ЭТАПЫ ВЫПОЛНЕНИЯ

1. Сопровождение таблиц:

1.1. Создание файла;

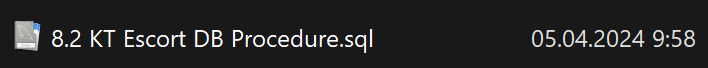
1.2. Сопровождение объектов.

Таблица 1 – Сопровождение таблиц баз данных

| Таблица | Поле | Тип данных | Обязательное | Ограничения | Было | Стало | Результат |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Статус вольера (Enclosure\_Status) | Enclosure\_Status\_Code | Serial | Да | Суррогатный ключ |  | 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); |  |
| Enclosure\_Status\_Name | Varchar (100) | Да | Уникальное |  |
| Перечень работ (Work\_List) | Work\_List\_Code | Serial | Да | Суррогатный ключ |  | create table Work\_List (  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  );  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); |  |
| Work\_List\_Name | Varchar(100) | Да |  |
| Work\_List\_Interval | Interval | Да | > 0,  <= Дата конца работ – Дата начала работ |
| 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 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\_Number varchar(20) not null constraint UQ\_Work\_Plan\_Number unique  constraint CH\_Work\_Plan\_Number 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\_Date date not null,  Work\_Plan\_Instruction varchar(100) not null,  Work\_Plan\_Enclosure 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); |  |
| 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 |
| Work\_Plan\_Start\_Date | Date | Да | >= Дата заявки |
| 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) | Work\_List\_Plan\_Code | Serial | Да | Суррогатный ключ |  | create table Work\_List\_Plan (  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)  );  create index if not exists index\_Work\_List\_Plan\_Code on Work\_List\_Plan (Work\_List\_Plan\_Code); |  |
| Work\_List\_Code | Int | Да | Уникальное,  Внешний ключ |
| Work\_Plan\_Code | Int | Да | Внешний ключ |
| 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\_Name varchar(100) not null  ); | alter table Habitat  add constraint UQ\_Habitat\_Name unique (Habitat\_Name); |  |
| Habitat\_Code | Serial | Да | Суррогатный ключ |  |
| 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 |
| 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\_Number\_Visitor varchar(6) not null,  Benefits\_Visitor int not null,  Password\_Visitor varchar(36) not null  ); | 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!@#$%^&\*()]{8,}'); |  |
| Login\_Visitor | Varchar(50) | Да | Уникальное  Length >= 6, [a-Z] |  |
| Name\_Visitor | Varchar(50) | Да |  |
| Surname\_Visitor | Varchar(50) | Да |  |
| Patronymic\_Visitor | Varchar(50) | Нет | Default “Нет данных“ |
| Passport\_Series\_Visitor | Varchar(4) | Да | [0-9]{4} |
| Passport\_Number\_Visitor | Varchar(6) | Да | [0-9]{6} |
| Benefits\_Visitor | Int | Да | >= 0 |
| Password\_Visitor | Varchar(36) | Да | Length >= 8, [a-Z][0-9][!@#$%^&\*()] |
| Вид посетителя (Visitor Type) | ID\_Visitor\_Type | 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\_Type  add constraint UQ\_Name\_Visitor\_Type unique (Name\_Visitor\_Type); |  |
| Name\_Visitor\_Type | Varchar(50) | Да | Уникальное |
| 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\*'); |  |
| 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}) |
| 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) null,  Password\_Employee varchar(36) not null,  Employee\_Post\_Code int not null references Employee\_Post (Employee\_Post\_Code)  ); | 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) null,  Password\_Employee varchar(36) not null,  Employee\_Post\_Code int not null references Employee\_Post (Employee\_Post\_Code)  ); |  |
| 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“ |
| Password\_Employee | Varchar(36) | Да | Length >= 8, [a-Z][0-9][!@#$%^&\*()] |
| Employee\_Post\_Code | int | Да | Внешний ключ |
| Сотрудники и вольер (Employee\_Enclosure) | ID\_Employee\_Enclosure | Serial | Да | Суррогатный ключ | create table if not exists Employee\_Enclosure (  ID\_Employee\_Enclosure serial not null constraint PK\_Employee\_Enclosure primary key,  ID\_Employee int not null references Employee (ID\_Employee),  ID\_Enclosure int not null references Enclosure (ID\_Enclosure)  ); |  |  |
