Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: программист

Профессиональный модуль: ПМ 11 Разработка, администрирование и защита баз данных

Междисциплинарный курс: МДК 11.01 Технология разработки и защиты баз данных

МПТ.09.02.07-П.П50-11/3-20.02.21

Отчёт

Тема: «Разработка, базы данных на примере предметной области: «Велосипедный магазин», реализация подсистем: Список сотрудников, Отдел продаж»

|  |  |
| --- | --- |
| Проверил: | Выполнил: |
| Щаников И.М.\_\_\_\_\_\_\_\_\_\_ | Студент(ка) группы |
| (\_\_\_\_\_\_\_\_)\_\_\_\_\_\_\_\_\_\_\_\_\_ | П50-11/3-20 |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021г. | \_\_\_\_\_\_\_\_\_\_\_\_Вишняков Д.А. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021г. |

2021

СОДЕРЖАНИЕ

[1. ОПИСАНИЕ ПРОБЛЕМНОЙ ЗОНЫ 3](#_Toc81481601)

[1.1. Описание предметной области 3](#_Toc81481602)

[1.2. Описание подсистем 3](#_Toc81481603)

[1.3. Описание сущностей 3](#_Toc81481604)

[2. ПРОЕКТИРОВАНИЕ 4](#_Toc81481605)

[2.1. Модель данных 4](#_Toc81481606)

[2.2. Словарь данных 4](#_Toc81481607)

[3. РЕАЛИЗАЦИЯ 6](#_Toc81481608)

[3.1. PostgreSQL 6](#_Toc81481609)

[3.2. MySQL 7](#_Toc81481610)

[4. СРАВНЕНИЕ СУБД 10](#_Toc81481611)

[В данном разделе в виде таблицы необходимо представить, сравнение двух систем управления базами данных, с указанием всех достоинств, удобств и недостатков. 10](#_Toc81481612)

[Таблица 4.1 – Сравнение СУБД 10](#_Toc81481613)

[Достоинства 10](#_Toc81481614)

[Недостатки 10](#_Toc81481615)

[PostgreSQL 10](#_Toc81481616)

[MySQL 10](#_Toc81481617)

[PGAdmin 10](#_Toc81481618)

[phpMyAdmin 10](#_Toc81481619)

[MySQL Workbench 10](#_Toc81481620)

[5. ВЫВОД 11](#_Toc81481621)

[6. СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 12](#_Toc81481622)

[В данном пункте, необходимо нумерованным списком перечислить, все интернет-источники, литературу, стандарты, которые учувствовали в разработке двух баз данных: 12](#_Toc81481623)

[1) Название книги, Фамилия инициалы, издательство, год издания, количество страниц, URL-ссылка на ресурс; 12](#_Toc81481624)

[2) Номер стандарта, название стандарта, URL-ссылка на ресурс; 12](#_Toc81481625)

[3) Название сайта, краткое описание информации заимствованной с сайта, URL-ссылка на ресурс. 12](#_Toc81481626)

1. ОПИСАНИЕ ПРОБЛЕМНОЙ ЗОНЫ
   1. Описание предметной области

В данном подразделе необходимо описать основную деятельность предприятия, а именно: кто является потребителем предполагаемой услуги, кто выполняет услуги, чем услуги регламентируются, какие документы получаются на выходе. Так же в описании необходимо указать, каким образом взаимосвязаны выбранные подсистемы, какими объектами и данными они взаимосвязаны.

Предметная область: велосипедный магазин  
Потребитель – физические лица и юридическое лица  
Документ на вход – запрос на приобретение техники, обрабатываемый менеджером, который составляет чек оплаты покупки техники.  
Документы на выход – гарантия, чек.

* 1. Описание подсистем

В данном подразделе, на основании описания предметной области и выбранных подсистем, необходимо описать функционирование подсистем, с тем же подходом, что и описание предметной области, только с большей детализацией. Описание не предполагает, наличие в описание абстрактных объектов, или серых действий, описание должно быть прозрачным, полноценным и ясным.

Подсистемы: список сотрудников, отдел продаж.  
 Отдел продаж – менеджер составляет с поставщиком запрос на приобретение техники по оптовой цене(наименование товара кол-во товара, цена) далее совершается закупка техники, после чего она обрабатывается поставщиком, и на выходе получается накладная, в котором указано: ФИО поставщика, цена накладной, счет поставщика, ИНН организации.  
Список сотрудников – обрабатывает запрос на приобретение техники, поиск остатка на складах, передает запрос поставщикам, после чего составляется накладная с поставщиком, далее товар продается клиенту по розничной цене и составляется чек ФИО продавца, цена техники, счет организации, номер карты клиента, ИНН организации.

* 1. Описание сущностей

В данном подразделе, на основании описания подсистем предметной области, из описания, необходимо выявить, существительные и глаголы, которые лягут в основу будущих сущностей. Описание сущностей идёт маркированным списком, с указанием названия сущности и через, тире, перечисление атрибутов, которые могут быть описаны, без правил нормализаций отношений в баз данных.

Личное дело сотрудника – ФИО, дата рождения, серия паспорта, номер паспорта, номер личного дела.

Чек – номер чека, дата печати чека, ИНН чека

Товар – наименование товара

Вид товара – наименование вида

Должность – наименование должности, оклад

Заказ – номер заказа, наименование товара, кол-во товара, цена товара

Расчёт заработной платы – дата выдачи зарплаты, премия, отпускные

Поставщик – компания, номер телефона, почта

1. ПРОЕКТИРОВАНИЕ
   1. Модель данных
      1. Инфологическая модель данных

В данном пункте, необходимо дать иллюстрацию логической модели данных, на основе, выявленных сущностей и атрибутов с учётом правил 3-ей нормальной формы нормализации отношений.

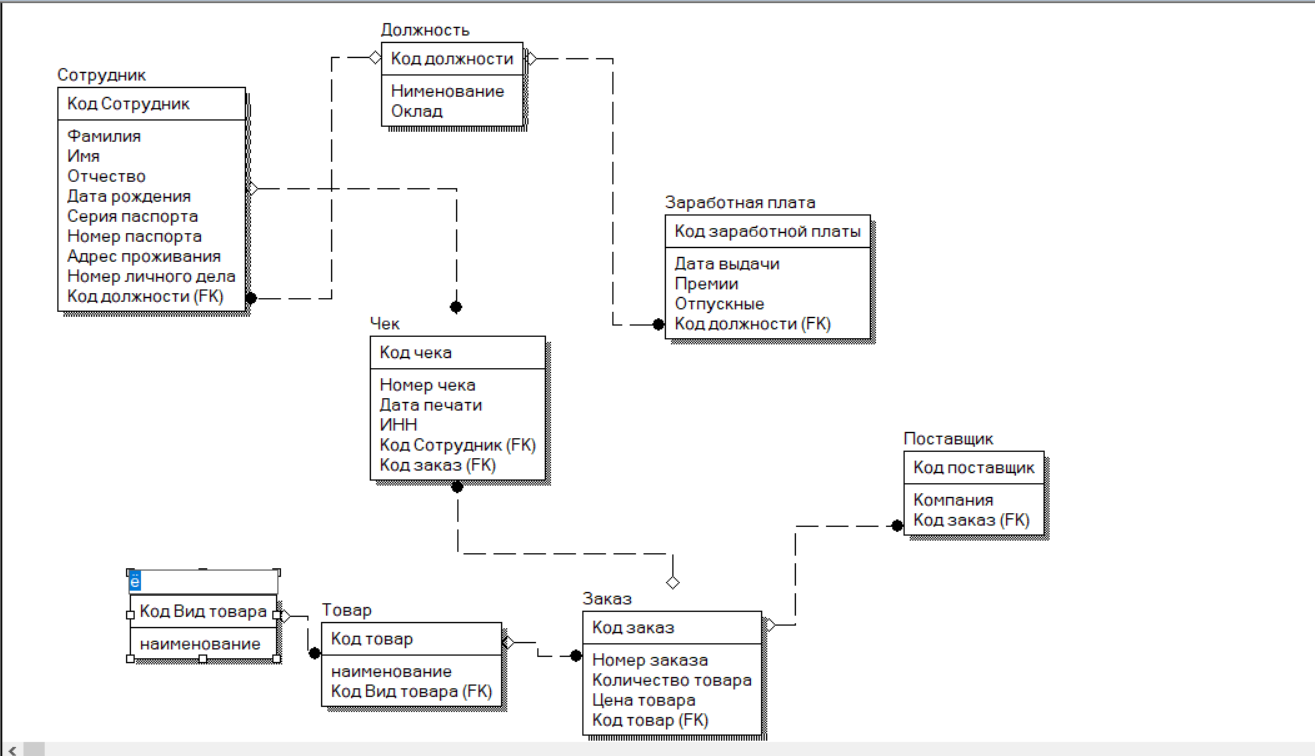


Рисунок 1 Инфологическая модель данных

* + 1. Дата логическая модель данных

В данном пункте, необходимо, дать иллюстрацию физической модели данных, на основе логической модели данных, с указанием типов данных и обязательности заполнения полей.

