* 1. Сопровождение объектов.

Таблица 1 – Сопровождение таблиц баз данных

| Таблица | Поле | Тип данных | Обязательное | Ограничения | Было | Стало | Результат |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Сотрудник (Employee) | Employee\_login | Varchar(50) | да | Length >= 8  [A-Z]{1,} [a-z]{1,} [!@#$%^&\*()]{1,} | create table if not exists Employee  (      Employee\_kod Serial not null constraint PK\_Employee primary key,      Employee\_surname Varchar(50) not null,      Employee\_name Varchar(50) not null,      Employee\_patronymic Varchar(50) null,      Employee\_login Varchar(50) not null,      Employee\_password Varchar(50) not null  ); | alter table public.employee  add constraint CH\_value\_employee\_login check (employee\_login similar to '%[A-Za-z]+[!@#$%^&\*()\_]+[A-Za-z]+%'),  add constraint CH\_length\_employee\_login  check (length(employee\_login) > 8)  alter table public.employee  add constraint CH\_value\_employee\_password  check (employee\_password similar to '%[A-Za-z]+[!@#$%^&\*()\_]+[A-Za-z]+%'),  add constraint CH\_length\_employee\_password  check (length(employee\_password) >= 8),  alter column employee\_patronymic set Default('-') |  |
| Employee\_password | Varchar(50) | да | Length >= 8  [A-Z]{1,} [a-z]{1,} [!@#$%^&\*()]{1,} |
| Employee\_kod | Int |  | Суррогатный ключ |
| Employee\_patronymic | Varchar(50) | нет | Default “-” |
| Зарегистрированные посетители (Registered visitors) | RV\_patronymic | Varchar(50) | Нет | Default “-” | create table if not exists Registered\_Visitors  (      RV\_surname Varchar(50) not null,      RV\_name Varchar(50) not null,      RV\_patronymic Varchar(50) null,      RV\_login Varchar(50) not null,      RV\_password Varchar(50) not null,      RV\_pasportS Varchar(5) not null,      RV\_pasportN Varchar(6) not null,      RV\_card Varchar(19) not null,      RV\_kod varchar(5) not null constraint PK\_Registered\_Visitors primary key  ); | alter table public.registered\_visitors  alter column rv\_patronymic set default('-'),  add constraint CH\_value\_rv\_login  check(rv\_login similar to '%[A-Za-z]+[!@#$%^&\*()\_]\*[A-Za-z]+%'),  add constraint CH\_lengrh\_rv\_login  check(length(rv\_login) >=8),  add constraint CH\_value\_rv\_password  check(rv\_password similar to '%[A-Za-z]+[!@#$%^&\*()\_]+[A-Za-z]+%'),  add constraint CH\_lengrh\_rv\_password  check(length(rv\_password) >=8),  add constraint CH\_value\_rv\_pasports  check (rv\_pasports similar to '%[0-9]{2}[0-9]{2}%'),  add constraint CH\_value\_rv\_pasportn  check (rv\_pasportn similar to '%[0-9]{6}%'),  add constraint CH\_value\_rv\_card  check (rv\_card similar to '%[0-9]{4}[0-9]{4}[0-9]{4}%'); |  |
| RV\_login | Varchar(50) | Да | Length >= 8  [A-Z]{1,} [a-z]{1,} [!@#$%^&\*()]{1,} |
| RV\_password | Varchar(50) | Да | Length >= 8  [A-Z]{1,} [a-z]{1,} [!@#$%^&\*()]{1,}[1-9]{1,} |
| RV\_pasportS | Varchar(5) | Да | [0-9]{2} [0-9]{2} |
| RV\_pasportN | Varchar(6) | Да | [0-9]{6} |
| RV\_card | Varchar(19) | Да | [0-9]{4} [0-9]{4} [0-9]{4} |
| RV\_kod | Int | Да | Суррогатный ключ |
| Незарегистрированные посетители (visitors) | V\_patonymic | Varchar(50) | нет | Default “Нет данных” | create table if not exists Visitors  (      V\_surname Varchar(50) not null,      V\_name Varchar(50) not null,      V\_patronymic Varchar(50) null,      V\_kod varchar(5) not null constraint PK\_Visitors primary key  ); | alter table visitors  alter column v\_patronymic set default('-') | - |
| V\_kod | Int | Да | Суррогатный ключ |
| Заказ (Orders) | Order\_name | Varchar(15) | Да | ЗКЗ-[0-9]{9}-[0-9]{2} | create table if not exists Orders  (      Order\_name Varchar(16) not null,      Order\_dishCount Int not null,      Order\_dishTime Time not null,      Order\_date Date not null,      Order\_openTime Time not null,      Order\_kod Serial not null constraint PK\_Orders primary key,      Table\_kod Int not null references Wood\_tables (Table\_kod),      Employee\_kod Int not null references Employee (Employee\_kod),      Status\_kod Int not null references Status (Status\_kod),      dish\_kod Int not null references dishes(dish\_kod)  ); | alter table orders  add constraint CH\_value\_order\_name  check(order\_name similar to '%ЗКЗ-[0-9]{9}-[0-9]{2}%'),  add constraint CH\_time\_order\_dishtime  check (order\_dishtime >= order\_opentime),  add constraint CH\_value\_order\_dishcount  check (order\_dishcount >=0); |  |
| Order\_dishCount | Int | Да | >=0 |
| Order\_dishTime | Time | Да | >=время открытия заказа |
| Order\_date | Date | Да | = Текущая дата |
| Order\_closeTime | Time | Да | = Текущее время |
| Order\_kod | Int | Да | Суррогатный ключ |
| Table\_kod | Int | Да | Внешний ключ |
| Employee\_kod | Int | Да | Внешний ключ |
| Status\_kod | Int | Да | Внешний ключ |
| Чек (Checkues) | Checkue\_number | Varchar(13) | Да | Уникальное  КЧ-[0-9]{7}/[0-9]{2} | create table if not exists Checkues  (      Checkue\_number Varchar(13) not null,      Checkue\_date Date not null,      Checkue\_time Time not null,      Checkue\_sumFinal Int not null,      Checkue\_sumGotten Int not null,      Checkue\_changeSum Int not null,      Checkue\_PayType Varchar(50) not null,      Checkue\_finalCost Int not null,      Checkue\_kod Serial not null constraint PK\_Checkues primary key,      Order\_kod Int not null references Orders(Order\_kod)  ); | alter table checkues  add constraint CH\_value\_checkue\_number  check (checkue\_number similar to '%КЧ-[0-9]{7}/[0-9]{2}%'),  add constraint UQ\_value\_checkue\_number unique(checkue\_number),  add constraint CH\_value\_checkue\_sumgotten  check(checkue\_sumgotten >= 0),  add constraint CH\_value\_checkue\_sumfinal  check(checkue\_sumfinal >= 0),  add constraint CH\_value\_checkue\_changesum  check(checkue\_changesum >= 0),  add constraint CH\_value\_checkue\_paytype  check(checkue\_paytype in ('Наличный','Безналичный')),  add constraint CH\_value\_checkue\_finalcost  check(checkue\_finalcost >=0); |  |
