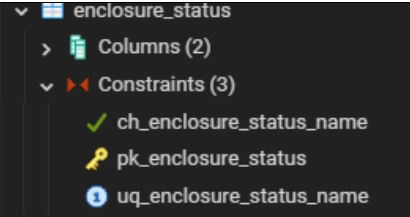


ЭТАПЫ ВЫПОЛНЕНИЯ

- 1. Сопровождение таблиц:
 - 1.1. Создание файла;
 - 1.2. Сопровождение объектов.

Таблица 1 – Сопровождение таблиц баз данных

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
Статус вольера (Enclosure_Status)	Enclosure_Status_Code	Serial	Да	Суррогатный ключ		<pre>create table Enclosure_Status (Enclosure_Status_Code serial not null constraint PK_Enclosure_Status primary key, Enclosure_Status_Name varchar(100) not null constraint UQ_Enclosure_Status_Name unique); create index if not exists index_Enclosure_Status_Code on Enclosure_Status(Enclosure_Status_Code); create index if not exists index_Enclosure_Status_Name on Enclosure_Status(Enclosure_Status_Name);</pre>	
	Enclosure_Status_Name	Varchar (100)	Да	Уникальное			
	Work_List_Code	Serial	Да	Суррогатный ключ		<pre>create table Work_List (</pre>	

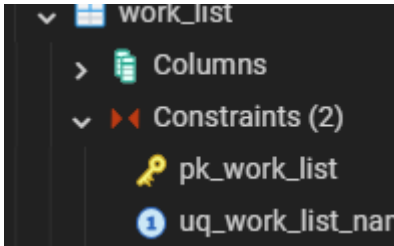
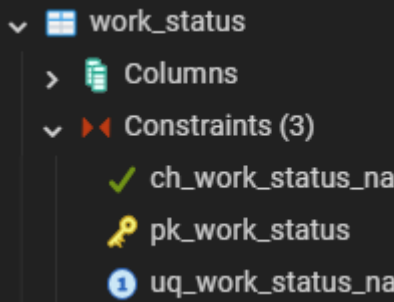
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

2

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
Перечень работ (Work_List)	Work_List_Name	Varchar(100)	Да			Work_List_Code serial not null constraint PK_Work_List primary key, Work_List_Name varchar(100) not null constraint UQ_Work_List_Name unique, Work_List_Interval interval not null);	
	Work_List_Interval	Interval	Да	> 0, <= Дата конца работ – Дата начала работ		create index if not exists index_Work_List_Code on Work_List (Work_List_Code); create index if not exists index_Work_List_Name on Work_List (Work_List_Name);	
Статус работ (Work_Status)	Work_Status_Code	Serial	Да	Суррогатный ключ			
	Work_Status_Name	Varchar(25)	Да	Уникальное [ожидается начало, ведутся ремонтные работы, завершён]		create table Work_Status (Work_Status_Code serial not null constraint PK_Work_Status primary key, Work_Status_Name varchar(100) not null constraint	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

3

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
						UQ_Work_Status_Name unique constraint CH_Work_Status_Name check (Work_Status_Name in ('Ожидается начало', 'Ведутся ремонтные работы', 'Завершен'))); create index if not exists index_Work_Status_Code on Work_Status (Work_Status_Code); create index if not exists index_Work_Status_Name on Work_Status (Work_Status_Name);	
План работ (Work_Plan)	Work_Plan_Code	Serial	Да	Суррогатный ключ		create table Work_Plan (Work_Plan_Code serial not null constraint PK_Work_Plan primary key, Work_Plan_Numbe r varchar(20) not null constraint UQ_Work_Plan_Number unique constraint CH_Work_Plan_Number	
	Work_Plan_Number	Varchar(100)	Да	Уникальное, (ГРМ)- (\d{2})- (\d{10})			
	Work_Plan_Date	Date	Да	= Текущая дата			
	Work_Plan_Enclosure	Int	Да	Внешний ключ			

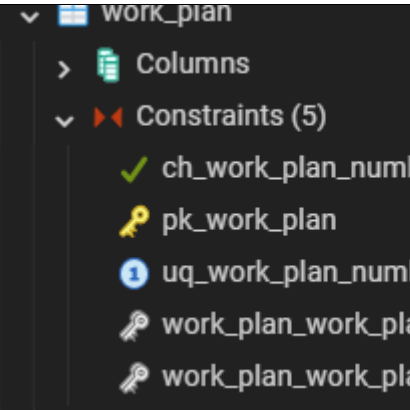
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

4

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Work_Plan_Start_Date	Date	Да	>= Дата заявки		<pre> check (Work_Plan_Number similar to '(ГРМ)-(\d{2})- (\d{10}')), Work_Plan_Date date not null, Work_Plan_Start_ Date date not null, Work_Plan_End_D ate date not null, Work_Plan_Instruct ion varchar(100) not null, Work_Plan_Enclos ure int not null references Enclosure (ID_Enclosure), Work_Plan_Status int not null references Work_Status (Work_Status_Code)); create index if not exists index_Work_Plan_Code on Work_Plan (Work_Plan_Code); create index if not exists index_Work_Plan_Number on Work_Plan (Work_Plan_Number); </pre>	
	Work_Plan_End_Date	Date	Да	> Дата начала работ			
	Work_Plan_Instruction	Varchar(100)	Да				
	Work_Plan_Status	Varchar(25)	Да	Внешний ключ			
Перечень и план работ	Work_List_Plan_Code	Serial	Да	Суррогатный ключ		create table Work_List_Plan (

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

5

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
(Work_List_Plan)	Work_List_Code	Int	Да	Уникальное, Внешний ключ		Work_List_Plan_Code serial not null constraint PK_Work_List_Plan primary key, Code_Work_List int not null references Work_List (Work_List_Code) constraint UQ_Code_Work_List unique, Code_Work_Status int not null references Work_Status (Work_Status_Code));	<div> <div>work_list_plan</div> <div>Columns</div> <div>Constraints (4)</div> <div> <div>pk_work_list_plan</div> <div>uq_code_work_list</div> <div>work_list_plan_code</div> <div>work_list_plan_code</div> </div> </div>
	Work_Plan_Code	Int	Да	Внешний ключ		create index if not exists index_Work_List_Plan_Code on Work_List_Plan (Work_List_Plan_Code);	
Ареал обитания (Habitat)	Habitat_Name	Varchar(100)	Да	Уникальное	create table if not exists Habitat (Habitat_Code serial not null constraint PK_Habitat primary key, Habitat_N	alter table Habitat add constraint UQ_Habitat_Name unique (Habitat_Name);	<div> <div>habitat</div> <div>Columns</div> <div>Constraints (2)</div> <div> <div>pk_habitat</div> <div>uq_habitat_n</div> </div> </div>

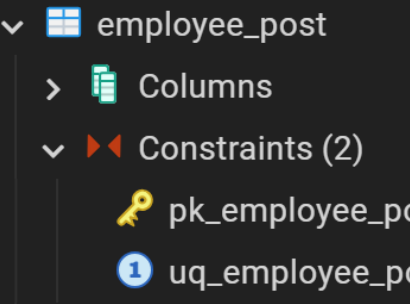
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

6

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Habitat_Code	Serial	Да	Суррогатный ключ	ame varchar(100) not null);		
Должность сотрудника (Employee_Post)	Employee_Post_Code	Int	Да	Суррогатный ключ	create table if not exists Employee_Post (Employee_ Post_Code serial not null constraint PK_Employee_Post primary key, Employee_ Post_Name varchar(50) not null);	alter table Employee_Post add constraint UQ_Employee_Post_Name unique (Employee_Post_Name);	
	Employee_Post_Name	Varchar(50)	Да	Уникальное			

Дисциплина: СУБД (PostgreSQL, MySQL)
 Контрольная точка: №8
 Группа: 2ПЗ.22
 Студент: Новгородов Иван

7

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
Посетитель (Visitor)	ID_Visitor	Serial	Да	Суррогатный ключ	create table if not exists Visitor (ID_Visitor serial not null constraint PK_Visitor primary key, Login_Visitor varchar(50) not null, Name_Visitor varchar(50) not null, Surname_Visitor varchar(50) not null, Patronymic_Visitor varchar(50) null, Passport_Series_Visitor varchar(4) not null, Passport_	alter table Visitor add constraint UQ_Login_Visitor unique (Login_Visitor), add constraint CH_Login_Visitor check (Login_Visitor similar to '[\w]{6,}'), alter column Patronymic_Visitor set default 'Нет данных', add constraint CH_Passport_Series_Visitor check (Passport_Series_Visitor similar to '[0-9]{4}'), add constraint CH_Passport_Number_Visitor check (Passport_Number_Visitor similar to '[0-9]{6}'), add constraint CH_Benefits_Visitor check (Benefits_Visitor >= 0), add constraint CH_Password_Visitor check (Password_Visitor similar to '[\w0-9!@#\$%^&*()]{8,}');	<div> <div>visitor</div> <div>Columns</div> <div>Constraints (7)</div> <div> <div>ch_benefits_visitor</div> <div>ch_login_visitor</div> <div>ch_passport_number</div> <div>ch_passport_series</div> <div>ch_password_visitor</div> <div>pk_visitor</div> <div>uq_login_visitor</div> </div> </div>
	Login_Visitor	Varchar(50)	Да	Уникальное Length >= 6, [a-Z]			
	Name_Visitor	Varchar(50)	Да				
	Surname_Visitor	Varchar(50)	Да				
	Patronymic_Visitor	Varchar(50)	Нет	Default “Нет данных”			
	Passport_Series_Visitor	Varchar(4)	Да	[0-9]{4}			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

8

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Passport_Number_Visitor	Varchar(6)	Да	[0-9]{6}	Number_Visitor varchar(6) not null,		
	Benefits_Visitor	Int	Да	>= 0	Benefits_Visitor int not null,		
	Password_Visitor	Varchar(36)	Да	Length >= 8, [a-Z][0-9][!@#\$%^&*()]	Password_Visitor varchar(36) not null);		
Вид посетителя (Visitor Type)	ID_Visitor_Type	Serial	Да	Суррогатный ключ	create table if not exists Visitor_Document (alter table Visitor_Type add constraint UQ_Name_Visitor_Type unique (Name_Visitor_Type);	<div> <div>visitor_type</div> <div>Columns</div> <div>Constraints (2)</div> <div> <div>pk_visitor_type</div> <div>uq_name_visitor_type</div> </div> </div>
	Name_Visitor_Type	Varchar(50)	Да	Уникальное	ID_Document serial not null constraint PK_Visitor_Document primary key, Number_Document varchar(20) not null, ID_Visitor_Type int not null references Visitor_Type (ID_Visitor_Type),		

Дисциплина: СУБД (PostgreSQL, MySQL)
 Контрольная точка: №8
 Группа: 2ПЗ.22
 Студент: Новгородов Иван