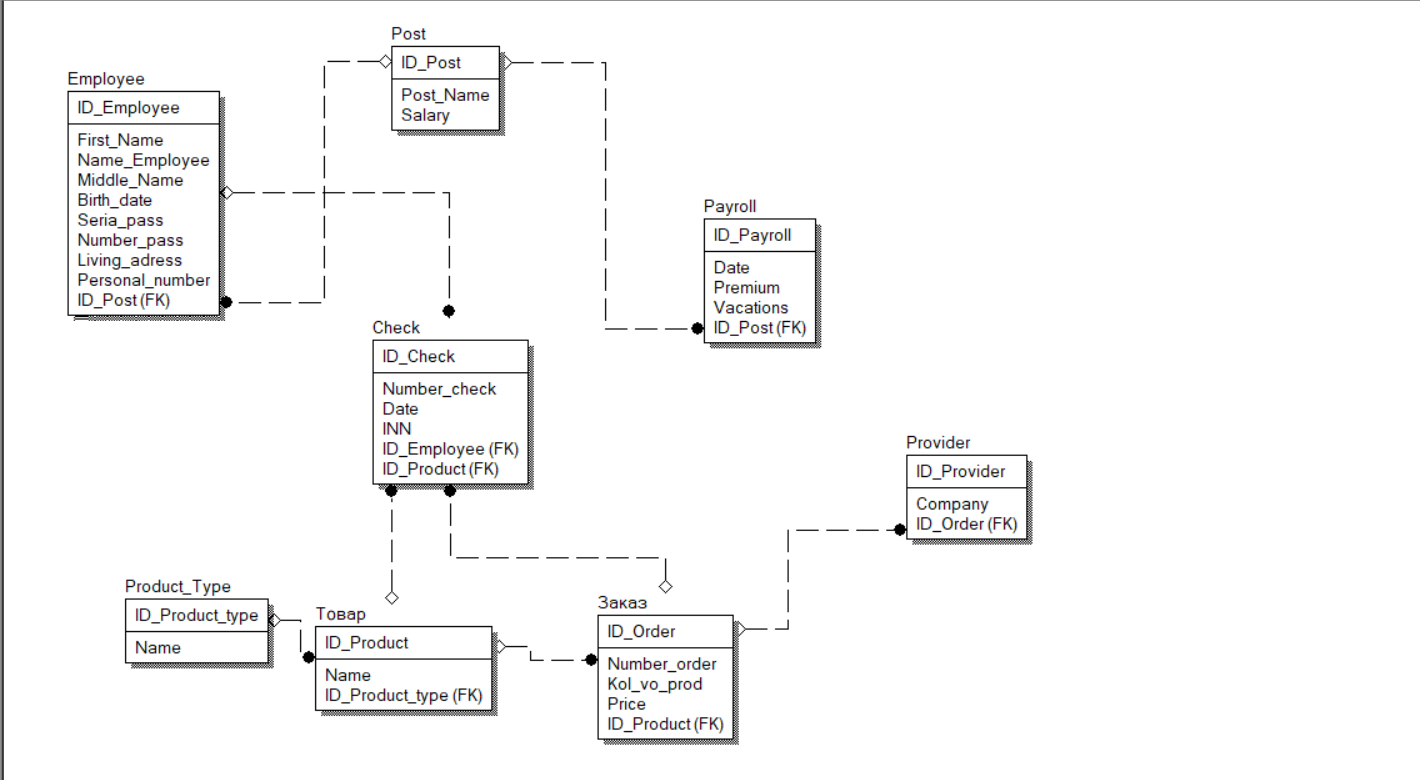


Рисунок 2 Логическая модель данных

* 1. Словарь данных
     1. Входные данные

В данном пункте, на основании логической и физической модели данных, необходимо в табличном виде представить описание всех входных данных, с указанием: логического и физического названия поля, типа данных со стороны СУБД, ограничений на поля (первичный или внешний ключ, уникальное поле, проверка на соответствие выполнения условий: масок ввода, на соответствие дат, числовых показателей и т.д.).

Таблица 2.1 – Входные данные

| Логическое название | Физическое название | Тип данных | Ограничения |
| --- | --- | --- | --- |
| Employee | | | |
| Код Сотрудника | ID\_Employee | int | Not null, PK |
| Фамилия | First\_Name | Varchar(30) | Not null, check (%[а-яА-Я]%) |
| Имя | Name\_Employee | Varchar(30) | Not null, check (%[а-яА-Я]%) |
| Отчество | Middle\_Name | Varchar(30) | Null, check (%[а-яА-Я]%), default('-') |
| Дата рождения | Date\_of\_birth | date | Not null |
| Серия паспорта | Passport\_seria | Varchar(4) | Not null, check ([0-9]{4}) |
| Номер паспорта | Passport\_Number | Varchar(6) | Not null, check ([0-9]{6}) |
| Адрес проживания | Living\_address | Varchar(30) | Not null |
| Номер личного дела | Personal\_business\_number | Varchar(5) | Not null, unique, check ([0-9]{5}) |
| Код Должности | ID\_Post | int | Not null, FK |
| Post | | | |
| Код Должности | ID\_Post | int | Not null, PK |
| Наименование | Post\_Name | Varchar(30) | Not null, unique, check (%[а-яА-Я]%) |
| Оклад | Salary | Decimal(38,2) | Null, check (>=0), default(0.0) |
| Check | | | |
| Код Чека | ID\_Check | int | Not null, PK |
| Номер чека | Check\_number | Varchar(10) | Not null, unique, check ([0-9]{10}) |
| Дата печати чека | Date\_of\_printing\_check | date | Not null |
| ИНН | INN | Varchar(10) | Not null, check ([0-9]{10}) |
| Код Сотрудника | ID\_Employee | int | Not null, FK |
| Код Заказа | ID\_Order | int | Not null, FK |
| Payroll\_preparation | | | |
| Код Расчёта заработной платы | ID\_Payroll\_preparation | int | Not null, PK |
| Дата выдачи зарплаты | Date\_of\_payroll | date | Not null |
| Премия | Premium | Decimal(38,2) | Null, check (>=0), default(0.0) |
| Отпускные | Vacations | Decimal(38,2) | Not null, check (>=0) |
| Код Должности | ID\_Post | int | Not null, FK |
| Provider | | | |
| Код Поставщика | ID\_Provider | int | Not null, PK |
| Компания | Company | Varchar(30) | Not null |
| Телефон | Telephone | Varchar(11) | Not null, check (+7([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]) |
| Mail | Почта | Varchar(30) | Null, check (‘%@%’), default (‘-‘) |
| Код Заказа | ID\_Order | int | Not null, FK |
| Product | | | |
| Код Товара | ID\_Product | int | Not null, PK |
| Наименование | Name\_of\_product | Varchar(30) | Not null, unique, check (%[а-яА-Я]%) |
| Код Вида товара | ID\_Product\_type | int | Not null, FK |
| Product\_type | | | |
| Код Вида товара | ID\_Product\_type | int | Not null, PK |
| Наименование вида | Name\_of\_type | Varchar(30) | Not null, unique, check (%[а-яА-Я]%) |
| Order | | | |
| Код Заказа | ID\_Order | int | Not null, PK |
| Номер заказа | Order\_number | Varchar(12) | Not null, unique, check ([0-9]{12}) |
| Кол-во товара | Number\_of\_product | int | Not null, check (%[1-9]%) |
| Цена товара | Product\_price | Decimal(38,2) | Not null, check (>=0) |
| Код Товара | ID\_Product | int | Not null, FK |

* + 1. Выходные данные

В данном пункте, на основании описания выходных документов из описания: предметной области и подсистем, необходимо указать в виде таблицы, название выходного документа, его атрибуты, и таблицы из которых будет формироваться документ.