| Checkue \_date | Date | Да | =дата открытия заказа  >=время заказа порции |
| Checkue \_time | Time | Да | =время открытия заказа  >=время заказа порции |
| Checkue \_sumFinal | Int | Да | >=0 |
| Checkue\_sumGotten | Int | Да | >=0 |
| Checkue\_ changeSum | Int | Да | >=0 |
| Checkue\_PayType | Varchar(50) | Да | Наличный, Безналичный |
| Checkue\_finalCost | Int | Да | >=0 |
| Checkue\_kod | Int | Да | Суррогатный ключ |
| Order\_kodста | Int | Да | Внешний ключ |
| Поставщики(providers) | Provider\_house | Int | Да | >=0 | create table if not exists Providers  (      Provider\_city Varchar(50) not null,      Provider\_street Varchar(50) not null,      Provider\_house Int not null,      Provider\_flatNumber Int not null,      Provider\_name Varchar(50) not null,      Provider\_okpo Varchar(10) not null,      Provider\_kod Serial not null constraint PK\_Providers primary key  ); | alter table providers  add constraint CH\_value\_provider\_house  check(provider\_house >= 0),  add constraint CH\_value\_provider\_flatnumber  check(provider\_flatnumber >= 0),  add constraint CH\_value\_provider\_okpo  check(provider\_okpo similar to '%[0-9]{8,10}%'),  add constraint UQ\_value\_provider\_okpo unique(provider\_okpo); |  |
| Provider\_flatNumber | Int | Да | >=0 |
| Provider \_okpo | Varchar(10) | Да | Уникально  [0-9]{8,10} |
| Provider\_kod | Int | Да | Суррогатный ключ |
| Смета (Smeta) | Smeta\_date | Date | Да | =текущая дата | create table if not exists Smeta  (      Smeta\_date Date not null,      Smeta\_number Varchar(13),      Smeta\_kod Serial not null constraint PK\_Smeta primary key  ); | alter table public.smeta  add constraint CH\_value\_smeta\_number  check (smeta\_number similar to '%СП-[0-9]{7}-[0-9]{2}%'),  add constraint UQ\_value\_smeta\_number unique(smeta\_number); |  |
| Smeta\_number | Varchar(13) | Да | Уникальное  СП-[0-9]{7}-[0-9]{2} |
| Smeta\_kod | Int | Да | Суррогатный ключ |
| Стол (Tables) | Table\_placesCount | Int | Да | >0 | create table if not exists Wood\_tables  (      Table\_placesCount Int not null,      Wood\_table\_name Varchar(4) not null,      Table\_kod Serial not null constraint PK\_Wood\_Tables primary key,      Zona\_kod Int not null references Zones(Zona\_kod)  ); | alter table public.wood\_tables  add constraint CH\_value\_table\_placescount  check(table\_placescount >= 0),  add constraint CH\_value\_wood\_table\_name  check(wood\_table\_name similar to '%[А-Яа-ё]{1,2}[0-9]{1,2}%'),  add constraint UQ\_value\_wood\_table\_name unique(wood\_table\_name); | u |
| Table\_name | Varchar(4) | Да | Уникальное  [А-Я]{1,2}[0-9]{1,2} |
| Table\_kod | Int | Да | Суррогатный ключ |
| Zona\_kod | Int | Да | Внешний ключ |
| Блюда (Dishes) | Dish\_name | Varchar(50) | Да | Уникальное | create table if not exists Dishes  (      Dish\_name Varchar(50) not null,      Dish\_cost Int not null,      Dish\_weight Decimal(6,2) not null,      Dish\_picture Varchar(1) not null,      Dish\_kod Serial not null constraint PK\_Dishes primary key  ); | alter table dishes  add constraint UQ\_value\_dish\_name unique(dish\_name),  add constraint CH\_value\_dish\_cost  check(dish\_cost >=0),  add constraint CH\_value\_dish\_weight  check(dish\_weight >=0),  alter column dish\_picture set default('no data');  alter table dishes  add column dish\_picture varchar;  update dishes  set dish\_picture = 'null\_1'  where  dish\_name = 'Филе порося';  update dishes  set dish\_picture = 'null\_2'  where  dish\_name = 'Суп мечты';  update dishes  set dish\_picture = 'null\_3'  where  dish\_name = 'Картофель по своему';  update dishes  set dish\_picture = 'null\_4'  where  dish\_name = 'Мясная тарелка';  update dishes  set dish\_picture = 'null\_5'  where  dish\_name = 'Гарнир офощной'; | alter table dishes  add constraint UQ\_value\_dish\_name unique(dish\_name),  add constraint CH\_value\_dish\_cost  check(dish\_cost >=0),  add constraint CH\_value\_dish\_weight  check(dish\_weight >=0),  alter column dish\_picture set default('no data'); |
| Dish\_cost | Int | Да | >=0 |
| Dish\_weight | Decimal(4,2) | Да | >=0 |
| Dish\_picture | Varchar() | Да | Default “Нет данных” |
| Dish\_kod | Int | Да | Суррогатный ключ |
| Dish\_picture | Varchar | Да | Уникальное |
| Статус (Status) | Status\_value | Varchar(50) | Да | выдан, ожидается, в готовке | create table if not exists Status  (      Status\_value Varchar(50) not null,      Status\_kod Serial not null constraint PK\_Status primary key  ); | alter table status  add constraint CH\_value\_status\_value  check(status\_value in ('Выдан','Ожидается','В готовке')) |  |
| Status\_kod | Int | Да | Суррогатный ключ |
| Зона (Zones) | Zona\_name | Varchar(100) | Да | Уникальное | create table if not exists Zones  (      Zona\_name Varchar(100) not null,      Zona\_kod Serial not null constraint PK\_Zones primary key  ); | alter table zones  add constraint UQ\_value\_zona\_name unique(zona\_name); |  |
| Zona\_kod | Int | Да | Суррогатный ключ |
| Ингредиенты(ingredientes) | Ingredient\_name | Varchar(50) | Да | Уникальное | create table if not exists Ingredientes  (      Ingredient\_name Varchar(50) not null,      Ingredient\_kod Serial not null constraint PK\_Ingredientes primary key  ); | alter table public.ingredientes  add constraint UQ\_value\_ingredient\_name unique(ingredient\_name) |  |
| Ingredient\_kod | Int | Да | Суррогатный ключ |