| ID\_Employee | Int | Да | Внешний ключ |
| ID\_Enclosure | Int | Да | Внешний ключ |
| Животное (Animal) | ID\_Animal | Serial | Да | Суррогатный ключ | create table if not exists Animal (  ID\_Animal serial not null constraint PK\_Animal primary key,  Number\_Animal varchar(11) not null,  Description\_Animal varchar(100) not null,  Picture\_Animal varchar(100) not null,  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 references Territory (ID\_Territory)  ); | alter table Animal  add constraint UQ\_Number\_Animal unique (Number\_Animal),  add constraint CH\_Number\_Animal check (Number\_Animal similar to '[A-Z]+\d{8}'); |  |
| Number\_Animal | Varchar(11) | Да | Уникальное  [A-Z]{2,3}[0-9]{8} |
| ID\_Animal\_Type | Int | Да | Внешний ключ |
| Habitat\_Code | Int | Да | Внешний ключ |
| Description\_Animal | Varchar(100) | Да |  |
| Picture\_Animal | Varchar(100) | Да |  |
| ID\_Enclosure | Int | Да | Внешний ключ |
| ID\_Territory | Int | Да | Внешний ключ |
| Отряд животного (Animal Squad) | ID\_Animal\_Squad | Serial | Да | Суррогатный ключ | create table if not exists Animal\_Squad (  ID\_Animal\_Squad serial not null constraint PK\_Animal\_Squad primary key,  Name\_Animal\_Squad varchar(100) not null  ); | alter table Animal\_Squad  add constraint UQ\_Name\_Animal\_Squad unique(Name\_Animal\_Squad); |  |
| Name\_Animal\_Squad | Varchar(100) | Да | Уникальное |
| Семейство животного (Animal Family) | ID\_Animal\_Family | Serial | Да | Суррогатный ключ | create table if not exists Animal\_Family (  ID\_Animal\_Family serial not null constraint PK\_Animal\_Family primary key,  Name\_Animal\_Family varchar(100) not null,  ID\_Animal\_Squad int not null references Animal\_Squad(ID\_Animal\_Squad)  ); | alter table Animal\_Family  add constraint UQ\_Name\_Animal\_Family unique(Name\_Animal\_Family); |  |
| Name\_Animal\_Family | Varchar(100) | Да | Уникальное |
| ID\_Animal\_Squad | Int | Да | Внешний ключ |
| 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); |  |
| Name\_Animal\_Type | Varchar(100) | Да | Уникальное |
| ID\_Animal\_Family | Int | Да | Внешний ключ |
| Территория для посетителя (Territory) | ID\_Territory | Serial | Да | Суррогатный ключ | create table if not exists Territory (  ID\_Territory serial not null constraint PK\_Territoty 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); |  |
| 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,  Datetime\_Ticket timestamp not null,  Price\_Ticket decimal(6, 2) not null,  Total\_Sum\_Ticket decimal(6, 2) not null,  ID\_Visitor int not null references Visitor (ID\_Visitor)  ); | 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); |  |
| ID\_Visitor | Int | Да | Внешний ключ |
| Number\_Ticket | Varchar(15) | Да | ПБ3-[0-9]{9}/Последние две цифры тек. года |
| Datetime\_Ticket | Timestamp | Да | = Текущая дата |
| Price\_Ticket | Decimal(6,2) | Да | >= 0 |
| Total\_Sum\_Ticket | Decimal(6,2) | Да | >= 0 |
| Территория и билет (Territory\_Ticket) | ID\_Territory\_Ticket | Serial | Да | Суррогатный ключ | create table if not exists Territory\_Ticket (  ID\_Territory\_Ticket serial not null constraint PK\_Territory\_Ticket primary key,  ID\_Territory int not null references Territory (ID\_Territory),  ID\_Ticket int not null references Ticket (ID\_Ticket)  ); |  |  |
| ID\_Territory | Int | Да | Внешний ключ |
| ID\_Ticket | Int | Да | Внешний ключ |
| 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 |
| Время ухода (Care\_Time) | ID\_Caretime | Serial | Да | Суррогатный ключ | create table if not exists Care\_Time (  ID\_Care\_Time serial not null constraint PK\_Care\_Time primary key,  ID\_Enclosure\_Care\_Day int not null references Enclosure\_Care\_Day(ID\_Enclosure\_Care\_Day),  Care\_Time time not null  ); |  |  |
| ID\_Enclosure\_Care\_Day | Int | Да | Внешний ключ |
| Care\_Time | Time | Да |  |