9

Таблица	Поле	Тип данных	Обязательное	Ограничения	Было	Стало	Результат
					ID_Visitor int not null references Visitor (ID_Visitor));		
Документ посетителя (Visitor Document)	ID_Document	Serial	Да	Суррогатный ключ	create table if not exists Visitor_Document (ID_Document serial not null constraint PK_Visitor_Document primary key, Number_Document varchar(20) not null, ID_Visitor_Type int not null references Visitor_Type (ID_Visitor_Type), ID_Visitor int not null references Visitor (ID_Visitor)	alter table Visitor_Document add constraint UQ_Number_Document unique (Number_Document), add constraint CH_Number_Document check (Number_Document similar to '\d+\/?\d*-\d*-\d*-\d*'); visitor_document Columns Constraints (5) ch_number_document pk_visitor_document uq_number_document visitor_document_id_v visitor_document_id_v	
	ID_Visitor_Type	Int	Да	Внешний ключ			
	ID_Visitor	Int	Да	Внешний ключ			
	Number_Document	Varchar(20)	Да	Уникальное ([0-9]{6}, [0-9]{3}-[0-9]{3}-[0-9]{3}-[0-9]{2}, [0-9]{1,2}/[0-9]{2,3}-[0-9]{8}-[0-9]{2})			

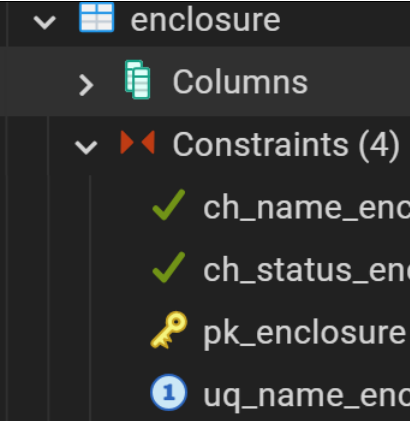
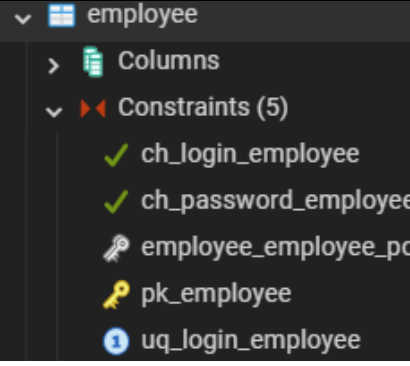
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

10

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
);		
Вольер (Enclosure)	ID_Enclosure	Serial	Да	Суррогатный ключ	create table if not exists Enclosure (ID_Enclosure serial not null constraint PK_Enclosure primary key, Name_Enclosure varchar(50) not null, Status_Enclosure varchar(16) not null);	alter table Enclosure add constraint UQ_Name_Enclosure unique (Name_Enclosure), add constraint CH_Name_Enclosure check (Name_Enclosure similar to '[A-Z]{1}[0-9]{1}'), add constraint CH_Status_Enclosure check (Status_Enclosure similar to 'Открыт Переоборудование Ремонт');	
	Name_Enclosure	Varchar(50)	Да	Уникальное [A-Z]{1}[0-9]{1}			
	Status_Enclosure	Varchar(16)	Да	Открыт, Переоборудование, Ремонт			
Сотрудник (Employee)	ID_Employee	Serial	Да	Суррогатный ключ	create table if not exists Employee (ID_Employee serial not null constraint PK_Employee primary key, Login_Employee varchar(100) not null, Surname_Employee varchar(50) not null, Name_Employee varchar(50) not null, Patronymic_Employee varchar(50) not null,);	create table if not exists Employee (ID_Employee serial not null constraint PK_Employee primary key, Login_Employee varchar(100) not null, Surname_Employee varchar(50) not null, Name_Employee varchar(50) not null, Patronymic_Employee varchar(50) not null,);	
	Login_Employee	Varchar(100)	Да	Уникальное Length >= 6, [a-Z]			
	Surname_Employee	Varchar(50)	Да				
	Name_Employee	Varchar(50)	Да				
	Patronymic_Employee	Varchar(50)	Нет	Default "Нет данных"			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

11

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Password_Employee	Varchar(36)	Да	Length >= 8, [a-Z][0-9][!@#%&*()]	Surname_Employee varchar(50) not null,	Name_Employee varchar(50) not null,	
	Employee_Post_Code	int	Да	Внешний ключ	Name_Employee varchar(50) not null, Patronymic_Employee varchar(50) null, Password_Employee varchar(36) not null, Employee_Post_Code int not null references Employee_Post (Employee_Post_Code); Employee_Post_Code int not null references Employee_Post (Employee_Post_Code);	Patronymic_Employee varchar(50) null, Password_Employee varchar(36) not null, Employee_Post_Code int not null references Employee_Post (Employee_Post_Code);	
Сотрудники и вольеры (Employee_Enclosure)	ID_Employee_Enclosure	Serial	Да	Суррогатный ключ	create table if not exists Employee_Enclosure (
	ID_Employee	Int	Да	Внешний ключ			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

12

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	ID_Enclosure	Int	Да	Внешний ключ	<p>ID_Employee_Enclosure serial not null constraint PK_Employee_Enclosure primary key,</p> <p>ID_Employee int not null references Employee (ID_Employee),</p> <p>ID_Enclosure int not null references Enclosure (ID_Enclosure);</p>		<div> <div>Constraints (3)</div> <div> <div>employee_enclosure</div> <div>employee_enclosure</div> <div>pk_employee_enclosure</div> </div> </div>
Животное (Animal)	ID_Animal	Serial	Да	Суррогатный ключ	create table if not exists Animal (alter table Animal	
	Number_Animal	Varchar(11)	Да	Уникальное [A-Z]{2,3}[0-9]{8}	ID_Animal serial not null constraint PK_Animal primary key,	add constraint UQ_Number_Animal unique (Number_Animal),	
	ID_Animal_Type	Int	Да	Внешний ключ		add constraint CH_Number_Animal check (Number_Animal similar to '[A-Z]+\d{8}');	
	Habitat_Code	Int	Да	Внешний ключ	Number_A		

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

13

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Description_Animal	Varchar(100)	Да		animal varchar(11) not null,		<div> <div>animal</div> <div>Columns</div> <div>Constraints (7)</div> <div> <div>animal_habitat_code_</div> <div>animal_id_animal_type</div> <div>animal_id_enclosure_t</div> <div>animal_id_territory_fk</div> <div>ch_number_animal</div> <div>pk_animal</div> <div>uq_number_animal</div> </div> </div>
	Picture_Animal	Varchar(100)	Да		Description_Animal varchar(100) not null,		
	ID_Enclosure	Int	Да	Внешний ключ	Picture_Animal varchar(100) not null,		
	ID_Territory	Int	Да	Внешний ключ	ID_Animal_Type int not null references Animal_Type (ID_Animal_Type), Habitat_Code int not null references Habitat (Habitat_Code), ID_Enclosure int not null references Enclosure (ID_Enclosure), ID_Territory int not null		

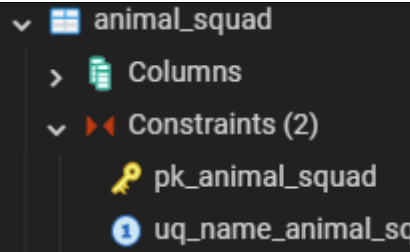
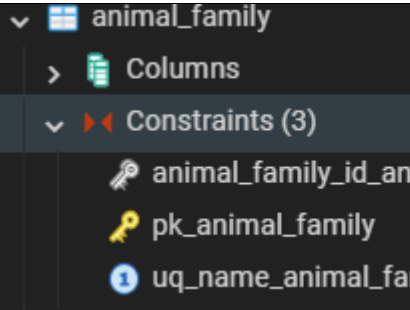
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

14

Таблица	Поле	Тип данных	Обязательное	Ограничения	Было	Стало	Результат
					references Territory (ID_Territory);		
Отряд животного (Animal Squad)	ID_Animal_Squad	Serial	Да	Суррогатный ключ	create table if not exists Animal_Squad (alter table Animal_Squad add constraint UQ_Name_Animal_Squad unique(Name_Animal_Squad);	
	Name_Animal_Squad	Varchar(100)	Да	Уникальное	ID_Animal_Squad serial not null constraint PK_Animal_Squad primary key, Name_Animal_Squad varchar(100) not null		
Семейство животного (Animal Family)	ID_Animal_Family	Serial	Да	Суррогатный ключ	create table if not exists Animal_Family (alter table Animal_Family add constraint UQ_Name_Animal_Family unique(Name_Animal_Family);	
	Name_Animal_Family	Varchar(100)	Да	Уникальное	ID_Animal_Family serial not null constraint PK_Animal_Family primary key,		
	ID_Animal_Squad	Int	Да	Внешний ключ	Name_Animal_Family		

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

15

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
					varchar(100) not null, ID_Animal_Squad int not null references Animal_Squad(ID_Animal_Squad));		
Вид животного (Animal Type)	ID_Animal_Type	Serial	Да	Суррогатный ключ	create table if not exists Animal_Type (ID_Animal_Type serial not null constraint PK_Animal_Type primary key, Name_Animal_Type varchar(100) not null, ID_Animal_Family int not null references Animal_Family(ID_Animal_Family));	alter table Animal_Family add constraint UQ_Name_Animal_Family unique(Name_Animal_Family);	<div> <div>animal_type</div> <div>Columns</div> <div>Constraints (3)</div> <div> <div>animal_type_id_animal_type</div> <div>pk_animal_type</div> <div>uq_name_animal_type</div> </div> </div>
	Name_Animal_Type	Varchar(100)	Да	Уникальное			
	ID_Animal_Family	Int	Да	Внешний ключ			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

16

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
Территория для посетителя (Territory)	ID_Territory	Serial	Да	Суррогатный ключ	create table if not exists Territory (ID_Territory serial not null constraint PK_Territory primary key, Name_Territory varchar(100) not null, Price_Territory decimal(5, 2) not null);	alter table Territory add constraint UQ_Name_Territory unique (Name_Territory), add constraint CH_Price_Territory check (Price_Territory >= 0);	<div> <div>territory</div> <div>Columns</div> <div>Constraints (3)</div> <div> <div>ch_price_territory</div> <div>pk_territoty</div> <div>uq_name_territory</div> </div> </div>
	Name_Territory	Varchar(100)	Да	Уникальное			
	Price_Territory	Decimal(5,2)	Да	>= 0			
Билет (Ticket)	ID_Ticket	Serial	Да	Суррогатный ключ	create table if not exists Ticket (ID_Ticket serial not null constraint PK_Ticket primary key, Number_Ticket varchar(16) not null,);	alter table Ticket add constraint CH_Number_Ticket check (Number_Ticket similar to 'ПБЗ-\d{9}\\d{2}'), add constraint CH_Price_Ticket check (Price_Ticket >= 0), add constraint CH_Total_Sum_Ticket check (Total_Sum_Ticket >= 0);	<div> <div>ticket</div> <div>Columns</div> <div>Constraints (5)</div> <div> <div>ch_number_ticket</div> <div>ch_price_ticket</div> <div>ch_total_sum_ticket</div> <div>pk_ticket</div> <div>ticket_id_visitor_fkey</div> </div> </div>
	ID_Visitor	Int	Да	Внешний ключ			
	Number_Ticket	Varchar(15)	Да	ПБЗ-[0-9]{9}/Последние две цифры тек. года			
	Datetime_Ticket	Timestamp	Да	= Текущая дата			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