| Индивидуальные заказы (IndividualOrders) | IndividualOrder\_kod | Int | Да | Суррогатный ключ | create table if not exists IndividualOrders  (      IndividualOrder\_kod Serial not null constraint PK\_IndividualOrders primary key,      Order\_kod Int not null references Orders(Order\_kod),      V\_kod varchar(5) not null,      person\_cost Int not null  ); | alter table public.individualorders  add constraint CH\_value\_person\_cost  check(person\_cost >= 0) |  |
| Order\_kod | Int | Да | Внешний ключ |
| V\_kod | Int | Да | Внешний ключ |
| RV\_kod | Int | Да | Внешний ключ |
| Person\_cost | Int | Да | >=0 |
| Закупка ингредиентов (PurchaseIngredient) | Purchase\_kod | Int | Да | Суррогатный ключ | create table if not exists PurchaseIngredient  (  Purchase\_kod Serial not null constraint PK\_PurchaseIngredient primary key,  Ingredient\_weight Int not null,  Shipment\_kod Int not null references Shipments (Shipment\_kod),  Ingredient\_kod Int not null references Ingredientes(Ingredient\_kod)  ); | alter table public.purchaseingredient  add constraint CH\_value\_ingredient\_weight  check(ingredient\_weight >= 0) |  |
| Ingredient\_weight | Int | Да | >=0 |
| Shipment\_kod | Int | Да | Внешний ключ |
| Ingredient\_kod | Int | Да | Внешний ключ |
| Поставки (Shipments) | Shipment\_kod | Int | Да | Суррогатный ключ | create table if not exists Shipments  (  Shipment\_kod Serial not null constraint PK\_Shipments primary key,  Smeta\_kod Int not null references Smeta (Smeta\_kod),  Provider\_kod Int not null references Providers(Provider\_kod)  ); |  |  |
| Smeta\_kod | Int | Да | Внешний ключ |
| Provider\_kod | Int | Да | Внешний ключ |
| Состав блюд (Sostav) | Sostav\_kod | Int | Да | Суррогатный ключ | create table if not exists Sostav  (  Sostav\_kod Serial not null constraint PK\_Sostav primary key,  Ingredient\_kod Int not null references Ingredientes(Ingredient\_kod),  Dish\_kod Int not null references Dishes(Dish\_kod)  ); |  |  |
| Ingredient\_kod | Int |  | Внешний ключ |
| Dish\_kod | Int |  | Внешний ключ |
| Бронирование (booking) | Book\_kod | Int | Да | Сурргоатный ключ |  | create table if not exists booking  (  booking\_kod serial not null constraint pk\_booking\_kod primary key,  book\_number varchar(16) not null  constraint ch\_value\_booking\_number  check (book\_number similar to '%БР/[0-9]{2}/[0-9]{10}%'),  book\_making\_date date not null,  book\_making\_time time not null,    book\_plan\_date date not null  constraint ch\_value\_plan\_date  check(book\_plan\_date >book\_making\_date),    book\_plan\_time time not null constraint ch\_value\_plan\_time  check (book\_plan\_time > book\_making\_time),    book\_guests\_count int not null constraint ch\_value\_guests\_count  check(book\_guests\_count >=1),  rv\_kod varchar(5) not null references public.registered\_visitors(rv\_kod),  v\_kod varchar(5) not null references public.visitors(v\_kod),  wood\_table\_kod int not null references public.wood\_tables(table\_kod)  )  create index if not exists index\_booking\_kod on booking(booking\_kod);  create index if not exists index\_plan\_timedate on booking(book\_plan\_date, book\_plan\_date);  create index if not exists index\_making\_timedate on booking(book\_making\_date, book\_making\_time); |  |
| Book\_number | Varchar(16) | Да | БР/[0-9]{2}/[0-9]{10} |
| Book\_making\_date | date | Да | >= текущая дата |
| Book\_making\_time | time | Да | >= текущая время |
| Book\_plan\_date | Date | Да | >= дата создания брони |
| Book\_plan\_time | time | Да | >= время создания брони |
| Book\_guests\_count | int | Да | >= количество мест стола  >= 1 |
| Rv\_kod | varchar(5) | Да | венишний ключ |
| V\_kod | varchar(5) | Да | венишний ключ |
| Wood\_table\_kod | varchar(5) | Да | венишний ключ |
| Предварительные заказы(planning\_orders) | Planning\_orders\_kod | int | Да | Сурргоатный ключ |  | create table if not exists planning\_orders  (  planning\_order\_kod serial not null constraint pk\_planning\_order\_kod primary key,  order\_kod int not null references orders(order\_kod),  book\_kod int not null references booking(booking\_kod)  );  create index if not exists index\_planning\_order\_kod on planning\_orders(planning\_order\_kod); | - |
| Order\_kod | int | Да | венишний ключ |
| Book\_kod | int | Да | венишний ключ |
| Состав блюд бронирования(planning\_dishes) | dish \_kod | int | Да | венишний ключ |  | create table if not exists planning\_dishes  (  planning\_dishes\_kod serial not null constraint pk\_plan\_dish\_kod primary key,  planning\_dishes\_time time not null,  dish\_kod int not null references dishes(dish\_kod),  book\_kod int not null references booking(booking\_kod)  );  create index if not exists index\_planing\_dishes\_kod on planning\_dishes(planning\_dishes\_kod); | - |
| planning\_dishes\_time | int | Да | <= время посещения |
| Book\_kod | int | Да | венишний ключ |
| Planning\_dishes\_kod | int | да | Сурргоатный ключ |

1. Сопровождение хранимых процедур:
   1. Создание файла;



* 1. Набор тестовых сценариев для сопровождения хранимых процедур;

Таблица 2 – Сценарии тестирования

| **№ Сценария** | **Характеристики** |
| --- | --- |
|  | ***Краткое описание теста*** |
| Ввод существующего стола, с выводом сообщения об ошибке. |
| ***Поля ввода*** |
| Номер стола. |
| ***Вводимые данные*** |
| Д2 |
| ***Ожидаемый результат*** |
| Указанный стол уже есть в таблице! |
|  | ***Краткое описание теста*** |
| Автоматическое формирование номера бронирования. |
| ***Поля ввода*** |
| Дата и время формирования, Код-ключ стола, Код-ключ посетителя, дата и время посещения, количество гостей. |
| ***Вводимые данные*** |
| Дата и время формирования: 01.03.2024 10:10:10, Стол: ОБ2, Посетитель: IvanovII, дата и время посещения: 02.03.2024 11:11:00, количество: 1 |