2. Сопровождение хранимых процедур;

2.1. Создание файла;



2.2. Набор тестовых сценариев для сопровождения хранимых процедур;  
Таблица 2 – Сценарии тестирования

| **№ Сценария** | **Характеристики** |
| --- | --- |
|  | ***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! |
|  | ***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: Х1, период: 01.06.2024-10.06.2024, перемещение: Тест, Статус: Ожидается начало. |
| ***Ожидаемый результат*** |
| ГРМ-24-0000000004 |
|  | ***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. |
|  | ***Краткое описание теста*** |
| 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! |
|  | ***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. Сопровождение и тестирование хранимых процедур;

Таблица 3 – Реализация хранимых процедур

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Сценария** | Скрипт | Результат | Статус тестирования |
|  | create or replace procedure Territory\_Insert (p\_Name\_Territory varchar(100), p\_Price\_Territory decimal(2, 5))  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('Океанариум', 100.00);  select \* from territory; |  | Пройден |
|  | 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), p\_Work\_Plan\_Enclosure int, p\_Work\_Plan\_Status int)  language plpgsql  as $$  declare  work\_plan\_year varchar(4) := date\_part('year', p\_Work\_Plan\_Date);  p\_Work\_plan\_number varchar(20) := 'ГРМ-'||substring(work\_plan\_year, 3, 2);  work\_plan\_counter integer := (select count(\*) from work\_plan) + 1;  work\_plan\_zeros varchar(11) := LPAD(work\_plan\_counter::text, 10, '0');  begin  p\_Work\_plan\_number := p\_Work\_plan\_number || '-' || work\_plan\_zeros;  insert into Work\_Plan(Work\_Plan\_Number, Work\_Plan\_Date, Work\_Plan\_Start\_Date, Work\_Plan\_End\_Date, Work\_Plan\_Instruction, Work\_Plan\_Enclosure, Work\_Plan\_Status)  values (p\_Work\_Plan\_Number, p\_Work\_Plan\_Date, p\_Work\_Plan\_Start\_Date, p\_Work\_Plan\_End\_Date, p\_Work\_Plan\_Instruction, p\_Work\_Plan\_Enclosure, p\_Work\_Plan\_Status);  end;  $$; |  | Пройден |
|  | create or replace procedure Enclosure\_Status\_Delete (p\_ID\_Enclosure\_Status int)  language plpgsql  as $$  declare  p\_Exist\_Enclosures smallint := count(\*) from Enclosure  where status\_enclosure = p\_ID\_Enclosure\_Status;  begin  if(p\_Exist\_Enclosures > 0) then  raise notice 'Данный статус вольера не может быть удалён, т.к. он используется в вольерах';  else  delete from Enclosure  where  ID\_Enclosure = p\_ID\_Enclosure;  end if;  end;  $$;  call Enclosure\_Status\_Delete(1); |  | Пройден |
|  | 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 $$  declare  p\_old\_animal\_habitat\_code int := (select Habitat\_Code from animal where id\_animal = p\_id\_animal);  begin  if(p\_old\_animal\_habitat\_code = p\_Habitat\_Code) then  raise notice 'Выбранный ареал уже есть у указанного животного!';  else  update Animal set  Number\_Animal = p\_Number\_Animal,  Description\_Animal = p\_Description\_Animal,  Picture\_Animal = p\_Picture\_Animal,  ID\_Animal\_Type = p\_ID\_Animal\_Type,  Habitat\_Code = p\_Habitat\_Code,  ID\_Enclosure = p\_ID\_Enclosure,  ID\_Territory = p\_ID\_Territory  where  ID\_Animal = p\_ID\_Animal;  end if;  end;  $$;  call animal\_update(1, 's', 's', 'f', 1, 1, 1, 1); |  | Пройден |
|  | create or replace procedure Territory\_Ticket\_Insert (p\_ID\_Territory int, p\_ID\_Ticket int)  language plpgsql  as $$  declare  p\_territory\_ticket\_old\_sum decimal(6, 2) := (select total\_sum\_ticket from ticket where id\_ticket = p\_id\_ticket);  p\_Visitor\_Id int := (select id\_visitor from ticket where id\_ticket = p\_id\_ticket);  p\_Visitor\_Benefits int := (select benefits\_visitor from visitor where id\_visitor = p\_Visitor\_Id);  p\_territory\_price decimal(5, 2) := (select price\_territory from territory where id\_territory = p\_ID\_Territory);  p\_territory\_ticket\_new\_sum decimal(6, 2) := 0;  begin  if (p\_Visitor\_Benefits > 0) then  p\_territory\_ticket\_new\_sum := p\_territory\_ticket\_old\_sum + (p\_territory\_price \* p\_Visitor\_Benefits);  else  p\_territory\_ticket\_new\_sum := p\_territory\_ticket\_old\_sum + p\_territory\_price;  end if;  update ticket set  total\_sum\_ticket = p\_territory\_ticket\_new\_sum  where id\_ticket = p\_id\_ticket;  insert into Territory\_Ticket (ID\_Territory, ID\_Ticket)  values (p\_ID\_Territory, p\_ID\_Ticket);  end;  $$;  call territory\_ticket\_insert(3, 1); |  | Пройден |

2.4. Создание не реализованных ранее хранимых процедур и дополнение процедур для верификации данных;