Таблица 2.2 – Выходные данные

| Название документа | Поля | Исходные таблицы |
| --- | --- | --- |
| Чек | Номер чека, ФИО сотрудника | «Чек», «Сотрудник», «Должность», «Заказ» |
| Номер заказа |

* + 1. Статистическое проектирование

В данном пункте в виде таблицы, на основании описания подсистем, логической и физической модели данных, необходимо описать все возможные, статистические данные, которые могут описывать, количественные показатели, денежные показатели, среднестатистические данные и т.д., с полями: название функции, формальные параметры, формула, результат функции:

Таблица 2.3 – Описание функций

| Название функции | Формальные параметры | Формула | Результат функции |
| --- | --- | --- | --- |
| Заработная плата сотрудников за отработанное время | Период начала – Дата, период конца - Дата | Должность.Стоимость \* Совместительство.Ставка \* Сумму часов(Период начала, период конца) \* 0,87 | ЗП – вещественное число |
| Заработанная премия от оклада | Оклад | Оклад \* 30% | Заработанная премия от оклада – вещественное число |

1. РЕАЛИЗАЦИЯ
   1. PostgreSQL
      1. Реализация таблиц
         1. Скрипт базы данных

-- Создание таблицы "Сотрудник" с ограничениями на поле First\_Name, Name\_Employee, Middle\_Name

-- на ввод кириллицей

-- поле Passport\_seria, Passport\_number, Personal\_business\_number имеют ограничения на ввод цифр с условием VARCHAR до

-- макс. указанного значения

create table Employee

(

ID\_Employee serial not null constraint PK\_Employee primary key,

First\_Name varchar(30) not null constraint CH\_First\_Name

check (First\_Name similar to '%[а-яА-Я]%'),

Name\_Employee varchar(30) not null constraint CH\_Name\_Employee

check (Name\_Employee similar to '%[а-яА-Я]%'),

Middle\_Name varchar(30) null constraint CH\_Middle\_Name

check (Middle\_Name similar to '%[а-яА-Я]%')

default('-'),

Date\_of\_birth date not null,

Passport\_seria varchar(4) not null constraint CH\_Passport\_seria

check (Passport\_seria similar to '[0-9]{4}'),

Passport\_number varchar(6) not null constraint CH\_Passport\_number

check (Passport\_number similar to '[0-9]{6}'),

Living\_address varchar(30) not null,

Personal\_business\_number varchar(5) not null constraint CH\_Personal\_business\_number

check (Personal\_business\_number similar to '[0-9]{5}')

constraint UQ\_Personal\_business\_number unique,

Post\_ID int not null references Post (ID\_Post)

);

-- Добавление данных в таблицу "Сотрудник"

insert into employee(First\_Name,Name\_Employee,Middle\_Name,date\_of\_birth,passport\_seria,passport\_number,living\_address,

personal\_business\_number,post\_id)

values ('Маслов','Тимофей','Дмитриевич','1998-12-25','4567','114567','ул. Тушкина д.7','47389',1),

('Тихонова','Дарья','Тиграновна','1997-02-15','4527','199567','ул. Карлушкино д.10','90372',2),

('Фролов','Роман','Ярославович','1999-03-10','4541','178290','ул. Пушкино д.28У7','11923',3),

('Иванов','Тимофей','Артёмович','1999-07-25','4117','219291','ул. Корлонова д.37','73483',4),

('Акимов','Денис','Русланович','1987-08-14','4589','127867','ул. Фушкина д.12','73484',5),

('Гаврилов','Арсений','Денисович','1985-10-19','4997','192837','ул. Душкина д.73','73485',6),

('Зотова','Оливия','Дамировна','1983-11-15','4535','614567','ул. Корушкина д.17','73486',7),

('Михайлова ','Таисия','Давидовна','1998-08-25','4997','119877','ул. Лушкина д.97','73487',8),

('Суханова','Алиса','Романовна','1998-12-25','4567','114567','ул. Тушкина д.7','73489',9),

('Быков','Павел','Георгиевич','1998-12-25','4567','114567','ул. Тушкина д.7','73490',10);

select \* from employee;

-- Создание таблицы "Должность" с ограничениями на поле Post\_Name на ввод кириллицей и на ограничение

-- unique - проверка на ввод уникальных данных

-- поле Post\_Price имеет ограничение, что оно должно быть >= 0.0 и если null то возвращает значение 0.0

create table Post (

ID\_Post SERIAL not null constraint PK\_Post primary key,

Post\_Name varchar(50) not null constraint CH\_Post\_name

check (Post\_name similar to '%[а-яА-Я]%')

constraint UQ\_Post\_Name unique,

Post\_Price decimal(38,2) null constraint CH\_Post\_Price check

(Post\_Price >=0.0) default(0.0)

);

-- Занесение данных в таблицу "Должность"

insert into post(Post\_Name,Post\_Price)

values ('Главный Администратор',110000.00),

('Администратор',90000.00),

('Главный бухгалтер',80000.00),

('Бухгалтер',70000.00),

('Продавец консультант',70000.00),

('Охранник',60000.00),

('Грузчик',70000.00),

('Уборщик',50000.00),

('Кассир',70000.00),

('Менеджер',80000.00);

select \* from post;

-- Создание таблицы "Поставщик" с ограничениями на поле Company на ввод кириллицей и

-- Telephone на ограничение ввод с маской, а также Mail на ввод латиницы

create table Provider (

ID\_Provider SERIAL not null constraint PK\_Sklad primary key,

Company varchar(30) not null constraint CH\_Company

check (Company similar to '%[а-яА-Я]%'),

Telephone varchar(30) not null constraint CH\_Telephone

check (Telephone similar to '\+7\([0-9][0-9][0-9]\)[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'),

Mail varchar(30) not null default('-') constraint CH\_Mail

check (Mail similar to '%[a-zA-Z]%'),

Zakaz\_ID int not null references Zakaz (ID\_Zakaz)

);

-- Создание таблицу "Вид Продукта" с ограничениями на поле Type\_name на ввод кириллицей и с уникальностью данных

create table Product\_type (

ID\_Product\_type SERIAL not null constraint PK\_Product\_type primary key,

Type\_name varchar(30) not null constraint CH\_Type\_name

check (Type\_name similar to '%[а-яА-Я]%')

constraint UQ\_Type\_name unique

);

-- Добавление данных в таблицу "Вид Продукта"

insert into Product\_type(Type\_name)

values (Велосипед),

(BMX),

(Горный);

select \* from Product\_type;

-- Создание таблицы "Продукт" с ограничениями на поле Name\_of\_product на ввод кириллицей и с уникальностью данных

create table Product (

ID\_Product SERIAL not null constraint PK\_Product primary key,

Name\_of\_product varchar(30) not null constraint CH\_Name\_of\_product

check (Name\_of\_product similar to '%[а-яА-Я]%')

constraint UQ\_Name\_of\_product unique,

Product\_type\_ID int not null references Product\_type (ID\_Product\_type)

);

-- Добавление данных в таблицу "Продукт"

insert into Product(Name\_of\_product,Product\_type\_ID)

values (Велосипед,1),

(BMX,2),

(Горный,3),

select \* from Product\_type;

-- Создание таблицы "Заказ" с ограничениями на поле Order\_number, Number\_of\_product на ввод цифр и