17

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
	Price_Ticket	Decimal(6,2)	Да	≥ 0	Datetime_Ticket timestamp not null,		
	Total_Sum_Ticket	Decimal(6,2)	Да	≥ 0	Price_Ticket decimal(6, 2) not null, Total_Sum_Ticket decimal(6, 2) not null, ID_Visitor int not null references Visitor (ID_Visitor);		
Территория и билет (Territory_Ticket)	ID_Territory_Ticket	Serial	Да	Суррогатный ключ	create table if not exists Territory_Ticket (<div> <div>territory_ticket</div> <div>Columns</div> <div>Constraints (3)</div> <div> <div>pk_territory_ticket</div> <div>territory_ticket_id_territo</div> <div>territory_ticket_id_ticket</div> </div> </div>
	ID_Territory	Int	Да	Внешний ключ	ID_Territory_Ticket serial not null constraint PK_Territory_Ticket primary key,		
	ID_Ticket	Int	Да	Внешний ключ	ID_Territory int not null		

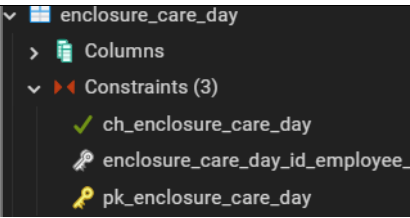
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

18

Таблица	Поле	Тип данных	Обязательно	Ограничения	Было	Стало	Результат
					references Territory (ID_Territory), ID_Ticket int not null references Ticket (ID_Ticket));		
Вольер и дни ухода (Enclosure_Care_Day)	ID_Enclosure_Care_Day	Serial	Да	Суррогатный ключ	create table if not exists Enclosure_Care_Day (ID_Enclosure_Care_Day serial not null constraint PK_Enclosure_Care_Day primary key, ID_Employee_Enclosure int not null references Employee_Enclosure (ID_Employee_Enclosure), Enclosure_Care_Day varchar(11) not null	alter table Enclosure_Care_Day add constraint CH_Enclosure_Care_Day check (Enclosure_Care_Day similar to 'Понедельник Вторник Среда Четверг Пятница Суббота Воскресенье');	
	ID_Employee_Enclosure	Int	Да	Внешний ключ			
	Enclosure_Care_Day	Varchar(11)	Да	Понедельник, Вторник, Среда, Четверг, Пятница, Суббота, Воскресенье			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

19

Таблица	Поле	Тип данных	Обязательное	Ограничения	Было	Стало	Результат
);		
Время ухода (Care_Time)	ID_Caretime	Serial	Да	Суррогатный ключ	create table if not exists Care_Time (
	ID_Enclosure_Care_Day	Int	Да	Внешний ключ	ID_Care_Time serial not null constraint PK_Care_Time primary key,		
	Care_Time	Time	Да		ID_Enclosure_Care_Day int not null references Enclosure_Care_Day(ID_Enclosure_Care_Day), Care_Time time not null);		

2. Сопровождение хранимых процедур;

2.1. Создание файла;



8.2 KT Escort DB Procedure.sql

05.04.2024 9:58

2.2. Набор тестовых сценариев для сопровождения хранимых процедур;

Таблица 2 – Сценарии тестирования

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

20

№ Сценария	Характеристики
1.	Краткое описание теста
	Ввод существующей области, с выводом сообщения об ошибке.
	Поля ввода
	Название области
	Вводимые данные
	Океанариум, 100.00
	Ожидаемый результат
	Указанная область уже есть в справочнике!
2.	Краткое описание теста
	Автоматическое формирование номера плана ремонтных работ
	Поля ввода
	Дата формирования, Код-ключ вольера, Дата начала работ, Дата конца работ, Инструкция, Код-ключ плана .
	Вводимые данные
	Дата формирования: 01.05.2024, Вольер: X1, период: 01.06.2024-10.06.2024, перемещение: Тест, Статус: Ожидается начало.
	Ожидаемый результат
	ГРМ-24-00000000004
3.	Краткое описание теста
	Удаление статуса вольера
	Поля ввода
	Удаление статуса вольера
	Вводимые данные
	Открыт
	Ожидаемый результат
	Указанный статус не может быть удалён, т.к. он распределён к вольерам.
4.	Краткое описание теста

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

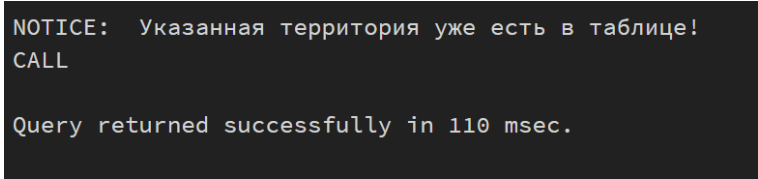
Студент: Новгородов Иван

21

№ Сценария	Характеристики
	Проверка сопоставления, названия животного с ареалом обитания.
	Поля ввода
	: Код-ключ животного, Код-ключ ареала
	Вводимые данные
	Животное: Орел, Ареал: Евразия.
	Ожидаемый результат
	Выбранный ареал уже есть у указанного животного!
5.	Краткое описание теста
	Автоматический перерасчёт стоимости билета, при добавлении дополнительной области.
	Поля ввода
	Код-ключ билета, Код-ключ области.
	Вводимые данные
	Билет: ПБЗ-000000001/23, Область: Контактный зоопарк.
	Ожидаемый результат
	1100

1.1. Сопровождение и тестирование хранимых процедур;

Таблица 3 – Реализация хранимых процедур

№ Сценария	Скрипт	Результат	Статус тестирования
1.	create or replace procedure Territory_Insert (p_Name_Territory varchar(100), p_Price_Territory decimal(2, 5))	 <pre>NOTICE: Указанная территория уже есть в таблице! CALL Query returned successfully in 110 msec.</pre>	Пройден

	<pre>language plpgsql as \$\$ begin insert into Territory (Name_Territory, Price_Territory) values (p_Name_Territory, p_Price_Territory); exception when others then raise notice 'Указанная территория уже есть в таблице!'; end; \$\$; call Territory_Insert('Океанариу м', 100.00); select * from territory;</pre>							
2.	create or replace procedure Work_Plan_Insert(p_Work_Plan_Date date, p_Work_Plan_Start_Date date,		work_plan_code [PK] integer	work_plan_number character varying (20)	work_plan_date date	work_plan_start_date date	work_plan_en date	Пройден
		1	1	ГРМ-24-0000000001	2024-01-05	2024-01-06	2024-10-06	
		2	2	ГРМ-24-0000000002	2024-01-05	2024-01-06	2024-10-06	
		3	3	ГРМ-24-0000000003	2024-01-05	2024-01-06	2024-10-06	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

23

	<pre>p_Work_Plan_End_Date date, p_Work_Plan_Instruction varchar(100), p_Work_Plan_Enclosure int, p_Work_Plan_Status int) language plpgsql as \$\$ declare work_plan_year varchar(4) := date_part('year', p_Work_Plan_Date); p_Work_plan_numbe r varchar(20) := 'TPM- ' substring(work_plan_year, 3, 2); work_plan_counter integer := (select count(*) from work_plan) + 1; work_plan_zeros varchar(11) :=</pre>		
--	---	--	--

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

24

<pre>LPAD(work_plan_counter::text, 10, '0'); begin p_Work_plan_number := p_Work_plan_number '-' work_plan_zeros; insert into Work_Plan(Work_Plan_Number, Work_Plan_Date, Work_Plan_Start_Date, Work_Plan_End_Date, Work_Plan_Instruction, Work_Plan_Enclosure, Work_Plan_Status) values (p_Work_Plan_Number, p_Work_Plan_Date, p_Work_Plan_Start_Date, p_Work_Plan_End_Date, p_Work_Plan_Instruction, p_Work_Plan_Enclosure, p_Work_Plan_Status); end; \$\$;</pre>		
---	--	--

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

25

3.	<pre>create or replace procedure Enclosure_Status_Delete (p_ID_Enclosure_Status int) language plpgsql as \$\$ declare p_Exist_Enclosures smallint := count(*) from Enclosure where status_enclosure = p_ID_Enclosure_Status; begin if(p_Exist_Enclosures > 0) then raise notice 'Данный статус вольера не может быть удалён, т.к. он используется в вольерах'; else delete from Enclosure where</pre>	<pre>NOTICE: Данный статус вольера не может быть удалён, т.к. он используется в вольерах CALL Query returned successfully in 80 msec.</pre>	Пройден
----	---	--	---------

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

26

	<pre> ID_Enclosure = p_ID_Enclosure; end if; end; \$\$; call Enclosure_Status_Delete(1);</pre>		
4.	<pre>create or replace procedure Animal_Update (p_ID_Animal int, p_Number_Animal varchar(11), p_Description_Animal varchar(100), p_Picture_Animal varchar(100), p_ID_Animal_Type int, p_Habitat_Code int, p_ID_Enclosure int, p_ID_Territory int) language plpgsql as \$\$ declare p_old_animal_habitat _code int := (select Habitat_Code from animal</pre>	<pre>NOTICE: Выбранный ареал уже есть у указанного животного! CALL Query returned successfully in 140 msec.</pre>	Пройден

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

27

<pre>where id_animal = p_id_animal); begin if(p_old_animal_habi tat_code = p_Habitat_Code) then raise notice 'Выбранный ареал уже есть у указанного животного!'; else update Animal set Number_Animal = p_Number_Animal, Description_Animal = p_Description_Animal, Picture_Animal = p_Picture_Animal, ID_Animal_Type = p_ID_Animal_Type,</pre>		
--	--	--

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

28

	<div>Habitat_Code = p_Habitat_Code,</div> <div>ID_Enclosure = p_ID_Enclosure,</div> <div>ID_Territory = p_ID_Territory</div> <div>where</div> <div>ID_Animal = p_ID_Animal;</div> <div>end if;</div> <div>end;</div> <div>\$\$;</div> <div>call animal_update(1, 's', 's', 'f', 1, 1, 1, 1);</div>																
5.	<div>create or replace procedure Territory_Ticket_Insert</div> <div>(p_ID_Territory int, p_ID_Ticket int)</div> <div>language plpgsql</div> <div>as \$\$</div> <div>declare</div>	<table><tr><th></th><th>id_ticket [PK] integer</th><th>number_ticket character varying (16)</th><th>datetime_ticket timestamp without time zone</th><th>price_ticket numeric (6,2)</th><th>total_sum_ticket numeric (6,2)</th><th>id_visitor integer</th></tr><tr><td>1</td><td>1</td><td>ПБ3-0000000001/23</td><td>2023-01-09 00:00:00</td><td>750.00</td><td>1100.00</td><td>1</td></tr></table>		id_ticket [PK] integer	number_ticket character varying (16)	datetime_ticket timestamp without time zone	price_ticket numeric (6,2)	total_sum_ticket numeric (6,2)	id_visitor integer	1	1	ПБ3-0000000001/23	2023-01-09 00:00:00	750.00	1100.00	1	Пройден
	id_ticket [PK] integer	number_ticket character varying (16)	datetime_ticket timestamp without time zone	price_ticket numeric (6,2)	total_sum_ticket numeric (6,2)	id_visitor integer											
1	1	ПБ3-0000000001/23	2023-01-09 00:00:00	750.00	1100.00	1											