Таблица 4 – Реализация хранимых процедур

| № ПП | Скрипт | Результат |
| --- | --- | --- |
|  | exception when others then  raise notice 'Указанный ареал обитания уже есть в таблице!'; |  |
|  | create or replace procedure Enclosure\_Status\_Update (p\_enclosure\_status\_code int, p\_enclosure\_status\_name Varchar(100))  language plpgsql  as $$  begin  update Enclosure\_Status set  enclosure\_status\_name = p\_enclosure\_status\_name  where  enclosure\_status\_code = p\_enclosure\_status\_code;  end;  $$; |
|  | create or replace procedure Enclosure\_Status\_Delete (p\_ID\_Enclosure\_Status int)  language plpgsql  as $$  declare  p\_Exist\_Enclosures smallint := count(\*) from Enclosure  where status\_enclosure = p\_ID\_Enclosure\_Status;  begin  if(p\_Exist\_Enclosures > 0) then  raise notice 'Данный статус вольера не может быть удалён, т.к. он используется в вольерах';  else  delete from Enclosure  where  ID\_Enclosure = p\_ID\_Enclosure;  end if;  end;  $$; |
|  | create or replace procedure work\_list\_Insert (p\_work\_list\_name varchar(100), p\_work\_list\_interval interval)  language plpgsql  as $$  begin  insert into work\_list (work\_list\_name, work\_list\_interval)  values (p\_work\_list\_name, p\_work\_list\_interval);  exception when others then  raise notice 'Указанная работа уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update work\_list set  work\_list\_name = p\_work\_list\_name,  work\_list\_interval = p\_work\_list\_interval  where  work\_list\_code = p\_work\_list\_code;  end;  $$; |
|  | create or replace procedure work\_list\_Delete (p\_work\_list\_code int)  language plpgsql  as $$  declare  p\_Exist\_Works smallint := count(\*) from Work\_List\_plan  where code\_work\_list = p\_work\_list\_code;  begin  if(p\_Exist\_Works > 0) then  raise notice 'Данная работа не может быть удалена, т.к. она используется в плане работ';  else  delete from work\_list  where  work\_list\_code = p\_work\_list\_code;  end if;  end;  $$; |
|  | create or replace procedure Work\_Status\_Insert (Work\_Status\_Name varchar(100))  language plpgsql  as $$  begin  insert into Work\_Status (Work\_Status\_name)  values (p\_Work\_Status\_name);  end;  $$; |  |
|  | create or replace procedure Work\_Status\_Update (p\_Work\_Status\_code int, p\_Work\_Status\_name Varchar(100))  language plpgsql  as $$  begin  update Work\_Status set  Work\_Status\_name = p\_Work\_Status\_name  where  Work\_Status\_code = p\_Work\_Status\_code;  end;  $$; |
|  | create or replace procedure Work\_Status\_Delete (p\_Work\_Status\_code int)  language plpgsql  as $$  begin  delete from Work\_Status  where  Work\_Status\_code = p\_Work\_Status\_code;  end;  $$; |
|  | create or replace procedure Work\_List\_Plan\_Insert (p\_code\_Work\_List int, p\_code\_Work\_Plan int)  language plpgsql  as $$  begin  insert into Work\_List\_Plan (code\_Work\_List, code\_Work\_Plan)  values (p\_code\_Work\_List, p\_code\_Work\_Plan);  end;  $$; |  |
|  | 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 $$  begin  update Work\_List\_Plan set  code\_work\_List = p\_code\_Work\_List,  code\_work\_plan = p\_code\_Work\_Plan  where  work\_list\_plan\_code = p\_work\_list\_plan\_code;  end;  $$; |
|  | create or replace procedure Work\_List\_Plan\_Delete (p\_work\_list\_plan\_code int)  language plpgsql  as $$  begin  delete from Work\_List\_Plan  where  work\_list\_plan\_code = p\_work\_list\_plan\_code;  end;  $$; |
|  | 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), p\_Work\_Plan\_Enclosure int, p\_Work\_Plan\_Status int)  language plpgsql  as $$  declare  work\_plan\_year varchar(4) := date\_part('year', p\_Work\_Plan\_Date);  p\_Work\_plan\_number varchar(20) := 'ГРМ-'||substring(work\_plan\_year, 3, 2);  work\_plan\_counter integer := (select count(\*) from work\_plan) + 1;  work\_plan\_zeros varchar(11) := LPAD(work\_plan\_counter::text, 10, '0');  begin  p\_Work\_plan\_number := p\_Work\_plan\_number || '-' || work\_plan\_zeros;  insert into Work\_Plan(Work\_Plan\_Number, Work\_Plan\_Date, Work\_Plan\_Start\_Date, Work\_Plan\_End\_Date, Work\_Plan\_Instruction, Work\_Plan\_Enclosure, Work\_Plan\_Status)  values (p\_Work\_Plan\_Number, p\_Work\_Plan\_Date, p\_Work\_Plan\_Start\_Date, p\_Work\_Plan\_End\_Date, p\_Work\_Plan\_Instruction, p\_Work\_Plan\_Enclosure, p\_Work\_Plan\_Status);  end;  $$; |  |