-- с типом данных VARCHAR с макс. заданой длины

-- Order\_number имеет уникальность данных

-- поле Product\_Price имеет ограничение на >= 0.0

create table Zakaz (

ID\_Zakaz SERIAL not null constraint PK\_Zakaz primary key,

Order\_number varchar(12) not null constraint CH\_Order\_number

check (Order\_number similar to '[0-9]{12}')

constraint UQ\_Order\_number unique,

Number\_of\_product varchar(1) not null constraint CH\_Number\_of\_product

check (Number\_of\_product similar to '[1-9]{1}'),

Product\_Price decimal(38,2) not null constraint CH\_Product\_Price check

(Product\_Price >= 0.0),

Product\_ID int not null references Product (ID\_Product)

);

-- Добавление данных в таблицу "Заказ"

insert into Zakaz(Order\_number,Number\_of\_product,Product\_price,Product\_ID)

values (123456789123,1,400.00,1),

(123456789124,3,1000.00,2),

(123456789125,2,2000.00,3),

(123456789126,2,3000.00,4),

(123456789127,1,4000.00,5),

(123456789128,1,5000.00,6),

(123456789129,4,6000.00,7),

(123456789132,5,7000.00,8),

(123456789133,1,8000.00,9),

(123456789134,2,9000.00,10);

-- Создание таблицы "Чек" с ограничениями на поле Chek\_number, INN на ввод цифр с типом VARCHAR с макс. заданой длины

-- Chek\_number имеет уникальность данных

create table Chek (

ID\_Chek SERIAL not null constraint PK\_Chek primary key,

Chek\_number varchar(10) not null constraint CH\_Chek\_number

check (Chek\_number similar to '[0-9]{10}')

constraint UQ\_Chek\_number unique,

Date\_of\_printing\_check date not null,

INN varchar(10) not null constraint CH\_INN

check (INN similar to '[0-9]{10}'),

Employee\_ID int not null references Employee (ID\_Employee),

Zakaz\_ID int not null references Zakaz (ID\_Zakaz)

);

-- Добавление данных в таблицу "Чек"

insert into Chek(Chek\_number,Date\_of\_printing\_check,INN,Employee\_ID,Zakaz\_ID)

values (1234567891,'2021-12-01',9998887771,1,1),

(1234567892,'2021-12-02',1112223334,2,2),

(1234567893,'2021-12-03',4445556667,3,3),

(1234567894,'2021-12-04',5556667778,4,4),

(1234567895,'2021-12-05',8889994442,5,5),

(1234567896,'2021-12-06',8887776661,6,6),

(1234567897,'2021-12-07',9998881112,7,7),

(1234567898,'2021-12-08',4443335551,8,8),

(1234567899,'2021-12-09',7778881112,9,9),

(1234567900,'2021-12-10',9990001112,10,10);

select \* from Chek;

-- Создание таблицы "Расчёт заработной платы" с ограничениями на поле Premium, Vacations

-- на >= 0.0

-- Premium имеет ограничение default на 0.0

create table Payroll\_preparation (

ID\_Payroll\_preparation SERIAL not null constraint PK\_Payroll\_preparation primary key,

Date\_of\_payroll date not null,

Premium decimal(38,2) null constraint CH\_Premium check

(Premium >=0.0) default(0.0),

Vacations decimal(38,2) not null constraint CH\_Vacations check

(Vacations >=0.0),

Post\_ID int not null references Post (ID\_Post)

);

-- Добавление данных в таблицу "Расчёт заработной платы"

insert into Payroll\_preparation(Date\_of\_payroll,Premium,Vacations,Post\_ID)

values ('2021-12-25',22000.00,5000.00,1),

('2021-12-25',18000.00,6000.00,2),

('2021-12-26',16000.00,5000.00,3),

('2021-12-25',14000.00,4000.00,4),

('2021-12-25',14000.00,7000.00,5),

('2021-12-27',12000.00,3000.00,6),

('2021-12-25',14000.00,8000.00,7),

('2021-12-25',10000.00,6000.00,8),

('2021-12-25',14000.00,7000.00,9),

('2021-12-25',16000.00,3000.00,10);

select \* from Payroll\_preparation;

* + - 1. Результат

В данном подпункте, необходимо в виде иллюстраций, продемонстрировать разработанные объекты базы данных из инструментальной среды разработки баз данных.

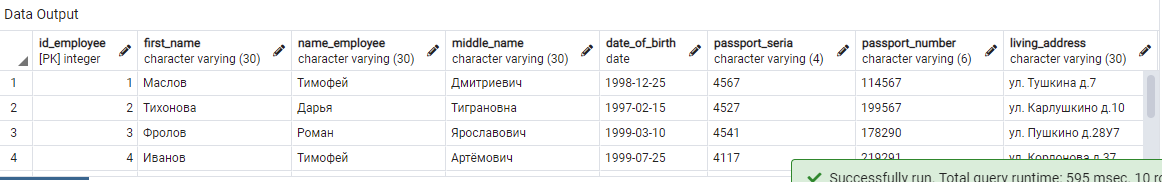


Рисунок 3 Таблица Employee

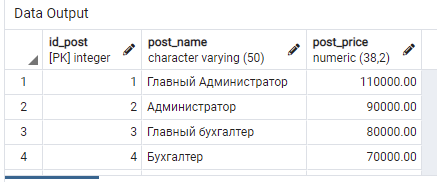
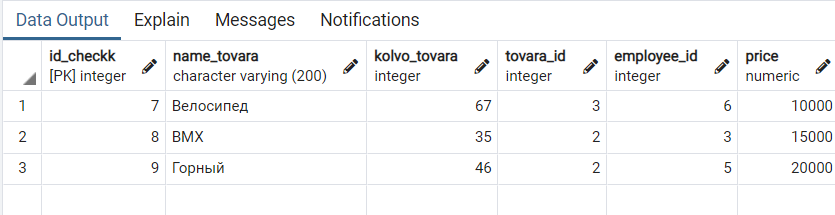


Рисунок 4 Таблица Должности



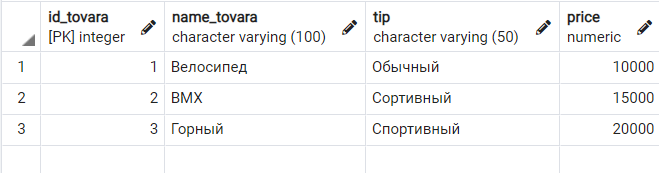


Рисунок 5 Товар

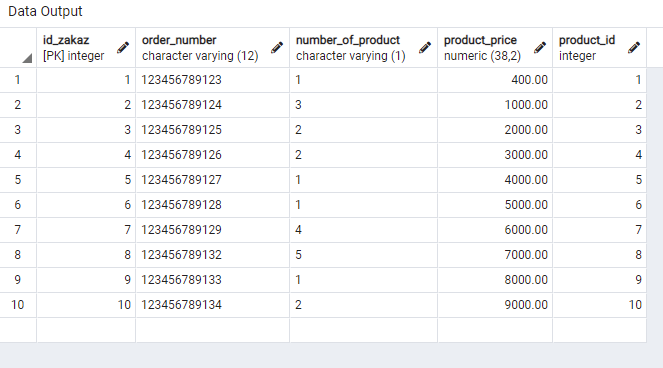


Рисунок 6 Заказ

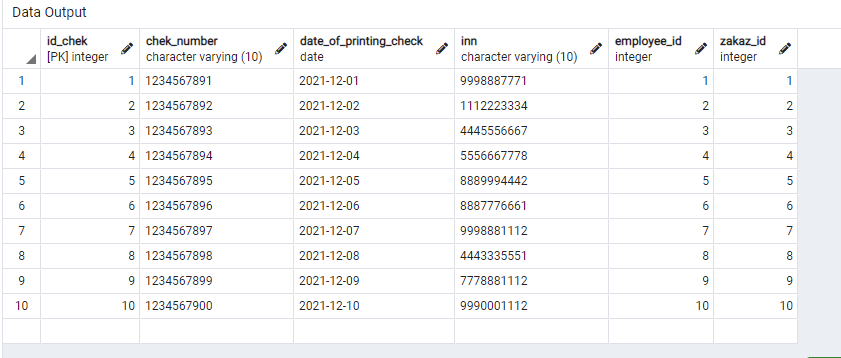


Рисунок 8 Чек

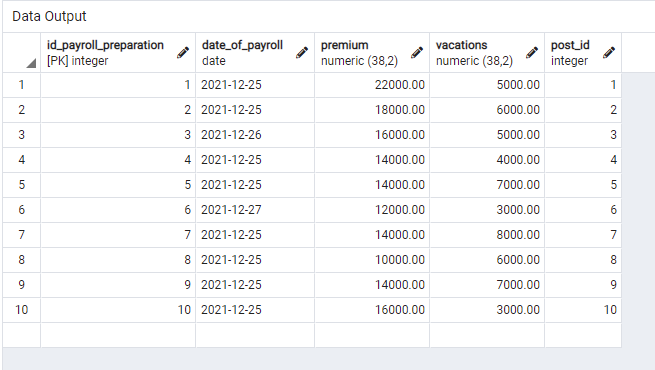


Рисунок 9 Заработная плата

* + 1. Реализация хранимых процедур
       1. Скрипт базы данных

-- Создание процедуры таблицы "Должность" на обновление данных

create or replace procedure Post\_Update(p\_ID\_Post int, p\_Post\_Name varchar(50),

p\_Post\_Price decimal(38,2))