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

29

	<pre> p_territory_ticket_old _sum decimal(6, 2) := (select total_sum_ticket from ticket where id_ticket = p_id_ticket); p_Visitor_Id int := (select id_visitor from ticket where id_ticket = p_id_ticket); p_Visitor_Benefits int := (select benefits_visitor from visitor where id_visitor = p_Visitor_Id); p_territory_price decimal(5, 2) := (select price_territory from territory where id_territory = p_ID_Territory); p_territory_ticket_ne w_sum decimal(6, 2) := 0; begin if (p_Visitor_Benefits > 0) then</pre>		
--	---	--	--

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

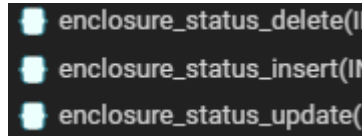
Студент: Новгородов Иван

30

	<pre> p_territory_ticket_new_sum w_sum := p_territory_ticket_old_sum + (p_territory_price p_Visitor_Benefits); else p_territory_ticket_new_sum w_sum := p_territory_ticket_old_sum + p_territory_price; end if; update ticket set total_sum_ticket = p_territory_ticket_new_sum where id_ticket = p_id_ticket; insert into Territory_Ticket (ID_Territory, ID_Ticket) values (p_ID_Territory, p_ID_Ticket); end; \$\$;</pre>	
--	---	--

Студент: Новгородов Иван

Таблица 4 – Реализация хранимых процедур

№ П П	Скрипт	Результат
1.	<pre>exception when others then raise notice 'Указанный ареал обитания уже есть в таблице!';</pre>	
2.	<pre>create or replace procedure Enclosure_Status_Update (p_enclosure_status_code int, p_enclosure_status_name Varchar(100)) language plpgsql as \$\$ begin update Enclosure_Status set enclosure_status_name = p_enclosure_status_name where</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

32

№ П П	Скрипт	Результат
	<pre> enclosure_status_code = p_enclosure_status_code; end; \$\$;</pre>	
3.	<pre>create or replace procedure Enclosure_Status_Delete (p_ID_Enclosure_Status int) language plpgsql as \$\$ declare p_Exist_Enclosures smallint := count(*) from Enclosure where status_enclosure = p_ID_Enclosure_Status; begin if(p_Exist_Enclosures > 0) then raise notice 'Данный статус вольера не может быть удалён, т.к. он используется в вольерах';</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

33

№ П П	Скрипт	Результат
	<pre>else delete from Enclosure where ID_Enclosure = p_ID_Enclosure; end if; end; \$\$;</pre>	
4.	<pre>create or replace procedure work_list_Insert (p_work_list_name varchar(100), p_work_list_interval interval) language plpgsql as \$\$ begin insert into work_list (work_list_name, work_list_interval) values (p_work_list_name, p_work_list_interval);</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

34

№ П П	Скрипт	Результат
	<pre>exception when others then raise notice 'Указанная работа уже есть в таблице!'; end; \$\$;</pre>	
5.	<pre>create or replace procedure work_list_Update (p_work_list_code int, p_work_list_name varchar(100), p_work_list_interval interval) language plpgsql as \$\$ begin update work_list set work_list_name = p_work_list_name, work_list_interval = p_work_list_interval where</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

35

№ П П	Скрипт	Результат
	<pre> work_list_code = p_work_list_code; end; \$\$;</pre>	
6.	<pre>create or replace procedure work_list_Delete (p_work_list_code int) language plpgsql as \$\$ declare p_Exist_Works smallint := count(*) from Work_List_plan where code_work_list = p_work_list_code; begin if(p_Exist_Works > 0) then raise notice 'Данная работа не может быть удалена, т.к. она используется в плане работ'; else</pre>	

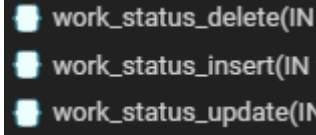
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

36

№ П П	Скрипт	Результат
	<pre>delete from work_list where work_list_code = p_work_list_code; end if; end; \$\$;</pre>	
7.	<pre>create or replace procedure Work_Status_Insert (Work_Status_Name varchar(100)) language plpgsql as \$\$ begin insert into Work_Status (Work_Status_name) values (p_Work_Status_name); end; \$\$;</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

37

№ П П	Скрипт	Результат
8.	<pre>create or replace procedure Work_Status_Update (p_Work_Status_code int, p_Work_Status_name Varchar(100)) language plpgsql as \$\$ begin update Work_Status set Work_Status_name = p_Work_Status_name where Work_Status_code = p_Work_Status_code; end; \$\$;</pre>	
9.	<pre>create or replace procedure Work_Status_Delete (p_Work_Status_code int) language plpgsql as \$\$</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

38

№ П П	Скрипт	Результат
	<pre>begin delete from Work_Status where Work_Status_code = p_Work_Status_code; end; \$\$;</pre>	
10.	<pre>create or replace procedure Work_List_Plan_Insert (p_code_Work_List int, p_code_Work_Plan int) language plpgsql as \$\$ begin insert into Work_List_Plan (code_Work_List, code_Work_Plan) values (p_code_Work_List, p_code_Work_Plan); end; \$\$;</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

39

№ П П	Скрипт	Результат
11.	<pre>create or replace procedure Work_List_Plan_Update (p_work_list_plan_code int, p_code_Work_List int, p_code_Work_Plan int) language plpgsql as \$\$ begin update Work_List_Plan set code_work_List = p_code_Work_List, code_work_plan = p_code_Work_Plan where work_list_plan_code = p_work_list_plan_code; end; \$\$;</pre>	

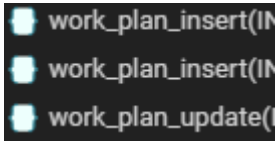
Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

40

№ П П	Скрипт	Результат
12.	<pre>create or replace procedure Work_List_Plan_Delete (p_work_list_plan_code int) language plpgsql as \$\$ begin delete from Work_List_Plan where work_list_plan_code = p_work_list_plan_code; end; \$\$;</pre>	
13.	<pre>create or replace procedure Work_Plan_Insert(p_Work_Plan_ Date date, p_Work_Plan_Start_Date date, p_Work_Plan_End_Date date, p_Work_Plan_Instruction varchar(100),</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

41

№ П П	Скрипт	Результат
	<pre>p_Work_Plan_Enclosure int, p_Work_Plan_Status int) language plpgsql as \$\$ declare work_plan_year varchar(4) := date_part('year', p_Work_Plan_Date); p_Work_plan_number varchar(20) := 'ГРМ- ' substring(work_plan_year, 3, 2); work_plan_counter integer := (select count(*) from work_plan) + 1; work_plan_zeros varchar(11) := LPAD(work_plan_counter::text, 10, '0'); begin p_Work_plan_number := p_Work_plan_number '-' work_plan_zeros;</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

42

№ П П	Скрипт	Результат
	<pre>insert into Work_Plan(Work_Plan_Number, Work_Plan_Date, Work_Plan_Start_Date, Work_Plan_End_Date, Work_Plan_Instruction, Work_Plan_Enclosure, Work_Plan_Status) values (p_Work_Plan_Number, p_Work_Plan_Date, p_Work_Plan_Start_Date, p_Work_Plan_End_Date, p_Work_Plan_Instruction, p_Work_Plan_Enclosure, p_Work_Plan_Status); end; \$\$;</pre>	
14.	<pre>create or replace procedure work_plan_Update (p_work_plan_code int, p_Work_Plan_Date date, p_Work_Plan_Start_Date date,</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

43

№ П П	Скрипт	Результат
	<pre>p_Work_Plan_End_Date date, p_Work_Plan_Instruction varchar(100), p_Work_Plan_Enclosure int, p_Work_Plan_Status int) language plpgsql as \$\$ begin update Work_Plan set Work_Plan_Date = p_Work_Plan_Date, Work_Plan_Start_Date = p_Work_Plan_Start_Date, Work_Plan_End_Date = p_Work_Plan_End_Date, Work_Plan_Instruction = p_Work_Plan_Instruction,</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

44

№ П П	Скрипт	Результат
	<pre> Work_Plan_Enclosure = p_Work_Plan_Enclosure, Work_Plan_Status = p_Work_Plan_Status where work_plan_code = p_work_plan_code; end; \$\$;</pre>	
15.	<pre>create or replace procedure work_plan_Delete (p_work_plan_code int) language plpgsql as \$\$ declare p_Exist_Work_Plans smallint := count(*) from Work_List_plan</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

45

№ П П	Скрипт	Результат
	<pre> where code_work_plan = p_work_plan_code; begin if(p_Exist_Works > 0) then raise notice 'Dанный план не может быть удален, т.к. он используется в списке работ'; else delete from work_plan where work_plan_code = p_work_plan_code; end if; end; \$\$;</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

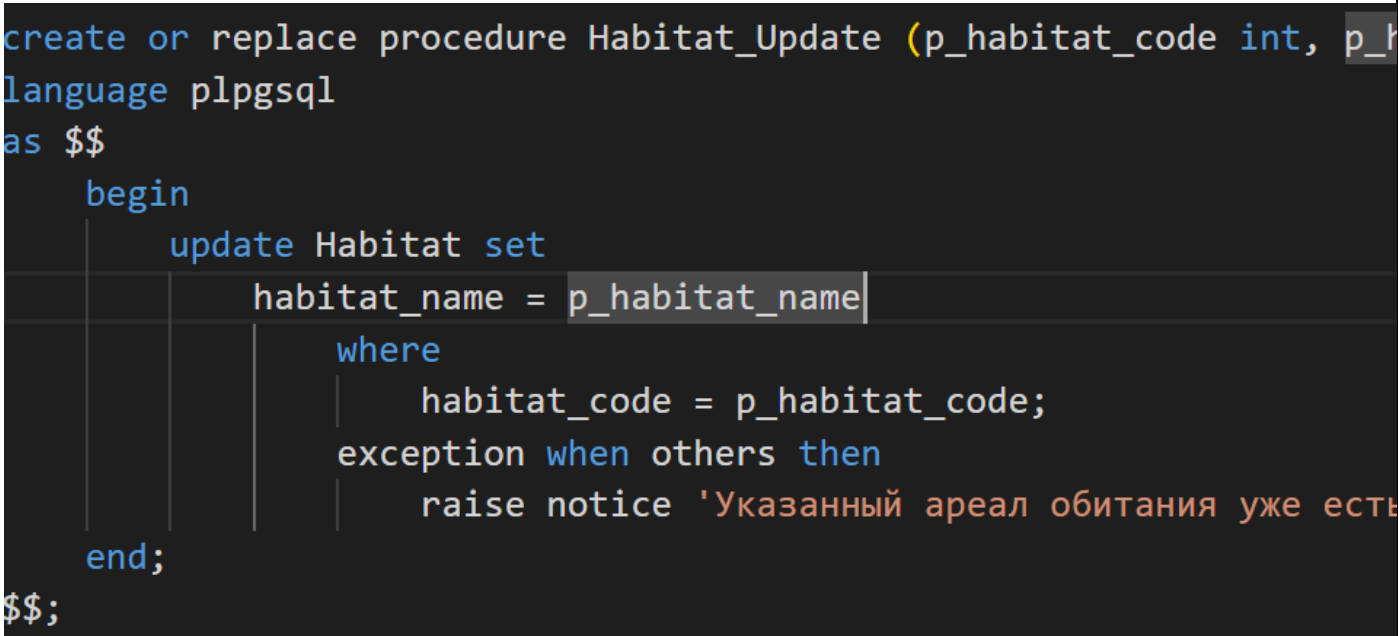
Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