|  | create or replace procedure work\_plan\_Update (p\_work\_plan\_code int, p\_Work\_Plan\_Date date, p\_Work\_Plan\_Start\_Date date,  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 $$  begin  update Work\_Plan set  Work\_Plan\_Date = p\_Work\_Plan\_Date,  Work\_Plan\_Start\_Date = p\_Work\_Plan\_Start\_Date,  Work\_Plan\_End\_Date = p\_Work\_Plan\_End\_Date,  Work\_Plan\_Instruction = p\_Work\_Plan\_Instruction,  Work\_Plan\_Enclosure = p\_Work\_Plan\_Enclosure,  Work\_Plan\_Status = p\_Work\_Plan\_Status  where  work\_plan\_code = p\_work\_plan\_code;  end;  $$; |
|  | create or replace procedure work\_plan\_Delete (p\_work\_plan\_code int)  language plpgsql  as $$  declare  p\_Exist\_Work\_Plans smallint := count(\*) from Work\_List\_plan  where code\_work\_plan = p\_work\_plan\_code;  begin  if(p\_Exist\_Works > 0) then  raise notice 'Данный план не может быть удален, т.к. он используется в списке работ';  else  delete from work\_plan  where  work\_plan\_code = p\_work\_plan\_code;  end if;  end;  $$; |
|  | create or replace procedure Habitat\_Insert (p\_habitat\_name varchar(100))  language plpgsql  as $$  begin  insert into Habitat (habitat\_name)  values (p\_habitat\_name);  exception when others then  raise notice 'Указанный ареал обитания уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Habitat\_Update (p\_habitat\_code int, p\_habitat\_name Varchar(100))  language plpgsql  as $$  begin  update Habitat set  habitat\_name = p\_habitat\_name  where  habitat\_code = p\_habitat\_code;  exception when others then  raise notice 'Указанный ареал обитания уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Employee\_Post\_Insert (p\_Employee\_Post\_Name varchar(100))  language plpgsql  as $$  begin  insert into Employee\_Post (Employee\_Post\_Name)  values (p\_Employee\_Post\_Name);  exception when others then  raise notice 'Указанная должность сотрудника уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Habitat\_Delete (p\_habitat\_code int)  language plpgsql  as $$  declare  p\_Exist\_animals smallint := count(\*) from animal  where habitat\_code = p\_habitat\_code;  begin  if(p\_Exist\_animals > 0) then  raise notice 'Данный ареал обитания не может быть удален, т.к. он используется у животных';  else  delete from Habitat  where  habitat\_code = p\_habitat\_code;  end if;  end;  $$;  call Habitat\_Delete(1); |  |
|  | create or replace procedure Employee\_Post\_Delete (p\_Employee\_Post\_Code int)  language plpgsql  as $$  declare  p\_Exist\_employees smallint := count(\*) from employee  where employee\_post\_code = p\_Employee\_Post\_Code;  begin  if(p\_Exist\_employees > 0) then  raise notice 'Данная должность сотрудника не может быть удалена, т.к. она используется в сотрудниках';  else  delete from Employee\_Post  where  Employee\_Post\_Code = p\_Employee\_Post\_Code;  end if;  end;  $$; |  |
|  | 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 $$  begin  insert into Visitor (Login\_Visitor, Name\_Visitor, Surname\_Visitor, Patronymic\_Visitor, Passport\_Series\_Visitor, Passport\_Number\_Visitor,  Benefits\_Visitor, Password\_Visitor)  values (p\_Login\_Visitor, p\_Name\_Visitor, p\_Surname\_Visitor, p\_Patronymic\_Visitor, p\_Passport\_Series\_Visitor, p\_Passport\_Number\_Visitor,  p\_Benefits\_Visitor, p\_Password\_Visitor);  exception when others then  raise notice 'Указанный посетитель уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Visitor set  Login\_Visitor = p\_Login\_Visitor,  Name\_Visitor = p\_Name\_Visitor,  Surname\_Visitor = p\_Surname\_Visitor,  Patronymic\_Visitor = p\_Patronymic\_Visitor,  Passport\_Series\_Visitor = p\_Passport\_Series\_Visitor,  Passport\_Number\_Visitor = p\_Passport\_Number\_Visitor,  Benefits\_Visitor = p\_Benefits\_Visitor,  Password\_Visitor = p\_Password\_Visitor  where  ID\_Visitor = p\_ID\_Visitor;  exception when others then  raise notice 'Указанный посетитель уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Visitor\_Delete (p\_ID\_Visitor int)  language plpgsql  as $$  declare  p\_Exist\_visitors smallint := count(\*) from ticket  where id\_visitor = p\_ID\_Visitor;  begin  if(p\_Exist\_visitors > 0) then  raise notice 'Данная посетитель не может быть удален, т.к. он используется в билетах';  else  delete from Visitor  where  ID\_Visitor = p\_ID\_Visitor;  end if;  end;  $$; |  |