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Post

where Post\_Name = p\_Post\_Name and Post\_Price = p\_Post\_Price;

begin

if have\_record>0 then

raise exception 'Должность с указынными: Названием и окладом, уже есть в таблице';

else

update Post set

Post\_Name = p\_Post\_Name,

Post\_Price = p\_Post\_Price

where ID\_Post = p\_ID\_Post;

end if;

end;

$$;

select \* from Post;

call Post\_Update(1,'Летчик',89000.00)

-- Создание процедуры таблицы "Сотрудник" на обновление данных

create or replace procedure Employee\_Update(p\_ID\_Employee int, p\_First\_Name varchar(30),

p\_Name\_Employee varchar(30), p\_Middle\_Name varchar(30),

p\_Date\_of\_birth date, p\_Passport\_seria varchar(4),

p\_Passport\_number varchar(6), p\_Living\_address varchar(30),

p\_Personal\_business\_number varchar(5), p\_Post\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Employee

where First\_Name = p\_First\_Name and Name\_Employee = p\_Name\_Employee and

Middle\_Name = p\_Middle\_Name and date\_of\_birth = p\_date\_of\_birth and

passport\_seria = p\_passport\_seria and passport\_number = p\_passport\_number and

living\_address = p\_living\_address and personal\_business\_number = p\_personal\_business\_number

and post\_id = p\_post\_id;

begin

if have\_record>0 then

raise exception 'Сотрудник с указынными: Фамилией, Именем, Отчеством и Личным номером уже есть в таблице';

else

update Employee set

First\_Name = p\_First\_Name,

Name\_Employee = p\_Name\_Employee,

Middle\_Name = p\_Middle\_Name,

date\_of\_birth = p\_date\_of\_birth,

passport\_seria = p\_passport\_seria,

passport\_number = p\_passport\_number,

living\_address = p\_living\_address,

personal\_business\_number = p\_personal\_business\_number,

post\_id = p\_post\_id

where ID\_Employee = p\_ID\_Employee;

end if;

end;

$$;

-- Создание процедуры таблицы "Поставщик" на обновление данных

create or replace procedure Provider\_Update(p\_ID\_Provider int, p\_Company varchar(30),

p\_Telephone varchar(30), p\_Mail varchar(30), p\_Zakaz\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Provider

where ID\_Provider = p\_ID\_Provider and Company = p\_Company and Telephone = p\_Telephone and Mail = p\_Mail and

Zakaz\_ID = p\_Zakaz\_ID;

begin

if have\_record>0 then

raise exception 'Такой поставщик уже существует';

else

update Provider set

ID\_Provider = p\_ID\_Provider,

Company = p\_Company,

Telephone = p\_Telephone,

Mail = p\_Mail,

Zakaz\_ID = p\_Zakaz\_ID

where ID\_Provider = p\_ID\_Provider;

end if;

end;

$$;

-- Создание процедуры таблицы "Вид Продукта" на обновление данных

create or replace procedure Product\_type\_Update(p\_ID\_Product\_type int, p\_Type\_name varchar(30))

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Product\_type

where Type\_name = p\_Type\_name;

begin

if have\_record>0 then

raise exception 'Вид товра с указынными: Наименованием, уже есть в таблице';

else

update Product\_type set

Type\_name = p\_Type\_name

where ID\_Product\_type = p\_ID\_Product\_type;

end if;

end;

$$;

-- Создание процедуры таблицы "Продукт" на обновление данных

create or replace procedure Product\_Update(p\_ID\_Product int, p\_Name\_of\_product varchar(30), p\_Product\_type\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Product

where Name\_of\_product = p\_Name\_of\_product and Product\_type\_ID = p\_Product\_type\_ID;

begin

if have\_record>0 then

raise exception 'Товар с указынными: Наименованием, уже есть в таблице';

else

update Product set

Name\_of\_product = p\_Name\_of\_product,

Product\_type\_ID = p\_Product\_type\_ID

where ID\_Product = p\_ID\_Product;

end if;

end;

$$;

-- Создание процедуры таблицы "Заказ" на обновление данных

create or replace procedure Zakaz\_Update(p\_ID\_Zakaz int, p\_Order\_number varchar(12),

p\_Number\_of\_product varchar(6), p\_Product\_Price decimal(38,2), p\_Product\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Zakaz

where Order\_number = p\_Order\_number and Number\_of\_product = p\_Number\_of\_product and

Product\_Price = p\_Product\_Price and Product\_ID = p\_Product\_ID;

begin

if have\_record>0 then

raise exception 'Заказ с указынными: Номер заказа и ценой продукта, уже есть в таблице';

else

update Zakaz set

Order\_number = p\_Order\_number,

Number\_of\_product = p\_Number\_of\_product,

Product\_Price = p\_Product\_Price,

Product\_ID = p\_Product\_ID

where ID\_Zakaz = p\_ID\_Zakaz;

end if;

end;

-- Создание процедуры таблицы "Чек" на обновление данных

create or replace procedure Chek\_Update(p\_ID\_Chek int, p\_Chek\_number varchar(10),

p\_Date\_of\_printing\_check date, p\_INN varchar(10), p\_Employee\_ID int,

p\_Zakaz\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Chek

where Chek\_number = p\_Chek\_number and Date\_of\_printing\_check = p\_Date\_of\_printing\_check and

INN = p\_INN and Employee\_ID = p\_Employee\_ID and Zakaz\_ID = p\_Zakaz\_ID;

begin

if have\_record>0 then

raise exception 'Чек с указынными: Номер чека, дата печати и ИНН, уже есть в таблице';

else

update Chek set

Chek\_number = p\_Chek\_number,

Date\_of\_printing\_check = p\_Date\_of\_printing\_check,

INN = p\_INN,

Employee\_ID = p\_Employee\_ID,

Zakaz\_ID = p\_Zakaz\_ID

where ID\_Chek = p\_ID\_Chek;

end if;

end;

$$;

-- Создание процедуры таблицы "Расчёт заработной платы" на обновление данных

create or replace procedure Payroll\_preparation\_Update(p\_ID\_Payroll\_preparation int, p\_Date\_of\_payroll date,

p\_Premium decimal(38,2), p\_Vacations decimal(38,2), p\_Post\_ID int)

language plpgsql

as $$

DECLARE have\_record int := count(\*) from Payroll\_preparation

where Date\_of\_payroll = p\_Date\_of\_payroll and Premium = p\_Premium and

Vacations = p\_Vacations and Post\_ID = p\_Post\_ID;

begin

if have\_record>0 then

raise exception 'Расчет заработной платы с указынными: Дата расчета, премия и отпускные, уже есть в таблице';

else

update Payroll\_preparation set

Date\_of\_payroll = p\_Date\_of\_payroll,

Premium = p\_Premium,

Vacations = p\_Vacations,

Post\_ID = p\_Post\_ID

where ID\_Payroll\_preparation = p\_ID\_Payroll\_preparation;

end if;

end;

$$;