46

№ П П	Скрипт	Результат
16.	<pre>create or replace procedure Habitat_Insert (p_habitat_name varchar(100)) language plpgsql as \$\$ begin insert into Habitat (habitat_name) values (p_habitat_name); exception when others then raise notice 'Указанный ареал обитания уже есть в таблице!'; end; \$\$;</pre>	<pre>create or replace procedure Habitat_Insert (p_habitat_name varchar(100)) language plpgsql as \$\$ begin insert into Habitat (habitat_name) values (p_habitat_name); exception when others then raise notice 'Указанный ареал обитания уже есть в таблице!'; end; \$\$;</pre>

№ П П	Скрипт	Результат
17.	<pre> create or replace procedure Habitat_Update (p_habitat_code int, p_habitat_name Varchar(100)) language plpgsql as \$\$ begin update Habitat set habitat_name = p_habitat_name where habitat_code = p_habitat_code; exception when others then raise notice 'Указанный ареал обитания уже есть в таблице!'; end; \$\$; </pre>	 <pre> create or replace procedure Habitat_Update (p_habitat_code int, p_h language plpgsql as \$\$ begin update Habitat set habitat_name = p_habitat_name where habitat_code = p_habitat_code; exception when others then raise notice 'Указанный ареал обитания уже есть end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

48

№ П П	Скрипт	Результат
18.	<pre>create or replace procedure Employee_Post_Insert (p_Employee_Post_Name varchar(100)) language plpgsql as \$\$ begin insert into Employee_Post (Employee_Post_Name) values (p_Employee_Post_Name); exception when others then raise notice 'Указанная должность сотрудника уже есть в таблице!'; end; \$\$;</pre>	<pre>create or replace procedure Employee_Post_Insert (p_Employee_Post_Name varchar(100)) language plpgsql as \$\$ begin insert into Employee_Post (Employee_Post_Name) values (p_Employee_Post_Name); exception when others then raise notice 'Указанная должность сотрудника уже есть в таблице!'; end; \$\$;</pre>

№ П П	Скрипт	Результат
19.	<pre> create or replace procedure Habitat_Delete (p_habitat_code int) language plpgsql as \$\$ declare p_Exist_animals smallint := count(*) from animal where habitat_code = p_habitat_code; begin if(p_Exist_animals > 0) then raise notice 'Dанный ареал обитания не может быть удален, т.к. он используется у животных'; else delete from Habitat where habitat_code = p_habitat_code; end if; end; </pre>	<pre> create or replace procedure Habitat_Delete (p_habitat_code int) language plpgsql as \$\$ declare p_Exist_animals smallint := count(*) from animal where habitat_code = p_habitat_code; begin if(p_Exist_animals > 0) then raise notice 'Данный ареал обитания не может быть удален, else delete from Habitat where habitat_code = p_habitat_code; end if; end; \$\$; </pre>

№ П П	Скрипт	Результат
	<pre> \$\$; call Habitat_Delete(1); </pre>	
20.	<pre> create or replace procedure Employee_Post_Delete (p_Employee_Post_Code int) language plpgsql as \$\$ declare p_Exist_employees smallint := count(*) from employee where employee_post_code = p_Employee_Post_Code; begin if(p_Exist_employees > 0) then raise notice 'Dанная должность сотрудника не может быть удалена, т.к. она используется в сотрудниках'; else </pre>	<pre> create or replace procedure Employee_Post_Delete (p_Employee_Post_Code language plpgsql as \$\$ declare p_Exist_employees smallint := count(*) from employee where employee_post_code = p_Employee_Post_Code; begin if(p_Exist_employees > 0) then raise notice 'Данная должность сотрудника не может быть уд else delete from Employee_Post where Employee_Post_Code = p_Employee_Post_Code; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

51

№ П П	Скрипт	Результат
	<pre>delete from Employee_Post where Employee_Post_Code = p_Employee_Post_Code; end if; end; \$\$;</pre>	
21.	<pre>create or replace procedure Visitor_Insert (p_Login_Visitor varchar(50), p_Name_Visitor varchar(50), p_Surname_Visitor varchar(50), p_Patronymic_Visitor varchar(50), p_Passport_Series_Visitor varchar(4), p_Passport_Number_Visitor varchar(6), p_Benefits_Visitor int, p_Password_Visitor varchar(36)) language plpgsql as \$\$ begin</pre>	<pre>create or replace procedure Visitor_Insert (p_Login_Visitor varchar(50) language plpgsql as \$\$ begin insert into Visitor (Login_Visitor, Name_Visitor, Surname_Visitor, Benefits_Visitor, Password_Visitor) values (p_Login_Visitor, p_Name_Visitor, p_Surname_Visitor, p_ p_Benefits_Visitor, p_Password_Visitor); exception when others then raise notice 'Указанный посетитель уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

52

№ П П	Скрипт	Результат
	<pre>insert into Visitor (Login_Visitor, Name_Visitor, Surname_Visitor, Patronymic_Visitor, Passport_Series_Visitor, Passport_Number_Visitor, Benefits_Visitor, Password_Visitor) values (p_Login_Visitor, p_Name_Visitor, p_Surname_Visitor, p_Patronymic_Visitor, p_Passport_Series_Visitor, p_Passport_Number_Visitor, p_Benefits_Visitor, p_Password_Visitor); exception when others then raise notice 'Указанный посетитель уже есть в таблице!';</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

53

№ П П	Скрипт	Результат
	end; \$\$;	

22.	<pre> create or replace procedure Visitor_Update (p_ID_Visitor int, p_Login_Visitor varchar(50), p_Name_Visitor varchar(50), p_Surname_Visitor varchar(50), p_Patronymic_Visitor varchar(50), p_Passport_Series_Visitor varchar(4), p_Passport_Number_Visitor varchar(6), p_Benefits_Visitor int, p_Password_Visitor varchar(36)) language plpgsql as \$\$ begin update Visitor set Login_Visitor = p_Login_Visitor, Name_Visitor = p_Name_Visitor, Surname_Visitor = p_Surname_Visitor, Patronymic_Visitor = p_Patronymic_Visitor, </pre>	<pre> create or replace procedure Visitor_Update (p_ID_Visitor int, p_Login language plpgsql as \$\$ begin update Visitor set Login_Visitor = p_Login_Visitor, Name_Visitor = p_Name_Visitor, Surname_Visitor = p_Surname_Visitor, Patronymic_Visitor = p_Patronymic_Visitor, Passport_Series_Visitor = p_Passport_Series_Visitor, Passport_Number_Visitor = p_Passport_Number_Visitor, Benefits_Visitor = p_Benefits_Visitor, Password_Visitor = p_Password_Visitor where ID_Visitor = p_ID_Visitor; exception when others then raise notice 'Указанный посетитель уже есть в таблице'; end; \$\$; </pre>
-----	---	--

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

55

№ П П	Скрипт	Результат
	<pre> Passport_Series_Visitor = p_Passport_Series_Visitor, Passport_Number_Visitor = p_Passport_Number_Visitor, Benefits_Visitor = p_Benefits_Visitor, Password_Visitor = p_Password_Visitor where ID_Visitor = p_ID_Visitor; exception when others then raise notice 'Указанный посетитель уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
23.	<pre> create or replace procedure Visitor_Delete (p_ID_Visitor int) language plpgsql as \$\$ declare p_Exist_visitors smallint := count(*) from ticket where id_visitor = p_ID_Visitor; begin if(p_Exist_visitors > 0) then raise notice 'Dанная посетитель не может быть удален, т.к. он используется в билетах'; else delete from Visitor where ID_Visitor = p_ID_Visitor; end if; end; \$\$; </pre>	<pre> create or replace procedure Visitor_Delete (p_ID_Visitor int) language plpgsql as \$\$ declare p_Exist_visitors smallint := count(*) from ticket where id_visitor = p_ID_Visitor; begin if(p_Exist_visitors > 0) then raise notice 'Данная посетитель не может быть удален, т.к. else delete from Visitor where ID_Visitor = p_ID_Visitor; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

57

№ П П	Скрипт	Результат
24.	<pre>create or replace procedure Visitor_Type_Insert (p_Name_Visitor_Type varchar(50)) language plpgsql as \$\$ begin insert into Visitor_Type (Name_Visitor_Type) values (p_Name_Visitor_Type); exception when others then raise notice 'Указанный вид посетителя уже есть в таблице!'; end; \$\$;</pre>	<pre>create or replace procedure Visitor_Type_Insert (p_Name_Visi language plpgsql as \$\$ begin insert into Visitor_Type (Name_Visitor_Type) values (p_Name_Visitor_Type); exception when others then raise notice 'Указанный вид посетителя уже е end; \$\$;</pre>

№ П П	Скрипт	Результат
25.	<pre> create or replace procedure Visitor_Type_Update (p_ID_Visitor_Type int, p_Name_Visitor_Type Varchar(50)) language plpgsql as \$\$ begin update Visitor_Type set Name_Visitor_Type = p_Name_Visitor_Type where ID_Visitor_Type = p_ID_Visitor_Type; exception when others then raise notice 'Указанный вид посетителя уже есть в таблице!'; end; \$\$; </pre>	<pre> create or replace procedure Visitor_Type_Update (p_ID_Visitor_Type int language plpgsql as \$\$ begin update Visitor_Type set Name_Visitor_Type = p_Name_Visitor_Type where ID_Visitor_Type = p_ID_Visitor_Type; exception when others then raise notice 'Указанный вид посетителя уже есть в табл end; \$\$; </pre>

№ П П	Скрипт	Результат
26.	<pre> create or replace procedure Visitor_Type_Delete (p_ID_Visitor_Type int) language plpgsql as \$\$ declare p_Exist_type_visitors smallint := count(*) from Visitor_Document where id_visitor = p_ID_Visitor_Type; begin if(p_Exist_type_visitors > 0) then raise notice 'Dанная вид посетителя не может быть удален, т.к. он используется в документах'; else delete from Visitor_Type where </pre>	<pre> create or replace procedure Visitor_Type_Delete (p_ID_Visitor_Type int language plpgsql as \$\$ declare p_Exist_type_visitors smallint := count(*) from Visitor_Docume where id_visitor = p_ID_Visitor_Type; begin if(p_Exist_type_visitors > 0) then raise notice 'Данная вид посетителя не может быть удален, else delete from Visitor_Type where ID_Visitor_Type = p_ID_Visitor_Type; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