|  | create or replace procedure Visitor\_Type\_Insert (p\_Name\_Visitor\_Type varchar(50))  language plpgsql  as $$  begin  insert into Visitor\_Type (Name\_Visitor\_Type)  values (p\_Name\_Visitor\_Type);  exception when others then  raise notice 'Указанный вид посетителя уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Visitor\_Type\_Update (p\_ID\_Visitor\_Type int, p\_Name\_Visitor\_Type Varchar(50))  language plpgsql  as $$  begin  update Visitor\_Type set  Name\_Visitor\_Type = p\_Name\_Visitor\_Type  where  ID\_Visitor\_Type = p\_ID\_Visitor\_Type;  exception when others then  raise notice 'Указанный вид посетителя уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Visitor\_Type\_Delete (p\_ID\_Visitor\_Type int)  language plpgsql  as $$  declare  p\_Exist\_type\_visitors smallint := count(\*) from Visitor\_Document  where id\_visitor = p\_ID\_Visitor\_Type;  begin  if(p\_Exist\_type\_visitors > 0) then  raise notice 'Данная вид посетителя не может быть удален, т.к. он используется в документах';  else  delete from Visitor\_Type  where  ID\_Visitor\_Type = p\_ID\_Visitor\_Type;  end if;  end;  $$; |  |
|  | create or replace procedure Visitor\_Document\_Insert (p\_Number\_Document varchar(100), p\_ID\_Visitor\_Type int, p\_ID\_Visitor int)  language plpgsql  as $$  begin  insert into Visitor\_Document (Number\_Document, ID\_Visitor\_Type, ID\_Visitor)  values (p\_Number\_Document, p\_ID\_Visitor\_Type, p\_ID\_Visitor);  exception when others then  raise notice 'Указанный документ уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Visitor\_Document set  Number\_Document = p\_Number\_Document,  ID\_Visitor\_Type = p\_ID\_Visitor\_Type,  ID\_Visitor = p\_ID\_Visitor  where  ID\_Document = p\_ID\_Document;  exception when others then  raise notice 'Указанный документ уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Enclosure\_Insert (p\_Name\_Enclosure varchar(50), p\_Status\_Enclosure varchar(16))  language plpgsql  as $$  begin  insert into Enclosure (Name\_Enclosure, Status\_Enclosure)  values (p\_Name\_Enclosure, p\_Status\_Enclosure);  exception when others then  raise notice 'Указанный вольер уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Enclosure\_Update (p\_ID\_Enclosure int, p\_Name\_Enclosure varchar(50), p\_Status\_Enclosure varchar(16))  language plpgsql  as $$  begin  update Enclosure set  Name\_Enclosure = p\_Name\_Enclosure,  Status\_Enclosure = p\_Status\_Enclosure  where  ID\_Enclosure = p\_ID\_Enclosure;  exception when others then  raise notice 'Указанный вольер уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Enclosure\_Delete (p\_ID\_Enclosure int)  language plpgsql  as $$  declare  p\_Exist\_Enclosures smallint := count(\*) from employee\_enclosure  where id\_enclosure = p\_ID\_Enclosure;  begin  if(p\_Exist\_Enclosures > 0) then  raise notice 'Данная вольер не может быть удален, т.к. он используется в сотрудниках-вольерах';  else  delete from Enclosure  where  ID\_Enclosure = p\_ID\_Enclosure;  end if;  end;  $$; |  |
|  | 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 $$  begin  insert into Employee (Login\_Employee, Surname\_Employee, Name\_Employee, Patronymic\_Employee, Password\_Employee, Employee\_Post\_Code)  values (p\_Login\_Employee, p\_Surname\_Employee, p\_Name\_Employee, p\_Patronymic\_Employee, p\_Password\_Employee, p\_Employee\_Post\_Code);  exception when others then  raise notice 'Указанный сотрудник уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Employee set  Login\_Employee = p\_Login\_Employee,  Surname\_Employee = p\_Surname\_Employee,  Name\_Employee = p\_Name\_Employee,  Patronymic\_Employee = p\_Patronymic\_Employee,  Password\_Employee = p\_Password\_Employee,  Employee\_Post\_Code = p\_Employee\_Post\_Code  where  ID\_Employee = p\_ID\_Employee;  exception when others then  raise notice 'Указанный сотрудник уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Employee\_Delete (p\_ID\_Employee int)  language plpgsql  as $$  declare  p\_Exist\_employees smallint := count(\*) from employee\_enclosure  where id\_employee = p\_ID\_Employee;  begin  if(p\_Exist\_employees > 0) then  raise notice 'Данная сотрудник не может быть удален, т.к. он используется в сотрудниках-вольерах';  else  delete from Employee  where  ID\_Employee = p\_ID\_Employee;  end if;  end;  $$; |  |