* + - 1. Результат

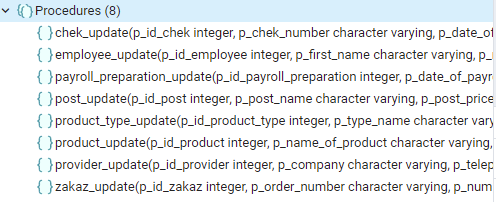


Рисунок 10 Результат

* + 1. Реализация представлений
       1. Скрипт базы данных

-- Создание представления "Чек"

create or replace view Chek\_cheking ("Фамилия и инициалы сотрудника", "Информация чека", "Занимаемая должность сотруднкиа",

"Информация заказа")

as

select First\_Name||' '|| substring (Name\_Employee from 1 for 1) ||'.' ||' '|| substring (Middle\_Name from 1 for 1) || '.',

'Номер чека: '||Chek\_Number ||', дата печати чека: '||Date\_of\_printing\_check ||', инн чека: '||INN,

'Название должности: '||Post\_Name,

'Номер заказа: '||Order\_number||', кол-во товаров: '||Number\_of\_product||', цена продукта: '||Product\_price

from Chek\_Combination

inner join Post on Post\_ID = ID\_Post

inner join Employee on Employee\_ID = ID\_Employee

inner join Zakaz on Zakaz\_ID = ID\_Zakaz

inner join Chek on Chek\_ID = ID\_Chek;

select \* from Chek\_cheking;

-- Создание таблицы для представления "Чек"

create table Chek\_Combination (

ID\_Chek\_Combination serial not null constraint PK\_Chek\_Combination primary key,

Chek\_ID int not null references Chek (ID\_Chek),

Employee\_ID int not null references Employee (ID\_Employee),

Post\_ID int not null references Post (ID\_Post),

Zakaz\_ID int not null references Zakaz (ID\_Zakaz),

Post\_Part decimal(38,1) null constraint CH\_Post\_Part check

(Post\_Part >0.0) default (0.1)

);

-- Добавление данных в созданную таблицу для представления "Чек"

insert into Chek\_Combination (Chek\_ID, Employee\_ID, Post\_ID, Zakaz\_ID, Post\_Part)

values (1,1,1,1,1),

(2,2,2,2,1),

(3,3,3,3,0.5),

(4,4,4,4,3),

(5,5,5,5,6),

(6,6,6,6,8),

(7,7,7,7,1),

(8,8,8,8,2),

(9,9,9,9,4),

(10,10,10,10,5);

select \* from Chek\_Combination;

* + - 1. Результат

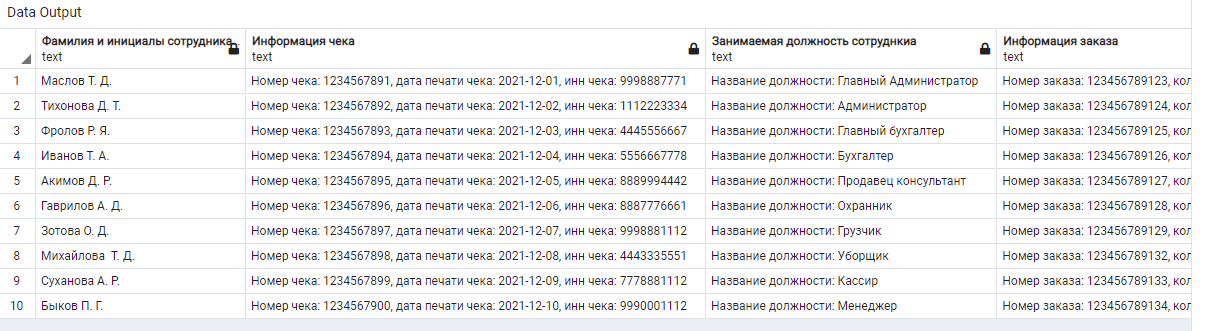


Рисунок 11 Представление

* + 1. Реализация функций
       1. Скрипт базы данных

-- Создание функции "Заработная плата сотрудников за отработанное время"

create or replace function Employee\_Amount(p\_id\_employee int, p\_worked\_hours decimal(38,2))

returns decimal(38,2)

language plpgsql

as $$

declare count\_money decimal(38,2);

begin

select (post\_part\*post\_price\*0.87)\*(p\_worked\_hours/32) into count\_money

from Chek\_Combination inner join post on post\_id = id\_post

where employee\_id = p\_id\_employee;

return count\_money;

end;

$$;

select Employee\_Amount(6,8);

-- Создание функции "Заработанная премия от оклада"

create or replace function Employee\_Premium(p\_id\_employee int)

returns decimal(38,2)

language plpgsql

as $$

declare count\_money decimal(38,2);

begin

select (post\_price \* 30)/100 into count\_money

from Chek\_Combination inner join post on post\_id = id\_post

where employee\_id = p\_id\_employee;

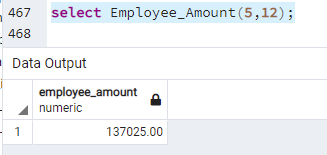
return count\_money;

end;

$$;

select Employee\_Premium(2);

* + - 1. Результат



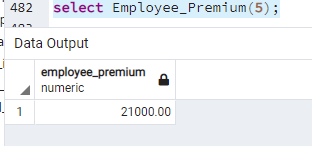


Рисунок 12 Функции

* + 1. Реализация триггеров
       1. Скрипт базы данных

-- Создание таблицы для триггера

create table Employee\_History

(

ID\_Employee\_History SERIAL not null constraint PK\_Employee\_History primary key,

Employee\_Info varchar not null,

Post\_Info varchar not null,

Chek\_Info varchar not null,

Zakaz\_Info varchar not null,

Date\_Create date not null

);

-- Создание триггерной функции на добавление

create or replace function fc\_History\_Insert ()

returns trigger

as $$

begin

insert into Employee\_History (Employee\_Info,Post\_Info,Chek\_Info,Zakaz\_Info,Date\_Create)

values ('Добавлена новая информация - '||(select first\_name|| ' '||name\_employee||' '||middle\_name from employee

where id\_employee = NEW.employee\_id),

(select post\_name||', оклад: '||post\_price from post

where id\_post= NEW.post\_id),

'Новый чек - '||(select chek\_number|| ' '||date\_of\_printing\_check||' '||inn from chek

where id\_chek = NEW.chek\_id),

'Новый заказ - '||(select order\_number|| ' '||number\_of\_product||' '||product\_price from zakaz

where id\_zakaz = NEW.zakaz\_id),

current\_date);

return new;

end;

$$

language plpgsql;

-- Создание триггера на добавление

create trigger tg\_History\_Insert

after Insert on public.Chek\_Combination

for each row

execute procedure fc\_History\_Insert();

insert into public.Chek\_Combination (Chek\_ID, Employee\_ID, Post\_ID, Zakaz\_ID, Post\_Part)

values (13,2,3,4,1.5);

* + - 1. Результат

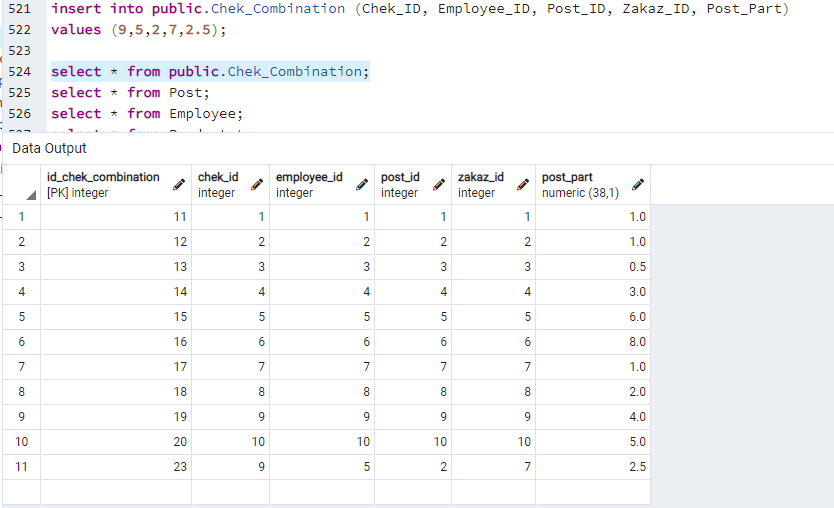
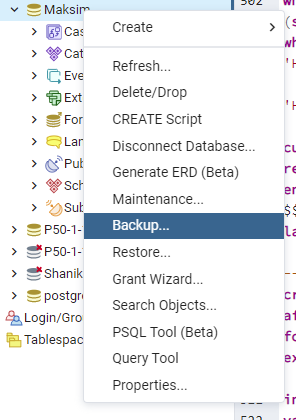