60

№ П П	Скрипт	Результат
	<pre> ID_Visitor_Type = p_ID_Visitor_Type; end if; end; \$\$;</pre>	
27.	<pre>create or replace procedure Visitor_Document_Insert (p_Number_Document varchar(100), p_ID_Visitor_Type int, p_ID_Visitor int) language plpgsql as \$\$ begin insert into Visitor_Document (Number_Document, ID_Visitor_Type, ID_Visitor) values (p_Number_Document, p_ID_Visitor_Type, p_ID_Visitor); exception when others then</pre>	<pre>create or replace procedure Visitor_Document_Insert (p_Number_Document language plpgsql as \$\$ begin insert into Visitor_Document (Number_Document, ID_Visitor_Type values (p_Number_Document, p_ID_Visitor_Type, p_ID_Visitor); exception when others then raise notice 'Указанный документ уже есть в таблице!'; end; \$\$;</pre>

№ П П	Скрипт	Результат
	<pre> raise notice 'Указанный документ уже есть в таблице!'; end; \$\$;</pre>	
28.	<pre> create or replace procedure Visitor_Document_Update (p_ID_Document int, p_Number_Document varchar(100), p_ID_Visitor_Type int, p_ID_Visitor int) language plpgsql as \$\$ begin update Visitor_Document set Number_Document = p_Number_Document, ID_Visitor_Type = p_ID_Visitor_Type, ID_Visitor = p_ID_Visitor</pre>	<pre> create or replace procedure Visitor_Document_Update (p_ID_Document in language plpgsql as \$\$ begin update Visitor_Document set Number_Document = p_Number_Document, ID_Visitor_Type = p_ID_Visitor_Type, ID_Visitor = p_ID_Visitor where ID_Document = p_ID_Document; exception when others then raise notice 'Указанный документ уже есть в таблице!' end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

62

№ П П	Скрипт	Результат
	<pre>where ID_Document = p_ID_Document; exception when others then raise notice 'Указанный документ уже есть в таблице!'; end; \$;\$;</pre>	

№ П П	Скрипт	Результат
29.	<pre> create or replace procedure Enclosure_Insert (p_Name_Enclosure varchar(50), p_Status_Enclosure varchar(16)) language plpgsql as \$\$ begin insert into Enclosure (Name_Enclosure, Status_Enclosure) values (p_Name_Enclosure, p_Status_Enclosure); exception when others then raise notice 'Указанный вольер уже есть в таблице!'; end; \$\$;</pre>	<pre> create or replace procedure Enclosure_Insert (p_Name_Enclosure varchar(50), p_Status_Enclosure varchar(16)) language plpgsql as \$\$ begin insert into Enclosure (Name_Enclosure, Status_Enclosure) values (p_Name_Enclosure, p_Status_Enclosure); exception when others then raise notice 'Указанный вольер уже есть в таблице!'; end; \$\$;</pre>

№ П П	Скрипт	Результат
30.	<pre> create or replace procedure Enclosure_Update (p_ID_Enclosure int, p_Name_Enclosure varchar(50), p_Status_Enclosure varchar(16)) language plpgsql as \$\$ begin update Enclosure set Name_Enclosure = p_Name_Enclosure, Status_Enclosure = p_Status_Enclosure where ID_Enclosure = p_ID_Enclosure; exception when others then </pre>	<pre> create or replace procedure Enclosure_Update (p_ID_Enclosure int, p_Na language plpgsql as \$\$ begin update Enclosure set Name_Enclosure = p_Name_Enclosure, Status_Enclosure = p_Status_Enclosure where ID_Enclosure = p_ID_Enclosure; exception when others then raise notice 'Указанный вольтер уже есть в таблице!'; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

65

№ П П	Скрипт	Результат
	<pre>raise notice 'Указанный вольер уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
31.	<pre> create or replace procedure Enclosure_Delete (p_ID_Enclosure int) language plpgsql as \$\$ declare p_Exist_Enclosures smallint := count(*) from employee_enclosure where id_enclosure = p_ID_Enclosure; begin if(p_Exist_Enclosures > 0) then raise notice 'Dанная вольер не может быть удален, т.к. он используется в сотрудниках-вольерах'; else delete from Enclosure where ID_Enclosure = p_ID_Enclosure; </pre>	<pre> create or replace procedure Enclosure_Delete (p_ID_Enclosure int) language plpgsql as \$\$ declare p_Exist_Enclosures smallint := count(*) from employee_enclosure where id_enclosure = p_ID_Enclosure; begin if(p_Exist_Enclosures > 0) then raise notice 'Данная вольер не может быть удален, т.к. он и else delete from Enclosure where ID_Enclosure = p_ID_Enclosure; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

67

№ П П	Скрипт	Результат
	end if; end; \$\$;	
32.	create or replace procedure Employee_Insert (p_Login_Employee varchar(100), p_Surname_Employee varchar(50), p_Name_Employee varchar(50), p_Patronymic_Employee varchar(50), p_Password_Employee varchar(36), p_Employee_Post_Code int) language plpgsql as \$\$ begin insert into Employee (Login_Employee, Surname_Employee, Name_Employee, Patronymic_Employee,	<pre>create or replace procedure Employee_Insert (p_Login_Employee varchar p_Patronymic_Employee var language plpgsql as \$\$ begin insert into Employee (Login_Employee, Surname_Employee, Name_ values (p_Login_Employee, p_Surname_Employee, p_Name_Employee exception when others then raise notice 'Указанный сотрудник уже есть в таблице! end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

68

№ П П	Скрипт	Результат
	<pre>Password_Employee, Employee_Post_Code) values (p_Login_Employee, p_Surname_Employee, p_Name_Employee, p_Patronymic_Employee, p_Password_Employee, p_Employee_Post_Code); exception when others then raise notice 'Указанный сотрудник уже есть в таблице!'; end; \$\$;</pre>	

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

69

№ П П	Скрипт	Результат
33.	<pre>create or replace procedure Employee_Update (p_ID_Employee int, p_Login_Employee varchar(100), p_Surname_Employee varchar(50), p_Name_Employee varchar(50), p_Patronymic_Employee varchar(50), p_Password_Employee varchar(36), p_Employee_Post_Code int) language plpgsql as \$\$ begin update Employee set Login_Employee = p_Login_Employee,</pre>	<pre>create or replace procedure Employee_Update (p_ID_Employee int, p_Log p_Patronymic_Employee var language plpgsql as \$\$ begin update Employee set Login_Employee = p_Login_Employee, Surname_Employee = p_Surname_Employee, Name_Employee = p_Name_Employee, Patronymic_Employee = p_Patronymic_Employee, Password_Employee = p_Password_Employee, Employee_Post_Code = p_Employee_Post_Code where ID_Employee = p_ID_Employee; exception when others then raise notice 'Указанный сотрудник уже есть в таблице! end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

70

№ П П	Скрипт	Результат
	<pre>Surname_Employee = p_Surname_Employee, Name_Employee = p_Name_Employee, Patronymic_Employee = p_Patronymic_Employee, Password_Employee = p_Password_Employee, Employee_Post_Code = p_Employee_Post_Code where ID_Employee = p_ID_Employee; exception when others then raise notice 'Указанный сотрудник уже есть в таблице!';</pre>	

№ П П	Скрипт	Результат
	end; \$\$;	
34.	<pre> create or replace procedure Employee_Delete (p_ID_Employee int) language plpgsql as \$\$ declare p_Exist_employees smallint := count(*) from employee_enclosure where id_employee = p_ID_Employee; begin if(p_Exist_employees > 0) then raise notice 'Dанная сотрудник не может быть удален, т.к. он используется в сотрудниках- вольерах'; else delete from Employee where </pre>	<pre> create or replace procedure Employee_Delete (p_ID_Employee int) language plpgsql as \$\$ declare p_Exist_employees smallint := count(*) from employee_enclosure where id_employee = p_ID_Employee; begin if(p_Exist_employees > 0) then raise notice 'Данная сотрудник не может быть удален, т.к. с else delete from Employee where ID_Employee = p_ID_Employee; end if; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

72

№ П П	Скрипт	Результат
	<pre> ID_Employee = p_ID_Employee; end if; end; \$\$;</pre>	
35.	<pre>create or replace procedure Animal_Squad_Insert (p_Name_Animal_Squad varchar(100)) language plpgsql as \$\$ begin insert into Animal_Squad (Name_Animal_Squad) values (p_Name_Animal_Squad); exception when others then raise notice 'Указанный отряд уже есть в таблице!'; end; \$\$;</pre>	<pre>create or replace procedure Animal_Squad_Insert (p_Name_Animal_Sc language plpgsql as \$\$ begin insert into Animal_Squad (Name_Animal_Squad) values (p_Name_Animal_Squad); exception when others then raise notice 'Указанный отряд уже есть в таблице' end; \$\$;</pre>

№ П П	Скрипт	Результат
36.	<pre> create or replace procedure Animal_Squad_Update (p_ID_Animal_Squad int, p_Name_Animal_Squad Varchar(100)) language plpgsql as \$\$ begin update Animal_Squad set Name_Animal_Squad = p_Name_Animal_Squad where ID_Animal_Squad = p_ID_Animal_Squad; exception when others then raise notice 'Указанный отряд уже есть в таблице!'; end; \$\$; </pre>	<pre> create or replace procedure Animal_Squad_Update (p_ID_Animal_Squad int language plpgsql as \$\$ begin update Animal_Squad set Name_Animal_Squad = p_Name_Animal_Squad where ID_Animal_Squad = p_ID_Animal_Squad; exception when others then raise notice 'Указанный отряд уже есть в таблице!'; end; \$\$; </pre>

№ П П	Скрипт	Результат
37.	<pre> create or replace procedure Animal_Squad_Delete (p_ID_Animal_Squad int) language plpgsql as \$\$ declare p_Exist_squads smallint := count(*) from Animal_Family where id_animal_squad = p_ID_Animal_Squad; begin if(p_Exist_squads > 0) then raise notice 'Dанный отряд не может быть удален, т.к. он используется в семействах'; else delete from Animal_Squad where ID_Animal_Squad = p_ID_Animal_Squad; </pre>	<pre> create or replace procedure Animal_Squad_Delete (p_ID_Animal_Squad int language plpgsql as \$\$ declare p_Exist_squads smallint := count(*) from Animal_Family where id_animal_squad = p_ID_Animal_Squad; begin if(p_Exist_squads > 0) then raise notice 'Данный отряд не может быть удален, т.к. он и else delete from Animal_Squad where ID_Animal_Squad = p_ID_Animal_Squad; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

75

№ П П	Скрипт	Результат
	<pre>end if; end; \$;\$</pre>	
38.	<pre>create or replace procedure Animal_Family_Insert (p_Name_Animal_Family varchar(100), p_ID_Animal_Squad int) language plpgsql as \$\$ begin insert into Animal_Family (Name_Animal_Family, ID_Animal_Squad) values (p_Name_Animal_Family, p_ID_Animal_Squad); exception when others then raise notice 'Указанное семейство уже есть в таблице!'; end; \$;\$</pre>	<pre>create or replace procedure Animal_Family_Insert (p_Name_Animal_Family language plpgsql as \$\$ begin insert into Animal_Family (Name_Animal_Family, ID_Animal_Squad) values (p_Name_Animal_Family, p_ID_Animal_Squad); exception when others then raise notice 'Указанное семейство уже есть в таблице!'; end; \$;\$</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