| ***Ожидаемый результат*** |
| БР/24/0000000004 |
|  | ***Краткое описание теста*** |
| Удаление существующего ингредиента, у которого есть блюдо |
| ***Поля ввода*** |
| Код-ключ ингредиента |
| ***Вводимые данные*** |
| Ингредиент: Перец |
| ***Ожидаемый результат*** |
| Выбранный ингредиент не возможно удалить, так как к нему привязано блюдо. |
|  | ***Краткое описание теста*** |
| Проверка на ввод существующего ингредиента к существующему блюду. |
| ***Поля ввода*** |
| Код-ключ блюда, Код-ключ ингредиента. |
| ***Вводимые данные*** |
| Блюдо: Суп мечты, Ингредиент: Огурцы. |
| ***Ожидаемый результат*** |
| Указанный ингредиент уже есть у указанного блюда. |
|  | ***Краткое описание теста*** |
| Перерасчёт стоимости существующего заказа с добавлением новой позиции |
| ***Поля ввода*** |
| Код-Ключ заказа, Код-Ключ блюда, количество позиции |
| ***Вводимые данные*** |
| Заказ: ЗКЗ-000000001-23, Блюдо: «Как бы здоровое питание», количество: 1. |
| ***Ожидаемый результат*** |
| 3990 р. |

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Сценария** | Скрипт | Результат | Статус тестирования |
|  | create or replace procedure wood\_tables\_Insert(p\_table\_placescount int, p\_wood\_table\_name Varchar(4),  p\_zona\_kod int)  language plpgsql  as $$  begin  insert into wood\_tables(table\_placescount, wood\_table\_name, zona\_kod)  values (p\_table\_placescount, p\_wood\_table\_name, p\_zona\_kod);  exception when others then  raise notice 'Такой стол уже существует в таблице';  end;  $$; |  | Пройден |
|  | call booking\_insert('2024.01.03', '10:10:10', '02.03.2024', '11:11:00', 1,1, 'зп1');  create procedure booking\_insert(p\_book\_making\_date date,  p\_book\_making\_time time, p\_book\_plan\_date date,  p\_book\_plan\_time time, p\_book\_guests\_count int, p\_wood\_table\_kod int, p\_v\_kod varchar(5))  language plpgsql  as $$  declare p\_new\_book\_number text:= 1 + count(\*) from booking;  declare p\_new\_book\_year text:= right(left(p\_book\_plan\_date::text, 4),2);  declare p\_loop\_max\_length int= 10 - length(p\_new\_book\_number);  declare p\_new\_number varchar(10) := '';  declare p\_booking\_number text = '';  begin  for i in 1..p\_loop\_max\_length loop  p\_new\_number := p\_new\_number||'0';  end loop;  p\_booking\_number := 'БР/'||p\_new\_book\_year||'/'||p\_new\_number||p\_new\_book\_number;    insert into booking(book\_number,book\_making\_date,book\_making\_time,book\_plan\_date,  book\_plan\_time,book\_guests\_count,wood\_table\_kod,v\_kod)  values(p\_booking\_number, p\_book\_making\_date,  p\_book\_making\_time,p\_book\_plan\_date,  p\_book\_plan\_time, p\_book\_guests\_count,p\_wood\_table\_kod, p\_v\_kod);  end;  $$; |  | Пройден |
|  | create or replace procedure ingredientes\_delete(p\_ingredient\_kod int)  language plpgsql  as $$  declare p\_count\_of\_usage int = count(\*) from sostav  where ingredient\_kod = p\_ingredient\_kod;  begin  if (p\_count\_of\_usage > 0) then  raise notice 'Данный ингредиент уже используется';  else  delete from ingredientes  where  ingredient\_kod = p\_ingredient\_kod;  end if;  end;  $$;  call ingredientes\_delete(1); |  | Пройден |
|  | create or replace procedure sostav\_Insert(p\_ingredient\_kod int, p\_dish\_kod int)  language plpgsql  as $$  declare count\_of\_usage int = count(\*) from sostav  where  ingredient\_kod = p\_ingredient\_kod and  dish\_kod = p\_dish\_kod;  begin  if (count\_of\_usage > 0) then  raise notice 'такой ингредмент уже есть у блюда';  else  insert into sostav(ingredient\_kod, dish\_kod)  values (p\_ingredient\_kod, p\_dish\_kod);  end if;  end;  $$;  call sostav\_insert(1,1); |  | Пройден |
|  | create or replace procedure orders\_Insert(p\_order\_name Varchar(15), p\_order\_dishcount int,  p\_order\_dishtime time, p\_order\_date date,  p\_order\_opentime time, p\_table\_kod int,p\_empoyee\_kod int,  p\_status\_kod int, p\_dish\_kod Int)  language plpgsql  as $$  declare total\_cost decimal(7,2)= sum(dish\_cost \* p\_order\_dishcount) from orders  join dishes on dishes.dish\_kod = p\_dish\_kod  where order\_name = p\_order\_name;  begin  update checkues set  checkue\_sumfinal = total\_cost  where checkues.order\_name = p\_order\_name;  insert into orders(order\_name, order\_dishcount, order\_dishtime, order\_date, order\_opentime, table\_kod,employee\_kod,  status\_kod, dish\_kod)  values (p\_order\_name, p\_order\_dishcount,  p\_order\_dishtime, p\_order\_date,  p\_order\_opentime, p\_table\_kod,p\_empoyee\_kod,  p\_status\_kod, p\_dish\_kod);  end;  $$;  call orders\_insert('ЗКЗ-000000001-23', 2, '14:05:01', '2023.09.01', '14:00:24', 3, 1, 1, 1); |  | Пройден |

Как мне обратться к другой таблице, 3 процедура, в пятой не понятно где это должно происходить, разница между as и begin

|  |  |  |
| --- | --- | --- |
|  | create or replace procedure Checkue\_Insert(p\_Checkue\_number Varchar(13), p\_Checkue\_date Date, p\_Checkue\_time Time,  p\_Checkue\_sumFinal Int, p\_Checkue\_sumGotten Int, p\_Checkue\_changeSum Int, p\_Checkue\_PayType Varchar(50),  p\_Checkue\_finalCost Int, p\_Order\_kod Int)  language plpgsql  as $$  declare metteng\_count int := count(\*) from checkues where checkue\_number = p\_Checkue\_number;  begin  if (metteng\_count > 0) then  raise notice 'such check is already in base';  else  insert into checkues(Checkue\_number, Checkue\_date, Checkue\_time,  Checkue\_sumFinal, Checkue\_sumGotten, Checkue\_changeSum, Checkue\_PayType,  Checkue\_finalCost, Order\_kod)  values (p\_Checkue\_number, p\_Checkue\_date, p\_Checkue\_time,  p\_Checkue\_sumFinal, p\_Checkue\_sumGotten, p\_Checkue\_changeSum, p\_Checkue\_PayType,  p\_Checkue\_finalCost, p\_Order\_kod);  end if;  end;  $$; |  |