Рисунок 13 Триггеры

* + - 1. Создание Back up



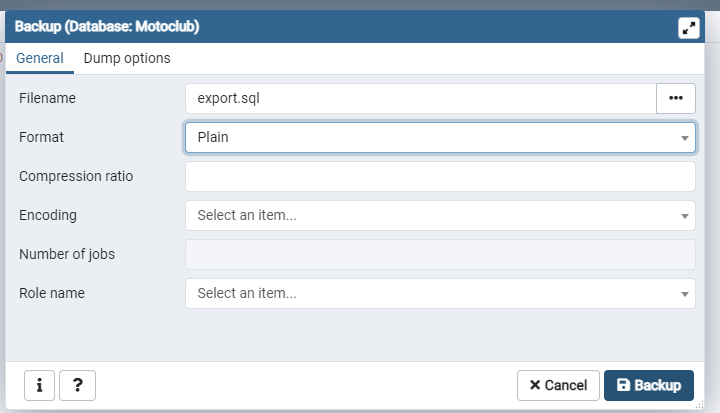
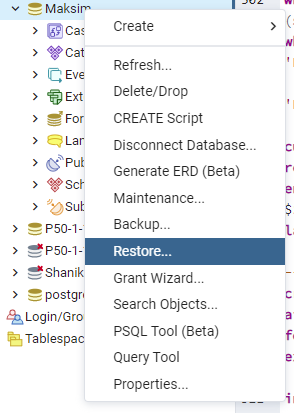


Рисунок 14 Back up

* + - 1. Восстановление данных



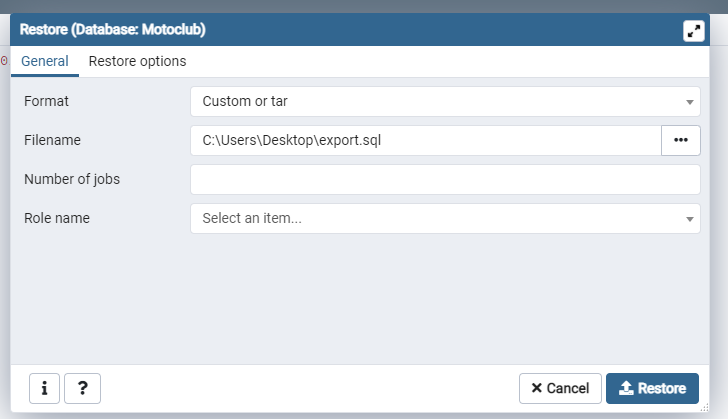


Рисунок 15 Восстановление данных

* 1. MySQL
     1. Реализация таблиц
        1. Скрипт базы данных

-- Создание таблицы "Сотрудник" с ограничениями на поле First\_Name, Name\_Employee, Middle\_Name

-- на ввод кириллицей

-- поле Passport\_seria, Passport\_number, Personal\_business\_number имеют ограничения на ввод цифр с условием VARCHAR до

-- макс. указанного значения

create table Employee

(

ID\_Employee int not null auto\_increment primary key,

First\_Name varchar(30) not null check(First\_Name regexp '[а-яА-Я]'),

Name\_Employee varchar(30) not null check(Name\_Employee regexp '[а-яА-Я]'),

Middle\_Name varchar(30) null default '-' check(Middle\_Name regexp '[а-яА-Я]'),

Date\_of\_birth date not null,

Passport\_seria varchar(4) not null check(Passport\_seria regexp '[0-9]{4}'),

Passport\_number varchar(6) not null check(Passport\_number regexp '[0-9]{4}'),

Living\_address varchar(30) not null,

Personal\_business\_number varchar(5) not null unique check(Personal\_business\_number regexp '[0-9]{5}'),

Post\_ID int not null,

foreign key (Post\_ID) references Post (ID\_Post)

);

-- Добавление данных в таблицу "Сотрудник"

insert into employee(First\_Name,Name\_Employee,Middle\_Name,date\_of\_birth,passport\_seria,passport\_number,living\_address,

personal\_business\_number,post\_id)

values ('Маслов','Тимофей','Дмитриевич','1998-12-25','4567','114567','ул. Тушкина д.7','47389',1),

('Тихонова','Дарья','Тиграновна','1997-02-15','4527','199567','ул. Карлушкино д.10','90372',2),

('Фролов','Роман','Ярославович','1999-03-10','4541','178290','ул. Пушкино д.28У7','11923',3),

('Иванов','Тимофей','Артёмович','1999-07-25','4117','219291','ул. Корлонова д.37','73483',4),

('Акимов','Денис','Русланович','1987-08-14','4589','127867','ул. Фушкина д.12','73484',5),

('Гаврилов','Арсений','Денисович','1985-10-19','4997','192837','ул. Душкина д.73','73485',6),

('Зотова','Оливия','Дамировна','1983-11-15','4535','614567','ул. Корушкина д.17','73486',7),

('Михайлова ','Таисия','Давидовна','1998-08-25','4997','119877','ул. Лушкина д.97','73487',8),

('Суханова','Алиса','Романовна','1998-12-25','4567','114567','ул. Тушкина д.7','73489',9),

('Быков','Павел','Георгиевич','1998-12-25','4567','114567','ул. Тушкина д.7','73490',10);

select \* from employee;

-- Создание таблицы "Должность" с ограничениями на поле Post\_Name на ввод кириллицей и на ограничение

-- unique - проверка на ввод уникальных данных

-- поле Post\_Price имеет ограничение, что оно должно быть >= 0.0 и если null то возвращает значение 0.0

create table Post (

ID\_Post int not null auto\_increment primary key,

Post\_Name varchar(50) not null unique check(Post\_name regexp '[а-яА-Я]'),

Post\_Price decimal(38,2) null default 0.0 check(Post\_Price >=0.0)

);

-- Занесение данных в таблицу "Должность"

insert into Post (Post\_Name, Post\_Price)

values ('Главный Администратор',110000.00),

('Администратор',90000.00),

('Главный бухгалтер',80000.00),

('Бухгалтер',70000.00),

('Продавец консультант',70000.00),

('Охранник',60000.00),

('Грузчик',70000.00),

('Уборщик',50000.00),

('Кассир',70000.00),

('Менеджер',80000.00);

select \* from post;

-- Создание таблицы "Поставщик" с ограничениями на поле Company на ввод кириллицей и

-- Telephone на ограничение ввод с маской, а также Mail на ввод латиницы

create table Provider (

ID\_Provider int not null auto\_increment primary key,

Company varchar(30) not null check (Company regexp '[а-яА-Я]'),

Telephone varchar(30) not null,

Mail varchar(30) not null default('-') check (Mail regexp '[a-zA-Z]'),

Zakaz\_ID int not null,

foreign key (Zakaz\_ID) references Zakaz (ID\_Zakaz)

);

-- Создание таблицу "Вид Продукта" с ограничениями на поле Type\_name на ввод кириллицей и с уникальностью данных

create table Product\_type (

ID\_Product\_type int not null auto\_increment primary key,

Type\_name varchar(30) not null unique check(Type\_name regexp '[а-яА-Я]')

);

-- Добавление данных в таблицу "Вид Продукта"

insert into Product\_type(Type\_name)

values (Велосипед),

(BMX),

(Горный),

-- Создание таблицы "Продукт" с ограничениями на поле Name\_of\_product на ввод кириллицей и с уникальностью данных

create table Product (

ID\_Product int not null auto\_increment primary key,

Name\_of\_product varchar(30) not null unique check(Name\_of\_product regexp '[а-яА-Я]'),

Product\_type\_ID int not null,

foreign key (Product\_type\_ID) references Product\_type (ID\_Product\_type)

);