76

№ П П	Скрипт	Результат
39.	<pre>create or replace procedure Animal_Family_Update (p_ID_Animal_Family int, p_Name_Animal_Family varchar(100), p_ID_Animal_Squad int) language plpgsql as \$\$ begin update Animal_Family set Name_Animal_Family = p_Name_Animal_Family, ID_Animal_Squad = p_ID_Animal_Squad where ID_Animal_Family = p_ID_Animal_Family; exception when others then</pre>	<pre>create or replace procedure Animal_Family_Update (p_ID_Animal_Family in language plpgsql as \$\$ begin update Animal_Family set Name_Animal_Family = p_Name_Animal_Family, ID_Animal_Squad = p_ID_Animal_Squad where ID_Animal_Family = p_ID_Animal_Family; exception when others then raise notice 'Указанное семейство уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

77

№ П П	Скрипт	Результат
	<pre>raise notice 'Указанное семейство уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
40.	<pre> create or replace procedure Animal_Family_Delete (p_ID_Animal_Family int) language plpgsql as \$\$ declare p_Exist_families smallint := count(*) from animal_type where id_animal_family = p_ID_Animal_Family; begin if(p_Exist_families > 0) then raise notice 'Dанное семейство не может быть удалено, т.к. оно используется в видах'; else delete from Animal_Family where </pre>	<pre> create or replace procedure Animal_Family_Delete (p_ID_Animal_Family int) language plpgsql as \$\$ declare p_Exist_families smallint := count(*) from animal_type where id_animal_family = p_ID_Animal_Family; begin if(p_Exist_families > 0) then raise notice 'Данное семейство не может быть удалено, т.к. else delete from Animal_Family where ID_Animal_Family = p_ID_Animal_Family; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

79

№ П П	Скрипт	Результат
	<pre> ID_Animal_Family = p_ID_Animal_Family; end if; end; \$\$;</pre>	
41.	<pre>create or replace procedure Animal_Type_Insert (p_Name_Animal_Type varchar(100), p_ID_Animal_Family int) language plpgsql as \$\$ begin insert into Animal_Type (Name_Animal_Type, ID_Animal_Family) values (p_Name_Animal_Type, p_ID_Animal_Family); exception when others then</pre>	<pre>create or replace procedure Animal_Type_Insert (p_Name_Animal_Type varchar(100), language plpgsql as \$\$ begin insert into Animal_Type (Name_Animal_Type, ID_Animal_Family) values (p_Name_Animal_Type, p_ID_Animal_Family); exception when others then raise notice 'Указанный вид уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

80

№ П П	Скрипт	Результат
	<pre>raise notice 'Указанный вид уже есть в таблице!'; end; \$\$;</pre>	
42.	<pre>create or replace procedure Animal_Type_Update (p_ID_Animal_Type int, p_Name_Animal_Type varchar(100), p_ID_Animal_Family int) language plpgsql as \$\$ begin update Animal_Type set Name_Animal_Type = p_Name_Animal_Type, ID_Animal_Family = p_ID_Animal_Family where</pre>	<pre>create or replace procedure Animal_Type_Update (p_ID_Animal_Type int, p language plpgsql as \$\$ begin update Animal_Type set Name_Animal_Type = p_Name_Animal_Type, ID_Animal_Family = p_ID_Animal_Family where ID_Animal_Type = p_ID_Animal_Type; exception when others then raise notice 'Указанный вид уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

81

№ П П	Скрипт	Результат
	<pre> ID_Animal_Type = p_ID_Animal_Type; exception when others then raise notice 'Указанный вид уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
43.	<pre> create or replace procedure Animal_Type_Delete (p_ID_Animal_Type int) language plpgsql as \$\$ declare p_Exist_types smallint := count(*) from animal_type where id_animal_family = p_ID_Animal_Family; begin if(p_Exist_types > 0) then raise notice 'Dанное семейство не может быть удалено, т.к. оно используется в видах'; else delete from Animal_Type where ID_Animal_Type = p_ID_Animal_Type; </pre>	<pre> create or replace procedure Animal_Type_Delete (p_ID_Animal_Type int) language plpgsql as \$\$ declare p_Exist_types smallint := count(*) from animal_type where id_animal_family = p_ID_Animal_Family; begin if(p_Exist_types > 0) then raise notice 'Данное семейство не может быть удалено, т.к. else delete from Animal_Type where ID_Animal_Type = p_ID_Animal_Type; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

83

№ П П	Скрипт	Результат
	end if; end; \$\$;	
44.	<pre>create or replace procedure Territory_Update (p_ID_Territory int, p_Name_Territory varchar(100), p_Price_Territory decimal(2, 5)) language plpgsql as \$\$ begin update Territory set Name_Territory = p_Name_Territory, Price_Territory = p_Price_Territory where ID_Territory = p_ID_Territory; exception when others then</pre>	<pre>create or replace procedure Territory_Update (p_ID_Territory int, p_Na language plpgsql as \$\$ begin update Territory set Name_Territory = p_Name_Territory, Price_Territory = p_Price_Territory where ID_Territory = p_ID_Territory; exception when others then raise notice 'Указанная территория уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

84

№ П П	Скрипт	Результат
	<pre>raise notice 'Указанная территория уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
45.	<pre> create or replace procedure Territory_Delete (p_ID_Territory int) language plpgsql as \$\$ declare p_Exist_territories smallint := count(*) from animal where id_territory = p_ID_Territory; begin if(p_Exist_territories > 0) then raise notice 'Dанная территория не может быть удалена, т.к. она используется в животных'; else delete from Territory where ID_Territory = p_ID_Territory; end if; </pre>	<pre> create or replace procedure Territory_Delete (p_ID_Territory int) language plpgsql as \$\$ declare p_Exist_territories smallint := count(*) from animal where id_territory = p_ID_Territory; begin if(p_Exist_territories > 0) then raise notice 'Данная территория не может быть удалена, т.к. else delete from Territory where ID_Territory = p_ID_Territory; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

86

№ П П	Скрипт	Результат
	end; \$\$;	
46.	<pre>create or replace procedure Animal_Insert (p_Number_Animal varchar(11), p_Description_Animal varchar(100), p_Picture_Animal varchar(100), p_ID_Animal_Type int, p_Habitat_Code int, p_ID_Enclosure int, p_ID_Territory int) language plpgsql as \$\$ begin insert into Animal (Number_Animal, Description_Animal, Picture_Animal, ID_Animal_Type, Habitat_Code, ID_Enclosure, ID_Territory) values (p_Number_Animal, p_Description_Animal, p_Picture_Animal, p_ID_Animal_Type,</pre>	<pre>create or replace procedure Animal_Insert (p_Number_Animal varchar(11) language plpgsql as \$\$ begin insert into Animal (Number_Animal, Description_Animal, Picture values (p_Number_Animal, p_Description_Animal, p_Picture_Animal exception when others then raise notice 'Указанное животное уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

87

№ П П	Скрипт	Результат
	<pre>p_Habitat_Code, p_ID_Enclosure, p_ID_Territory); exception when others then raise notice 'Указанное животное уже есть в таблице!'; end; \$\$;</pre>	
47.	<pre>create or replace procedure Ticket_Insert (p_Number_Ticket varchar(16), p_Datetime_Ticket timestamp, p_Price_Ticket decimal(6, 2), p_Total_Sum_Ticket decimal(6, 2), p_ID_Visitor int) language plpgsql as \$\$ begin insert into Ticket (Number_Ticket, Datetime_Ticket, Price_Ticket, Total_Sum_Ticket, ID_Visitor) values (p_Number_Ticket,</pre>	<pre>create or replace procedure Ticket_Insert (p_Number_Ticket varchar(16), p_Dat language plpgsql as \$\$ begin insert into Ticket (Number_Ticket, Datetime_Ticket, Price_Ticket, Tot values (p_Number_Ticket, p_Datetime_Ticket, p_Price_Ticket, p_Total_S exception when others then raise notice 'Указанный билет уже есть в таблице!'; end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

88

№ П П	Скрипт	Результат
	<pre>p_Datetime_Ticket, p_Price_Ticket, p_Total_SUm_Ticket, p_ID_Visitor); exception when others then raise notice 'Указанный билет уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
48.	<pre> create or replace procedure Ticket_Update (p_ID_Ticket int, p_Number_Ticket varchar(15), p_Datetime_Ticket timestamp, p_Price_Ticket decimal(6,2), p_Total_Sum_Ticket decimal(6, 2), p_ID_Visitor int) language plpgsql as \$\$ begin update Ticket set Number_Ticket = p_Number_Ticket, Datetime_Ticket = p_Datetime_Ticket, Price_Ticket = p_Price_Ticket, Total_SUm_Ticket = p_Total_SUm_Ticket, ID_Visitor = p_ID_Visitor </pre>	<pre> create or replace procedure Ticket_Update (p_ID_Ticket int, p_Number_Ticket va language plpgsql as \$\$ begin update Ticket set Number_Ticket = p_Number_Ticket, Datetime_Ticket = p_Datetime_Ticket, Price_Ticket = p_Price_Ticket, Total_SUm_Ticket = p_Total_SUm_Ticket, ID_Visitor = p_ID_Visitor where ID_Ticket = p_ID_Ticket; exception when others then raise notice 'Указанный билет уже есть в таблице!'; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

90

№ П П	Скрипт	Результат
	<pre>where ID_Ticket = p_ID_Ticket; exception when others then raise notice 'Указанный билет уже есть в таблице!'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
49.	<pre> create or replace procedure Ticket_Delete (p_ID_Ticket int) language plpgsql as \$\$ declare p_Exist_tickets smallint := count(*) from territory_ticket where id_ticket = p_ID_Ticket; begin if(p_Exist_tickets > 0) then raise notice 'Dанный билет не может быть удален, т.к. он используется в территориях-билетах'; else delete from Ticket where ID_Ticket = p_ID_Ticket; end if; end; \$\$; </pre>	<pre> create or replace procedure Ticket_Delete (p_ID_Ticket int) language plpgsql as \$\$ declare p_Exist_tickets smallint := count(*) from territory_ticket where id_ticket = p_ID_Ticket; begin if(p_Exist_tickets > 0) then raise notice 'Данный билет не может быть удален, т.к. он использ else delete from Ticket where ID_Ticket = p_ID_Ticket; end if; end; \$\$; </pre>

№ П П	Скрипт	Результат
50.	<pre> create or replace procedure Enclosure_Care_Day_Insert (p_ID_Employee_Enclosure int, p_Enclosure_Care_Day varchar(11)) language plpgsql as \$\$ begin insert into Enclosure_Care_Day (ID_Employee_Enclosure,Enclos ure_Care_Day) values (p_ID_Employee_Enclosure, p_Enclosure_Care_Day); exception when others then raise notice 'Указанный день недели ухода не существует. Используйте один из дней недели'; end; \$\$;</pre>	<pre> create or replace procedure Enclosure_Care_Day_Insert (p_ID_Employee_Enclosur language plpgsql as \$\$ begin insert into Enclosure_Care_Day (ID_Employee_Enclosure,Enclosure_Care_ values (p_ID_Employee_Enclosure, p_Enclosure_Care_Day); exception when others then raise notice 'Указанный день недели ухода не существует. Испо end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