|  | create or replace procedure Dishes\_Insert(p\_dish\_name Varchar(50), p\_dish\_cost int, p\_dish\_weight Decimal(4,2), p\_dish\_picture Varchar(1))  language plpgsql  as $$  declare meeting\_count int := count(\*) from dishes where dish\_name = p\_dish\_name;  begin  if (meeting\_count > 0) then  raise notice 'such dish is already it table';  else  insert into dishes(dish\_name, dish\_cost, dish\_weight, dish\_picture)  values (p\_dish\_name, p\_dish\_cost, p\_dish\_weight, p\_dish\_picture);  end if;  end;  $$; |  |
|  | create or replace procedure Dishes\_Delete(p\_dish\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from orders where orders.dish\_kod = p\_dish\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such dish is used';  else  delete from dishes  where  dish\_kod = p\_dish\_kod;  end if;  end;  $$; |
|  | create or replace procedure Employee\_Insert(p\_employee\_surname Varchar(50),  p\_employee\_name Varchar(50), p\_employee\_patronymic Varchar(50), p\_employee\_login Varchar(50), p\_employee\_password Varchar(50))  language plpgsql  as $$  declare meeting\_count int := count(\*) from Employee where employee\_login = p\_employee\_login;  begin  if (meeting\_count > 0) then  raise notice 'such dish is used';  else  insert into employee(employee\_surname,  employee\_name,employee\_patronymic, employee\_login, employee\_password)  values (p\_employee\_surname,  p\_employee\_name,p\_employee\_patronymic, p\_employee\_login, p\_employee\_password);  end if;  end;  $$; |  |
|  | create or replace procedure Employee\_Delete(p\_employee\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from orders where orders.employee\_login = p\_employee\_login;  begin  if (meeting\_count > 0) then  raise notice 'employee is working lol';  else  delete from employee  where  employee\_kod = p\_employee\_kod;  end if;  end;  $$; |
|  | create or replace procedure individualorders\_Insert(p\_order\_kod int, p\_v\_kod varchar(5),  p\_person\_cost int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from individualorders\_Insert  where order\_kod = p\_order\_kod and \_v\_kod = p\_v\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such relation is made';  else  insert into individualorders(order\_kod,  v\_kod, person\_cost)  values (p\_order\_kod, p\_v\_kod, p\_person\_cost);  end if;  end;  $$; |  |
|  | create or replace procedure orders\_Delete(p\_order\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from checkues where checkues.order\_kod = p\_order\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such order is in documentation';  else  delete from orders  where  order\_kod = p\_order\_kod;  end if;  end;  $$; |  |
|  | create or replace procedure providers\_Insert(p\_provider\_city Varchar(50), p\_provider\_street Varchar(50),  p\_provider\_house int, p\_provider\_flatnumber int,  p\_provider\_name Varchar(50), p\_provider\_okpo Varchar(50))  language plpgsql  as $$  declare meeting\_count int := count(\*) from providers where provider\_name = p\_provider\_name;  begin  if (meeting\_count > 0) then  raise notice 'such provider if made';  else  insert into providers(provider\_city, provider\_street, provider\_house, provider\_flatnumber,  provider\_name, provider\_okpo)  values (p\_provider\_city, p\_provider\_street, p\_provider\_house, p\_provider\_flatnumber,  p\_provider\_name, p\_provider\_okpo);  end if;  end;  $$; |  |
|  | create or replace procedure providers\_Delete(p\_provider\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from shipments where shipments.provider\_kod = p\_provider\_kod;  begin  if (meeting\_count > 0) then  raise notice 'provider is used';  else  delete from providers  where  provider\_kod = p\_provider\_kod;  end if;  end;  $$; |
|  | create or replace procedure purchaseingredient\_Insert(p\_ingredient\_weight int, p\_shipment\_kod int,  p\_ingredient\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from purchaseingredient  where shipment\_kod = p\_shipment\_kod and ingredient\_kod = p\_ingredient\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such purchase is made';  else  insert into purchaseingredient(ingredient\_weight,shipment\_kod,ingredient\_kod)  values (p\_ingredient\_weight, p\_shipment\_kod, p\_ingredient\_kod);  end if;  end;  $$; |  |
|  | create or replace procedure registered\_visitors\_Insert(p\_rv\_surname Varchar(50), p\_rv\_name Varchar(50),  p\_rv\_patronymic Varchar(50), p\_rv\_login Varchar(50),  p\_rv\_password Varchar(50), p\_rv\_pasports Varchar(5),  p\_rv\_pasportn Varchar(6), p\_rv\_card Varchar(19), p\_rv\_kod varchar(5))  language plpgsql  as $$  declare meeting\_count int := count(\*) from registered\_visitors where rv\_login = p\_rv\_login;  begin  if (meeting\_count > 0) then  raise notice 'such person is added';  else  insert into registered\_visitors(rv\_surname, rv\_name,  rv\_patronymic, rv\_login,  rv\_password, rv\_pasports,  rv\_pasportn, rv\_card, rv\_kod)  values (p\_rv\_surname, p\_rv\_name,  p\_rv\_patronymic, p\_rv\_login,  p\_rv\_password, p\_rv\_pasports,  p\_rv\_pasportn, p\_rv\_card, p\_rv\_kod);  end if;  end;  $$; |  |
|  | create or replace procedure shipments\_Insert(p\_smeta\_kod int, p\_provider\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from shipments  where smeta\_kod = p\_smeta\_kod and provider\_kod = p\_provider\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such shipment is made';  else  insert into shipments(smeta\_kod, provider\_kod)  values (p\_smeta\_kod, p\_provider\_kod);  end if;  end;  $$; |  |
|  | create or replace procedure shipments\_Delete(p\_shipment\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from purchaseingredient where shipment\_kod = p\_shipment\_kod;  begin  if (meeting\_count > 0) then  raise notice 'shipment is used';  else  delete from shipments  where  shipment\_kod = p\_shipment\_kod;  end if;  end;  $$; |
