\$
3 ▼	BEGIN
4	CALL delete_placement_by_id(null);
5	END;
6	\$\$\$;

Data Output	Messages	Notifications
	ERROR: Введено недопустимое значение NULL CONTEXT: функция PL/pgSQL delete_placement_by_id(integer) SQL-оператор: "CALL delete_placement_by_id(null)" функция PL/pgSQL inline_code_block, строка 3, оператор CAL	
	ОШИБКА: Введено недопустимое значение NULL SQL state: P0001	

Рис. 25 – Удаление данных в таблице Размещение с проверкой на корректность

- Запрос животных по первичному ключу, по имени, по помещению.

```
CREATE OR REPLACE FUNCTION get_animal_by_id(ai INT)
  RETURNS SETOF animal
  STABLE LANGUAGE plpgsql
AS $$
BEGIN
  RETURN QUERY
  SELECT *
  FROM animal a
  WHERE a.animal_id = ai;
END;
$$$;
```

Создание функции, запрашивающей данные о животных по первичному ключу

Query

Query History

1

`select * from get_animal_by_id(1);`

Data Output

Messages

Notifications

<

Рис. 26 – демонстрация работы функции

```
CREATE OR REPLACE FUNCTION get_animal_by_name(an VARCHAR)
  RETURNS SETOF animal
  STABLE LANGUAGE plpgsql
AS $$
BEGIN
  RETURN QUERY
  SELECT *
  FROM animal a
  WHERE a.animal_name = an;
END;
$$;
```

Создание функции, запрашивающей данные о животных по имени

Query

Query History

1

select * from get_animal_by_name('John');

Data Output

Messages

Notifications

	animal_id integer	animal_name character varying	birth_date date	species_id integer	enclosure_id integer
1	5	John	2011-04-19	43	6
2	204	John	2021-02-16	39	28
3	217	John	2016-05-02	17	2
4	257	John	2006-05-02	12	24
5	339	John	2003-05-30	45	23
6	379	John	1996-08-02	10	24
7	413	John	2005-06-14	16	4
8	452	John	1995-11-16	24	15
9	478	John	2010-06-09	7	21
10	614	John	2012-07-14	1	1
11	829	John	2017-06-07	34	15
12	1000	John	2003-08-21	43	17

Рис. 27 – демонстрация работы функции

```
CREATE OR REPLACE FUNCTION get_animal_by_enclosure(encl INT)-- enclosure number
ВВОДИМ
    RETURNS SETOF animal
    STABLE LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT a.*
    FROM animal a
    JOIN enclosure e ON e.enclosure_id = a.enclosure_id
    WHERE e.enclosure_number = encl;
END;
$$;
```

Создание функции, запрашивающей данные о животных по номеру вольера

Query Query History

1 select * from get_animal_by_enclosure(13);

Data Output Messages Notifications

	animal_id integer	animal_name character varying	birth_date date	species_id integer	enclosure_id integer
1	57	Craig	1996-04-01	12	10
2	127	Terri	1987-11-01	13	10
3	200	Karen	1990-05-29	2	10
4	248	Tiffany	2002-03-20	34	10
5	252	Joshua	2020-08-16	2	10
6	255	Amy	1991-04-20	11	10
7	268	Timothy	1988-03-13	4	10
8	313	Madison	2008-07-08	6	10
9	327	Michael	2023-02-17	43	10
10	328	Melanie	1999-03-27	42	10
11	334	Michael	2016-06-24	41	10
12	368	Katherine	2014-09-02	8	10
13	409	Anthony	2016-02-21	12	10
14	468	Nathan	2010-06-28	31	10
15	527	Eric	1990-11-07	20	10
16	573	Daniel	2007-01-22	41	10

Рис. 28 – демонстрация работы функции

- Запрос на вывод вида животного проживающего в заданном помещении.

```
CREATE OR REPLACE FUNCTION get_species_by_enclosure(encl INT) -- enclosure number
ВВОДИМ
    RETURNS SETOF species
    STABLE LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT s.*
    FROM placement p, enclosure e, species s
```










```
WHERE e.enclosure_number = encl and e.enclosure_id = p.enclosure_id and  
p.species_id = s.species_id;  
END;  
$$;
```

Создание функции, запрашивающей данные о виде животного по вольеру

Query Query History

1 `select * from get_species_by_enclosure(11);`

Data Output Messages Notifications








	species_id integer 	lifespan integer 	species_name character varying 	habitat character varying 	family_name character varying 
1	29	5	Gonzales	ocean	Karen

Рис. 29 – демонстрация работы функции

Вывод: были освоены навыки создания и использования пользовательских функций и процедур.