93

№ П П	Скрипт	Результат
51.	<pre>create or replace procedure Enclosure_Care_Day_Update (p_ID_Enclosure_Care_Day int, p_ID_Employee_Enclosure int, p_Enclosure_Care_Day varchar(11)) language plpgsql as \$\$ begin update Enclosure_Care_Day set ID_Employee_Enclosure = p_ID_Employee_Enclosure, Enclosure_Care_Day = p_Enclosure_Care_Day where ID_Enclosure_Care_Day = p_ID_Enclosure_Care_Day; exception when others then</pre>	<pre>create or replace procedure Enclosure_Care_Day_Update (p_ID_Enclosure_Care_D language plpgsql as \$\$ begin update Enclosure_Care_Day set ID_Employee_Enclosure = p_ID_Employee_Enclosure, Enclosure_Care_Day = p_Enclosure_Care_Day where ID_Enclosure_Care_Day = p_ID_Enclosure_Care_Day; exception when others then raise notice 'Указанный день недели ухода не существует. Исп end; \$\$;</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

94

№ П П	Скрипт	Результат
	<pre>raise notice 'Указанный день недели ухода не существует. Используйте один из дней недели'; end; \$\$;</pre>	

№ П П	Скрипт	Результат
52.	<pre> create or replace procedure Enclosure_Care_Day_Delete (p_ID_Enclosure_Care_Day int) language plpgsql as \$\$ declare p_Exist_care_days smallint := count(*) from Care_Time where id_enclosure_care_day = p_ID_Enclosure_Care_Day; begin if(p_Exist_care_days > 0) then raise notice 'Dанный день ухода не может быть удален, т.к. он используется во времени ухода'; else delete from Enclosure_Care_Day where </pre>	<pre> create or replace procedure Enclosure_Care_Day_Delete (p_ID_Enclosure_Care_Da language plpgsql as \$\$ declare p_Exist_care_days smallint := count(*) from Care_Time where id_enclosure_care_day = p_ID_Enclosure_Care_Day; begin if(p_Exist_care_days > 0) then raise notice 'Данный день ухода не может быть удален, т.к. он ис else delete from Enclosure_Care_Day where ID_Enclosure_Care_Day = p_ID_Enclosure_Care_Day; end if; end; \$\$; </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

96

№ П П	Скрипт	Результат
	<pre>ID_Enclosure_Care_Day = p_ID_Enclosure_Care_Day; end if; end; \$\$;</pre>	

3. Распределение прав доступа к таблицам и хранимым процедурам БД;

Таблица 5 – Права доступа к таблицам и хранимым процедурам БД

Роли		Посетитель	Сотрудник	Администратор
Название объекта	Функции			
Enclosure_Status	Выборка		X	X
	Добавление			X
	Изменение			X
	Удаление			
Work_List	Выборка		X	X
	Добавление			X
	Изменение			X
	Удаление			
Work_Status	Выборка		X	X
	Добавление			X
	Изменение			X
	Удаление			
Work_Plan	Выборка		X	X
	Добавление			X

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

97

Роли		Посетитель	Сотрудник	Администратор
Название объекта	Функции			
Work_List_Plan	Изменение			X
	Удаление			
	Выборка			X
	Добавление			X
	Изменение			X
	Удаление			
Enclosure_Status_Insert	Вызов			X
Enclosure_Status_Update	Вызов			X
Enclosure_Status_Delete	Вызов			
Work_List_Insert	Вызов			X
Work_List_Update	Вызов			X
Work_List_Delete	Вызов			
Work_Status_Update	Вызов			X
Work_Status_Insert	Вызов			X
Work_Status_Delete	Вызов			
Work_Plan_Update	Вызов			X
Work_Plan_Insert	Вызов			X
Work_Plan_Delete	Вызов			
Work_List_Plan_Insert	Вызов			X
Work_List_Plan_Delete	Вызов			
Work_List_Plan_Update	Вызов			X

4. Выдача прав доступа к таблицам и хранимым процедурам БД;

Таблица 6 – Реализация разграничения прав доступа

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

98

Название роли	Название объекта	Функция	Скрипт
zoo_visitor	-	-	-
zoo_employee	Enclosure_Status	Select	grant select on Enclosure_Status to zoo_employee;
	Work_List		grant select on Work_List to zoo_employee;
	Work_Status		grant select on Work_Status to zoo_employee;
	Work_Plan		grant select on Work_Plan to zoo_employee;
	Work_List_Plan		grant select on Work_List_Plan to zoo_employee;
zoo_administrator	Enclosure_Status	Select, Insert, Update	grant select, insert, update on Enclosure_Status to zoo_administrator; grant usage, select on sequence enclosure_status_enclosure_status_code_seq to zoo_administrator;
	Work_List		grant select, insert, update on Work_List to zoo_administrator; grant usage, select on sequence work_list_work_list_code_seq to zoo_administrator;
	Work_Status		grant select, insert, update on Work_Status to zoo_administrator; grant usage, select on sequence work_status_work_status_code_seq to zoo_administrator;
	Work_Plan		grant select, insert, update on Work_Plan to zoo_administrator; grant usage, select on sequence work_plan_work_plan_code_seq to zoo_administrator;
	Work_List_Plan		grant select, insert, update on Work_List_Plan to zoo_administrator; grant usage, select on sequence work_list_plan_work_list_plan_code_seq to zoo_administrator;

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

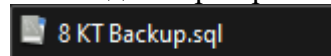
Группа: 2ПЗ.22

Студент: Новгородов Иван

99

Название роли	Название объекта	Функция	Скрипт
	Enclosure_Status_Insert	Execute	
	Enclosure_Status_Update		
	Work_List_Insert		
	Work_List_Update		
	Work_Status_Update		
	Work_Status_Insert		
	Work_Plan_Update		
	Work_Plan_Insert		
	Work_List_Plan_Insert		
	Work_List_Plan_Update		

5. Создание резервной копии БД;



6. Версия базы данных:

6.1. Отчёт о созданных объектах;

Таблица 7 – Перечень созданных объектов

	Информация по объектам
Запрос	<pre>select information_schema.tables.table_name as "Название таблиц", string_agg(distinct information_schema.columns.column_name, ', ') as "Столбцы", string_agg(distinct pg_indexes.indexname, ', ') as "Индексы", string_agg(distinct information_schema.routines.routine_name, ', ') as "Список процедур", n_live_tup as "Кол-во записей в таблицах" from information_schema.tables inner join information_schema.columns on information_schema.columns.table_name = information_schema.tables.table_name inner join pg_indexes</pre>

	Информация по объектам
	<pre> on information_schema.tables.table_name = pg_indexes.tablename inner join information_schema.routines on information_schema.tables.table_name = substring(information_schema.routines.routine_name, 1, length(information_schema.tables.table_name)) inner join pg_stat_user_tables on information_schema.tables.table_name = pg_stat_user_tables.relname where information_schema.tables.table_schema = 'public' and routine_type = 'PROCEDURE' and information_schema.tables.table_schema = 'public' and indexname not like 'pk_%' group by information_schema.tables.table_name, n_live_tup union all select (select count(*)::text from information_schema.tables where table_schema = 'public'), (select count(information_schema.columns.column_name)::text from information_schema.tables inner join information_schema.columns on information_schema.columns.table_name = information_schema.tables.table_name where </pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

101

	Информация по объектам
	<pre>information_schema.tables.table_schema = 'public'), (select count(pg_indexes.indexname)::text from information_schema.tables inner join pg_indexes on information_schema.tables.table_name = pg_indexes.tablename where information_schema.tables.table_schema = 'public' and indexname not like 'pk_%'), (select count(information_schema.routines.routine_name)::text from information_schema.routines where routine_type = 'PROCEDURE' and routine_name not in ('structure_create','structure_re_create')), (select sum(n_live_tup) from pg_stat_user_tables);</pre>

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

102

Информация по объектам				
Результат локального БД		Название таблиц name	Столбцы text	Индексы text
	1	animal	description_animal, habitat_code, id_animal, id_animal_type, id_enclosure, id_territory, number_animal, picture_animal	index_id_animal, index_number_animal, uq_n
	2	animal_family	id_animal_family, id_animal_squad, name_animal_family	index_id_animal_family, index_name_animal_
	3	animal_squad	id_animal_squad, name_animal_squad	index_id_animal_squad, index_name_animal_
	4	animal_type	id_animal_family, id_animal_type, name_animal_type	index_id_animal_type, index_name_animal_ty
	5	care_time	care_time, id_care_time, id_enclosure_care_day	index_id_caretime
	6	employee	employee_post_code, id_employee, login_employee, name_employee, password_employee, patronymic_employee, surname_employee	index_employee_login_password, index_empl
	7	employee_enclosure	id_employee, id_employee_enclosure, id_enclosure	index_id_employee_enclosure
	8	employee_post	employee_post_code, employee_post_name	index_employee_post_code, index_employee.
	9	enclosure	id_enclosure, name_enclosure, status_enclosure	index_id_enclosure, index_name_enclosure, u
	10	enclosure_care_day	enclosure_care_day, id_employee_enclosure, id_enclosure_care_day	index_id_enclosure_care_day
	11	enclosure_status	enclosure_status_code, enclosure_status_name	index_enclosure_status_code, index_enclosu
	12	habitat	habitat_code, habitat_name	index_habitat_code, index_habitat_name, uq_
	13	territory	id_territory, name_territory, price_territory	index_id_territory, index_name_territory, uq_n
	14	territory_ticket	id_territory, id_territory_ticket, id_ticket	index_id_territory_ticket
	15	ticket	datetime_ticket, id_ticket, id_visitor, number_ticket, price_ticket, total_sum_ticket	index_id_ticket, index_number_ticket
	16	visitor	benefits_visitor, id_visitor, login_visitor, name_visitor, passport_number_visitor, passport_series_visitor, password_visitor, patronymic_visitor, surname_v...	index_id_visitor, index_visitor_login_password
	17	visitor_document	id_document, id_visitor, id_visitor_type, number_document	index_id_document, index_number_document
	18	visitor_type	id_visitor_type, name_visitor_type	index_id_visitor_type, index_name_visitor_typ
	19	work_list	work_list_code, work_list_interval, work_list_name	index_work_list_code, index_work_list_name,
	20	work_list_plan	code_work_list, code_work_plan, work_list_plan_code	uq_code_work_list
	21	work_plan	work_plan_code, work_plan_date, work_plan_enclosure, work_plan_end_date, work_plan_instruction, work_plan_number, work_plan_start_date, work_pl...	index_work_plan_code, index_work_plan_num
	22	work_status	work_status_code, work_status_name	index_work_status_code, index_work_status_
	23	22	84	58
Результат удалённого БД	Сервер не доступен			

Дисциплина: СУБД (PostgreSQL, MySQL)

Контрольная точка: №8

Группа: 2ПЗ.22

Студент: Новгородов Иван

103

6.2. Версия БД.

Таблица 8 – Версия файла БД

Параметры	PostgreSQL
Номер версии	3.1.0.1
Что сделано	<ul style="list-style-type: none">- Созданы 5 таблиц;- Созданы 18 столбцов;- Созданы 9 индексов;- Созданы 15 хранимые процедуры;- Добавлено 10 строк во всю структуру БД;- Произведено распределение доступа ролей к таблицам и хранимым процедурам;- Создан Backup файл.