|  | create or replace procedure smeta\_Insert(p\_smeta\_date date, p\_smeta\_number Varchar(13))  language plpgsql  as $$  declare meeting\_count int := count(\*) from smeta where smeta\_number = p\_smeta\_number;  begin  if (meeting\_count > 0) then  raise notice 'such smeta is made';  else  insert into smeta(smeta\_date, smeta\_number)  values (p\_smeta\_date, p\_smeta\_number);  end if;  end;  $$; |  |
|  | create or replace procedure smeta\_Delete(p\_smeta\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from shipments where smeta\_kod = p\_smeta\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such smeta is used';  else  delete from smeta  where  smeta\_kod = p\_smeta\_kod;  end if;  end;  $$; |
|  | create or replace procedure status\_Insert(p\_status\_value varchar(50))  language plpgsql  as $$  declare meeting\_count int := count(\*) from status where status\_value = p\_status\_value;  begin  if (meeting\_count > 0) then  raise notice 'such dish is used';  else  insert into status(status\_value)  values (p\_status\_value);  end if;  end;  $$; |  |
|  | create or replace procedure status\_Delete(p\_status\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from orders where status\_kod = p\_status\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such status is used';  else  delete from status  where  status\_kod = p\_status\_kod;  end if;  end;  $$; |
|  | create or replace procedure visitors\_Insert(p\_v\_surname varchar(50), p\_v\_name varchar(50),  p\_v\_patronymic varchar(50), p\_v\_kod varchar(5))  language plpgsql  as $$  declare meeting\_count int := count(\*) from visitors where v\_kod = p\_v\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such visitor is made';  else  insert into visitors(v\_surname, v\_name, v\_patronymic, v\_kod)  values (p\_v\_surname, p\_v\_name, p\_v\_patronymic, p\_v\_kod);  end if;  end;  $$; |  |
|  | create or replace procedure visitors\_Delete(p\_v\_kod varchar(5))  language plpgsql  as $$  declare meeting\_count int := count(\*) from individualorders where v\_kod = p\_v\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such visitor is used';  else  delete from visitors  where  v\_kod = p\_v\_kod;  end if;  end;  $$; |
|  | create or replace procedure wood\_tables\_Delete(p\_table\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from orders where table\_kod = p\_table\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such table is used';  else  delete from wood\_tables  where  table\_kod = p\_table\_kod;  end if;  end;  $$; |  |
|  | create or replace procedure zones\_Insert(p\_zona\_name varchar(50))  language plpgsql  as $$  declare meeting\_count int := count(\*) from zones where zona\_name = p\_zona\_name;  begin  if (meeting\_count > 0) then  raise notice 'such zone is made';  else  insert into zones(zona\_name)  values (p\_zona\_name);  end if;  end;  $$; |  |
|  | create or replace procedure zones\_Delete(p\_zona\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from wood\_tables where p\_zona\_kod = p\_zona\_kod;  begin  if (meeting\_count > 0) then  raise notice 'such zone is used';  else  delete from zones  where  zona\_kod = p\_zona\_kod;  end if;  end;  $$; |
|  | create procedure booking\_update(p\_booking\_number Varchar(17), p\_book\_making\_date date,  p\_book\_making\_time time, p\_book\_plan\_date date,  p\_book\_plan\_time time, p\_book\_guests\_count int, p\_wood\_table\_kod int, p\_v\_kod varchar(5))  language plpgsql  as $$  begin  update booking set  book\_making\_date = p\_book\_making\_date,  book\_making\_time = p\_book\_making\_time,  book\_plan\_date = p\_book\_plan\_date,  book\_plan\_time = p\_book\_plan\_time,  book\_guests\_count = p\_book\_guests\_count,  wood\_table\_kod = p\_wood\_table\_kod,  v\_kod = p\_v\_kod  where book\_number = p\_booking\_number;    end;  $$; |  |
|  | create or replace procedure booking\_update(p\_booking\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from planning\_orders where planning\_orders.book\_kod = p\_booking\_kod ;  begin  if (p\_booking\_number > 0) then  raise notice 'such booking is used';  else  update booking set  book\_making\_date = p\_book\_making\_date,  book\_making\_time = p\_book\_making\_time,  book\_plan\_date = p\_book\_plan\_date,  book\_plan\_time = p\_book\_plan\_time,  book\_guests\_count = p\_book\_guests\_count,  wood\_table\_kod = p\_wood\_table\_kod,  v\_kod = p\_v\_kod;  where book\_number = p\_booking\_number;  end if;  end;  $$; |  |
|  | create procedure booking\_delete(p\_booking\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from planning\_orders where book\_kod = p\_booking\_kod;  begin  if (meeting\_count > 0) then  raise notice 'this booking is used';  else  delete from booking  where booking\_kod = p\_booking\_kod;  end if;  end;  $$; |  |
|  | create procedure planning\_orders\_insert(p\_order\_kod int, p\_book\_kod int)  language plpgsql  as $$  declare meeting\_count int := count(\*) from planning\_orders  where order\_kod = p\_order\_kod and book\_kod = p\_book\_kod;  begin  if (meeting\_count > 0) then  raise notice 'this plan is made';  else  insert into booking(order\_kod, book\_kod)  values (p\_order\_kod, p\_book\_kod);  end if;  end;  $$; |  |
|  | create procedure planning\_orders\_update(p\_order\_kod int, p\_book\_kod int, p\_planning\_order\_kod int)  language plpgsql  as $$  begin  update planning\_orders set  order\_kod = p\_order\_kod,  book\_kod = p\_book\_kod,  planning\_order\_kod = p\_planning\_order\_kod  where planning\_order\_kod = p\_planning\_order\_kod;  end;  $$; |