|  | create or replace procedure Animal\_Squad\_Insert (p\_Name\_Animal\_Squad varchar(100))  language plpgsql  as $$  begin  insert into Animal\_Squad (Name\_Animal\_Squad)  values (p\_Name\_Animal\_Squad);  exception when others then  raise notice 'Указанный отдряд уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Animal\_Squad\_Update (p\_ID\_Animal\_Squad int, p\_Name\_Animal\_Squad Varchar(100))  language plpgsql  as $$  begin  update Animal\_Squad set  Name\_Animal\_Squad = p\_Name\_Animal\_Squad  where  ID\_Animal\_Squad = p\_ID\_Animal\_Squad;  exception when others then  raise notice 'Указанный отдряд уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Animal\_Squad\_Delete (p\_ID\_Animal\_Squad int)  language plpgsql  as $$  declare  p\_Exist\_squads smallint := count(\*) from Animal\_Family  where id\_animal\_squad = p\_ID\_Animal\_Squad;  begin  if(p\_Exist\_squads > 0) then  raise notice 'Данный отряд не может быть удален, т.к. он используется в семействах';  else  delete from Animal\_Squad  where  ID\_Animal\_Squad = p\_ID\_Animal\_Squad;  end if;  end;  $$; |  |
|  | create or replace procedure Animal\_Family\_Insert (p\_Name\_Animal\_Family varchar(100), p\_ID\_Animal\_Squad int)  language plpgsql  as $$  begin  insert into Animal\_Family (Name\_Animal\_Family, ID\_Animal\_Squad)  values (p\_Name\_Animal\_Family, p\_ID\_Animal\_Squad);  exception when others then  raise notice 'Указанное семейство уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Animal\_Family set  Name\_Animal\_Family = p\_Name\_Animal\_Family,  ID\_Animal\_Squad = p\_ID\_Animal\_Squad  where  ID\_Animal\_Family = p\_ID\_Animal\_Family;  exception when others then  raise notice 'Указанное семейство уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Animal\_Family\_Delete (p\_ID\_Animal\_Family int)  language plpgsql  as $$  declare  p\_Exist\_families smallint := count(\*) from animal\_type  where id\_animal\_family = p\_ID\_Animal\_Family;  begin  if(p\_Exist\_families > 0) then  raise notice 'Данное семейство не может быть удалено, т.к. оно используется в видах';  else  delete from Animal\_Family  where  ID\_Animal\_Family = p\_ID\_Animal\_Family;  end if;  end;  $$; |  |
|  | create or replace procedure Animal\_Type\_Insert (p\_Name\_Animal\_Type varchar(100), p\_ID\_Animal\_Family int)  language plpgsql  as $$  begin  insert into Animal\_Type (Name\_Animal\_Type, ID\_Animal\_Family)  values (p\_Name\_Animal\_Type, p\_ID\_Animal\_Family);  exception when others then  raise notice 'Указанный вид уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Animal\_Type set  Name\_Animal\_Type = p\_Name\_Animal\_Type,  ID\_Animal\_Family = p\_ID\_Animal\_Family  where  ID\_Animal\_Type = p\_ID\_Animal\_Type;  exception when others then  raise notice 'Указанный вид уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Animal\_Type\_Delete (p\_ID\_Animal\_Type int)  language plpgsql  as $$  declare  p\_Exist\_types smallint := count(\*) from animal\_type  where id\_animal\_family = p\_ID\_Animal\_Family;  begin  if(p\_Exist\_types > 0) then  raise notice 'Данное семейство не может быть удалено, т.к. оно используется в видах';  else  delete from Animal\_Type  where  ID\_Animal\_Type = p\_ID\_Animal\_Type;  end if;  end;  $$; |  |
|  | create or replace procedure Territory\_Update (p\_ID\_Territory int, p\_Name\_Territory varchar(100), p\_Price\_Territory decimal(2, 5))  language plpgsql  as $$  begin  update Territory set  Name\_Territory = p\_Name\_Territory,  Price\_Territory = p\_Price\_Territory  where  ID\_Territory = p\_ID\_Territory;  exception when others then  raise notice 'Указанная территория уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Territory\_Delete (p\_ID\_Territory int)  language plpgsql  as $$  declare  p\_Exist\_territories smallint := count(\*) from animal  where id\_territory = p\_ID\_Territory;  begin  if(p\_Exist\_territories > 0) then  raise notice 'Данная территория не может быть удалена, т.к. она используется в животных';  else  delete from Territory  where  ID\_Territory = p\_ID\_Territory;  end if;  end;  $$; |  |