|  | create procedure planning\_orders\_delete(p\_planning\_order\_kod int)  language plpgsql  as $$  begin  delete from planning\_orders  where planning\_order\_kod = p\_planning\_order\_kod;  end;  $$; |
|  | create or replace procedure planning\_dishes\_delete(p\_planning\_dishes\_kod int)  language plpgsql  as $$  begin  delete from planning\_dishes  where planning\_dishes\_kod = p\_planning\_dishes\_kod;  end;  $$; |
|  | create procedure planning\_dishes\_update(p\_planning\_dishes\_time time, p\_dish\_kod int,  p\_book\_kod int, p\_planning\_dishes\_kod int)  language plpgsql  as $$  begin  update planning\_dishes set  planning\_dishes\_time = p\_planning\_dishes\_time,  dish\_kod = p\_dish\_kod,  book\_kod = p\_book\_kod  where planning\_dishes\_kod = p\_planning\_dishes\_kod;  end;  $$; |  |
|  | create procedure planning\_dishes\_insert(p\_planning\_dishes\_time time, p\_dish\_kod int, p\_book\_kod int)  language plpgsql  as $$  begin  insert into planning\_dishes(planning\_dishes\_time, dish\_kod, book\_kod)  values (p\_planning\_dishes\_time, p\_dish\_kod, p\_book\_kod);  end;  $$; |

| Роли | | Официант | Зарегистрированный посетитель | Посетитель | Шеф | Администратор |
| --- | --- | --- | --- | --- | --- | --- |
| Название объекта | Функции |
| Checkue\_Insert | Вызов | X |  |  |  |  |
| Checkues\_update | Вызов | X |  |  |  |  |
| Checkues\_Delete | Вызов | X |  |  |  |  |
| Dishes\_Insert | Вызов |  |  |  | X |  |
| Dishes\_Update | Вызов |  |  |  | X |  |
| Dishes\_Delete | Вызов |  |  |  | X |  |
| Employee\_Insert | Вызов |  |  |  | X | X |
| Employee\_Update | Вызов | X |  |  | X | X |
| Employee\_Delete | Вызов |  |  |  |  | X |
| individualorders\_Insert | Вызов | X |  |  |  |  |
| individualorders\_Update | Вызов | X |  |  |  |  |
| individualorders\_Delete | Вызов | X |  |  |  |  |
| ingredientes\_Insert | Вызов |  |  |  | X |  |
| ingredientes\_Update | Вызов |  |  |  | X |  |
| ingredientes\_Delete | Вызов |  |  |  | X |  |
| orders\_Insert | Вызов | X |  |  |  |  |
| orders\_Update | Вызов | X |  |  |  |  |
| orders\_Delete | Вызов | X |  |  |  |  |
| providers\_Insert | Вызов |  |  |  | X | X |
| providers\_Update | Вызов |  |  |  | X | X |
| providers\_Delete | Вызов |  |  |  | X | X |
| purchaseingredient\_Insert | Вызов |  |  |  | X | X |
| purchaseingredient\_Update | Вызов |  |  |  | X |  |
| purchaseingredient\_Delete | Вызов |  |  |  | X |  |
| registered\_visitors\_Insert | Вызов | X |  |  | X | X |
| registered\_visitors\_Update | Вызов | X | X |  |  | X |
| registered\_visitors\_Delete | Вызов |  |  |  |  | X |
| shipments\_Insert | Вызов |  |  |  | X |  |
| shipments\_Update | Вызов |  |  |  | X |  |
| shipments\_Delete | Вызов |  |  |  | X |  |
| smeta\_Insert | Вызов |  |  |  | X | X |
| smeta\_Update | Вызов |  |  |  | X | X |
| smeta\_Delete | Вызов |  |  |  | X | X |
| sostav\_Insert | Вызов |  |  |  | X |  |
| sostav\_Update | Вызов |  |  |  | X |  |
| sostav\_Delete | Вызов |  |  |  | X |  |
| status\_Insert | Вызов | X |  |  |  |  |
| status\_Update | Вызов | X |  |  |  |  |
| status\_Delete | Вызов | X |  |  |  |  |
| visitors\_Insert | Вызов | X |  |  |  | X |
| visitors\_Update | Вызов | X |  | X |  | X |
| visitors\_Delete | Вызов |  |  |  |  | X |
| wood\_tables\_Insert | Вызов |  |  |  |  | X |
| wood\_tables\_Update | Вызов |  |  |  |  | X |
| wood\_tables\_Delete | Вызов |  |  |  |  | X |
| zones\_Insert | Вызов |  |  |  |  | X |
| zones\_Update | Вызов |  |  |  |  | X |
| zones\_Delete | Вызов |  |  |  |  | X |
| Booking\_insert | Вызов |  | X | X |  | X |
| Booking\_update | Вызов |  | X | X |  | X |
| Booking\_delete | Вызов |  | X | X |  | X |
| planning\_orders\_insert | Вызов |  | X | X |  |  |
| planning\_orders\_delete | Вызов |  | X | X |  |  |
| planning\_orders\_upadete | Вызов |  | X | X |  |  |
| planning\_dishes\_insert | Вызов |  |  |  |  |  |
| planning\_dishes\_update | Вызов |  |  |  |  |  |
| planning\_dishes\_delete | Вызов |  |  |  |  |  |

| Название роли | Название объекта | Функция | Скрипт |
| --- | --- | --- | --- |
| Res\_waiter | status\_Insert | Execute | grant execute on procedure status\_Insert to res\_waiter; |
| status\_Update | grant execute on procedure status\_Update to res\_waiter; |
| status\_Delete | grant execute on procedure status\_Delete to res\_waiter; |
| visitors\_Insert | grant execute on procedure visitors\_Insert to res\_waiter; |
| registered\_visitors\_Insert | grant execute on procedure registered\_visitors\_Insert to res\_waiter; |
| registered\_visitors\_Update | grant execute on procedure registered\_visitors\_Update to res\_waiter; |
| orders\_Insert | grant execute on procedure orders\_Insert to res\_waiter; |
| orders\_Update | grant execute on procedure orders\_Update to res\_waiter; |
| orders\_Delete | grant execute on procedure orders\_Delete to res\_waiter; |
| individualorders\_Insert | grant execute on procedure individualorders\_Insert to res\_waiter; |
| individualorders\_Update | grant execute on procedure individualorders\_Update to res\_waiter; |
| individualorders\_Delete | grant execute on procedure individualorders\_Delete to res\_waiter; |
| Checkues\_Delete | grant execute on procedure Checkues\_Delete to res\_waiter; |
| Employee\_Update | grant execute on procedure Employee\_Update to res\_waiter; |
| Res\_guest | registered\_visitors\_Update | Execute | grant execute on procedure registered\_visitors\_Update to res\_guest; |
| visitors\_Update | grant execute on procedure booking\_insert to res\_guest; |
| booking \_insert | grant execute on procedure booking\_delete to res\_guest; |
| booking \_delete | grant execute on procedure booking\_update to res\_guest; |
| booking \_upadete | grant execute on procedure planning\_dishes\_insert to res\_guest; |
| planning\_dishes\_insert | grant execute on procedure planning\_dishes\_update to res\_guest; |
| planning\_dishes\_update |
| planning\_dishes\_delete | grant execute on procedure planning\_dishes\_delete to res\_guest; |
| Res\_dguest | visitors\_Update | Execute | grant execute on procedure visitors\_Update to res\_guest; |
| booking \_insert | grant execute on procedure booking\_insert to res\_dguest; |
| booking \_delete | grant execute on procedure booking\_delete to res\_dguest; |
| booking \_upadete | grant execute on procedure booking\_update to res\_dguest; |
| planning\_dishes\_insert | grant execute on procedure planning\_dishes\_insert to res\_dguest; |
| planning\_dishes\_update | grant execute on procedure planning\_dishes\_update to res\_dguest; |
| planning\_dishes\_delete | grant execute on procedure planning\_dishes\_delete to res\_dguest; |
| Res\_chef | shipments\_Insert | Execute | grant execute on procedure shipments\_Insert to res\_chef; |
| shipments\_Update | grant execute on procedure shipments\_Update to res\_chef; |
| shipments\_Delete | grant execute on procedure shipments\_Delete to res\_chef; |
| smeta\_Insert | grant execute on procedure smeta\_Insert to res\_chef; |
| smeta\_Update | grant execute on procedure smeta\_Update to res\_chef; |
| smeta\_Delete | grant execute on procedure smeta\_Delete to res\_chef; |
| sostav\_Insert | grant execute on procedure sostav\_Insert to res\_chef; |
| sostav\_Update | grant execute on procedure sostav\_Update to res\_chef; |
| sostav\_Delete | grant execute on procedure sostav\_Delete to res\_chef; |
| providers\_Insert | grant execute on procedure providers\_Insert to res\_chef; |
| providers\_Update | grant execute on procedure providers\_Update to res\_chef; |
| providers\_Delete | grant execute on procedure providers\_Delete to res\_chef; |
| purchaseingredient\_Insert | grant execute on procedure purchaseingredient\_Insert to res\_chef; |
| purchaseingredient\_Update | grant execute on procedure purchaseingredient\_Update to res\_chef; |
| purchaseingredient\_Delete | grant execute on procedure purchaseingredient\_Delete to res\_chef; |
| registered\_visitors\_Insert | grant execute on procedure registered\_visitors\_Insert to res\_chef; |
| ingredientes\_Insert | grant execute on procedure ingredientes\_Insert to res\_chef; |
| ingredientes\_Update | grant execute on procedure ingredientes\_Update to res\_chef; |
| ingredientes\_Delete | grant execute on procedure ingredientes\_Delete to res\_chef; |
| Employee\_Insert | grant execute on procedure Employee\_Insert to res\_chef; |
| Employee\_Update | grant execute on procedure Employee\_Update to res\_chef; |
| Dishes\_Insert | grant execute on procedure Dishes\_Insert to res\_chef; |
| Dishes\_Update | grant execute on procedure Dishes\_Update to res\_chef; |
| Dishes\_Delete | grant execute on procedure Dishes\_Delete to res\_chef; |
| Res\_administrator | visitors\_Insert | execute | grant execute on procedure visitors\_Insert to res\_administrator; |
| visitors\_Update | grant execute on procedure visitors\_Update to res\_administrator; |
| visitors\_Delete | grant execute on procedure visitors\_Delete to res\_administrator; |
| wood\_tables\_Insert | grant execute on procedure wood\_tables\_Insert to res\_administrator; |
| wood\_tables\_Update | grant execute on procedure wood\_tables\_Update to res\_administrator; |
| wood\_tables\_Delete | grant execute on procedure wood\_tables\_Delete to res\_administrator; |
| zones\_Insert | grant execute on procedure zones\_Insert to res\_administrator; |
| zones\_Update | grant execute on procedure zones\_Update to res\_administrator; |
| zones\_Delete | grant execute on procedure zones\_Delete to res\_administrator; |
| smeta\_Insert | grant execute on procedure smeta\_Insert to res\_administrator; |
| smeta\_Update | grant execute on procedure smeta\_Update to res\_administrator; |
| smeta\_Delete | grant execute on procedure smeta\_Delete to res\_administrator; |
| registered\_visitors\_Insert | grant execute on procedure registered\_visitors\_Insert to res\_administrator; |
| registered\_visitors\_Update | grant execute on procedure registered\_visitors\_Update to res\_administrator; |
| registered\_visitors\_Delete | grant execute on procedure registered\_visitors\_Delete to res\_administrator; |
| providers\_Insert | grant execute on procedure providers\_Insert to res\_administrator; |
| providers\_Update | grant execute on procedure providers\_Update to res\_administrator; |
| providers\_Delete | grant execute on procedure providers\_Delete to res\_administrator; |
| purchaseingredient\_Insert | grant execute on procedure purchaseingredient\_Insert to res\_administrator; |
| Employee\_Insert | grant execute on procedure Employee\_Insert to res\_administrator; |
| Employee\_Update | grant execute on procedure Employee\_Update to res\_administrator; |
| Employee\_Delete | grant execute on procedure Employee\_Delete to res\_administrator; |
| Booking\_insert | grant execute on procedure booking\_insert to Res\_administrator; |
| Booking\_update | grant execute on procedure booking\_delete to Res\_administrator; |
| Booking\_delete | grant execute on procedure booking\_update to Res\_administrator; |



|  | 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 |  | |
| Параметры | | PostgreSQL |
| Номер версии | | 3.1.0.1 |
| Что сделано | | * Созданы 3 таблиц; * Созданы 17 столбцов; * Созданы 10 индексjd; * Созданы 9 хранимые процедуры; * Произведено распределение доступа ролей к таблицам и хранимым процедурам; * Создан Backup файл. |