|  | 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 $$  begin  insert into Animal (Number\_Animal, Description\_Animal, Picture\_Animal, ID\_Animal\_Type, Habitat\_Code, ID\_Enclosure, ID\_Territory)  values (p\_Number\_Animal, p\_Description\_Animal, p\_Picture\_Animal, p\_ID\_Animal\_Type, p\_Habitat\_Code, p\_ID\_Enclosure, p\_ID\_Territory);  exception when others then  raise notice 'Указанное животное уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  insert into Ticket (Number\_Ticket, Datetime\_Ticket, Price\_Ticket, Total\_Sum\_Ticket, ID\_Visitor)  values (p\_Number\_Ticket, p\_Datetime\_Ticket, p\_Price\_Ticket, p\_Total\_SUm\_Ticket, p\_ID\_Visitor);  exception when others then  raise notice 'Указанный билет уже есть в таблице!';  end;  $$; |  |
|  | 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 $$  begin  update Ticket set  Number\_Ticket = p\_Number\_Ticket,  Datetime\_Ticket = p\_Datetime\_Ticket,  Price\_Ticket = p\_Price\_Ticket,  Total\_SUm\_Ticket = p\_Total\_SUm\_Ticket,  ID\_Visitor = p\_ID\_Visitor  where  ID\_Ticket = p\_ID\_Ticket;  exception when others then  raise notice 'Указанный билет уже есть в таблице!';  end;  $$; |  |
|  | create or replace procedure Ticket\_Delete (p\_ID\_Ticket int)  language plpgsql  as $$  declare  p\_Exist\_tickets smallint := count(\*) from territory\_ticket  where id\_ticket = p\_ID\_Ticket;  begin  if(p\_Exist\_tickets > 0) then  raise notice 'Данный билет не может быть удален, т.к. он используется в территориях-билетах';  else  delete from Ticket  where  ID\_Ticket = p\_ID\_Ticket;  end if;  end;  $$; |  |
|  | create or replace procedure Enclosure\_Care\_Day\_Insert (p\_ID\_Employee\_Enclosure int, p\_Enclosure\_Care\_Day varchar(11))  language plpgsql  as $$  begin  insert into Enclosure\_Care\_Day (ID\_Employee\_Enclosure,Enclosure\_Care\_Day)  values (p\_ID\_Employee\_Enclosure, p\_Enclosure\_Care\_Day);  exception when others then  raise notice 'Указанный день недели ухода не существует. Используйте один из дней недели';  end;  $$; |  |
|  | 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 $$  begin  update Enclosure\_Care\_Day set  ID\_Employee\_Enclosure = p\_ID\_Employee\_Enclosure,  Enclosure\_Care\_Day = p\_Enclosure\_Care\_Day  where  ID\_Enclosure\_Care\_Day = p\_ID\_Enclosure\_Care\_Day;  exception when others then  raise notice 'Указанный день недели ухода не существует. Используйте один из дней недели';  end;  $$; |  |
|  | create or replace procedure Enclosure\_Care\_Day\_Delete (p\_ID\_Enclosure\_Care\_Day int)  language plpgsql  as $$  declare  p\_Exist\_care\_days smallint := count(\*) from Care\_Time  where id\_enclosure\_care\_day = p\_ID\_Enclosure\_Care\_Day;  begin  if(p\_Exist\_care\_days > 0) then  raise notice 'Данный день ухода не может быть удален, т.к. он используется во времени ухода';  else  delete from Enclosure\_Care\_Day  where  ID\_Enclosure\_Care\_Day = p\_ID\_Enclosure\_Care\_Day;  end if;  end;  $$; |  |

1. 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 |  | Х | X |
| Добавление |  |  | X |
| Изменение |  |  | X |
| Удаление |  |  |  |
| Work\_Plan | Выборка |  | Х | Х |
| Добавление |  |  | X |
| Изменение |  |  | X |
| Удаление |  |  |  |
| Work\_List\_Plan | Выборка |  |  | X |
| Добавление |  |  | X |
| Изменение |  |  | X |
| Удаление |  |  |  |
| Enclosure\_Status\_Insert | Вызов |  |  | X |
| Enclosure\_Status\_Update | Вызов |  |  | Х |
| 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 |

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

Таблица 6 – Реализация разграничения прав доступа

| 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; |
| 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 |  |

1. Создание резервной копии БД;



1. Версия базы данных:
   1. Отчёт о созданных объектах;

Таблица 7 – Перечень созданных объектов

|  | Информация по объектам |
| --- | --- |
| Запрос | 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  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  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); |
| Результат локальной БД |  |
| Результат удалённой БД | Сервер не доступен |

* 1. Версия БД.

Таблица 8 – Версия файла БД

|  |  |
| --- | --- |
| Параметры | PostgreSQL |
| Номер версии | 3.1.0.1 |
| Что сделано | * 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. |