-- Добавление данных в таблицу "Продукт"

insert into Product(Name\_of\_product,Product\_type\_ID)

values ('Велосипед',1),

('BMX',2),

('Горный',3),

-- Создание таблицы "Заказ" с ограничениями на поле Order\_number, Number\_of\_product на ввод цифр и

-- с типом данных VARCHAR с макс. заданой длины

-- Order\_number имеет уникальность данных

-- поле Product\_Price имеет ограничение на >= 0.0

create table Zakaz (

ID\_Zakaz int not null auto\_increment primary key,

Order\_number varchar(12) not null unique check(Order\_number regexp '[0-9]{12}'),

Number\_of\_product varchar(1) not null check(Number\_of\_product regexp '[1-9]{1}'),

Product\_Price decimal(38,2) not null check(Product\_Price >= 0.0),

Product\_ID int not null,

foreign key (Product\_ID) references Product (ID\_Product)

);

-- Добавление данных в таблицу "Заказ"

insert into Zakaz(Order\_number,Number\_of\_product,Product\_price,Product\_ID)

values (123456789123,1,400.00,1),

(123456789124,3,1000.00,2),

(123456789125,2,2000.00,3),

(123456789126,2,3000.00,4),

(123456789127,1,4000.00,5),

(123456789128,1,5000.00,6),

(123456789129,4,6000.00,7),

(123456789132,5,7000.00,8),

(123456789133,1,8000.00,9),

(123456789134,2,9000.00,10);

-- Создание таблицы "Чек" с ограничениями на поле Chek\_number, INN на ввод цифр с типом VARCHAR с макс. заданой длины

-- Chek\_number имеет уникальность данных

create table Chek (

ID\_Chek int not null auto\_increment primary key,

Chek\_number varchar(10) not null unique check(Chek\_number regexp '[0-9]{10}'),

Date\_of\_printing\_check date not null,

INN varchar(10) not null check(INN regexp '[0-9]{10}'),

Employee\_ID int not null,

Zakaz\_ID int not null,

foreign key (Employee\_ID) references Employee (ID\_Employee),

foreign key (Zakaz\_ID) references Zakaz (ID\_Zakaz)

);

-- Добавление данных в таблицу "Чек"

insert into Chek(Chek\_number,Date\_of\_printing\_check,INN,Employee\_ID,Zakaz\_ID)

values (1234567891,'2021-12-01',9998887771,1,1),

(1234567892,'2021-12-02',1112223334,2,2),

(1234567893,'2021-12-03',4445556667,3,3),

(1234567894,'2021-12-04',5556667778,4,4),

(1234567895,'2021-12-05',8889994442,5,5),

(1234567896,'2021-12-06',8887776661,6,6),

(1234567897,'2021-12-07',9998881112,7,7),

(1234567898,'2021-12-08',4443335551,8,8),

(1234567899,'2021-12-09',7778881112,9,9),

(1234567900,'2021-12-10',9990001112,10,10);

-- Создание таблицы "Расчёт заработной платы" с ограничениями на поле Premium, Vacations

-- на >= 0.0

-- Premium имеет ограничение default на 0.0

create table Payroll\_preparation (

ID\_Payroll\_preparation int not null auto\_increment primary key,

Date\_of\_payroll date not null,

Premium decimal(38,2) null default 0.0 check(Premium >=0.0),

Vacations decimal(38,2) not null check(Vacations >=0.0),

Post\_ID int not null,

foreign key (Post\_ID) references Post (ID\_Post)

);

-- Добавление данных в таблицу "Расчёт заработной платы"

insert into Payroll\_preparation(Date\_of\_payroll,Premium,Vacations,Post\_ID)

values ('2021-12-25',22000.00,5000.00,1),

('2021-12-25',18000.00,6000.00,2),

('2021-12-26',16000.00,5000.00,3),

('2021-12-25',14000.00,4000.00,4),

('2021-12-25',14000.00,7000.00,5),

('2021-12-27',12000.00,3000.00,6),

('2021-12-25',14000.00,8000.00,7),

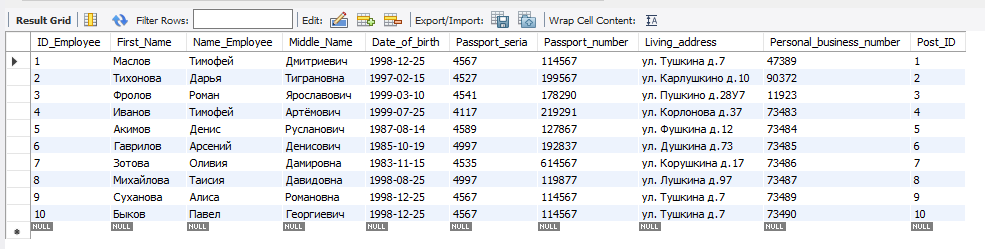
('2021-12-25',10000.00,6000.00,8),

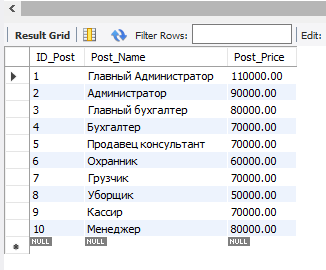
('2021-12-25',14000.00,7000.00,9),

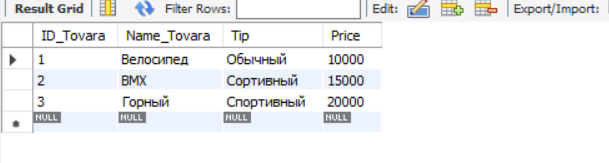
('2021-12-25',16000.00,3000.00,10);

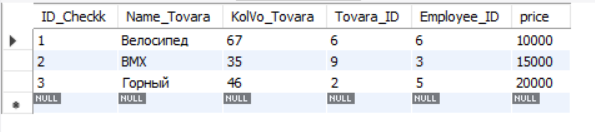
* + - 1. Результат

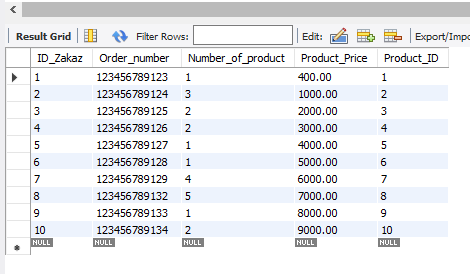
В данном подпункте, необходимо в виде иллюстраций, продемонстрировать разработанные объекты базы данных из инструментальной среды разработки баз данных.

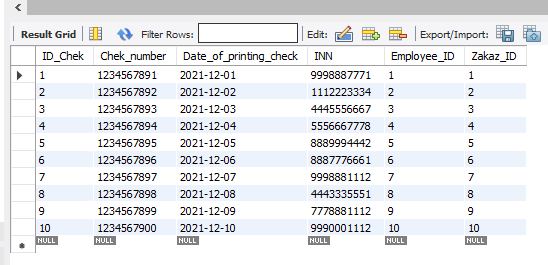


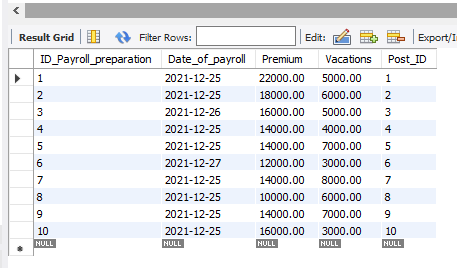












* + 1. Реализация хранимых процедур
       1. Скрипт базы данных

-- Создание процедуры таблицы "Сотрудник" на обновление данных

DELIMITER //

create or replace procedure Employee\_Update(p\_ID\_Employee int, p\_First\_Name varchar(30),

p\_Name\_Employee varchar(30), p\_Middle\_Name varchar(30),

p\_Date\_of\_birth date, p\_Passport\_seria varchar(4),

p\_Passport\_number varchar(6), p\_Living\_address varchar(30),

p\_Personal\_business\_number varchar(5), p\_Post\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Employee

where First\_Name = p\_First\_Name and Name\_Employee = p\_Name\_Employee and

Middle\_Name = p\_Middle\_Name and date\_of\_birth = p\_date\_of\_birth and

passport\_seria = p\_passport\_seria and passport\_number = p\_passport\_number and

living\_address = p\_living\_address and personal\_business\_number = p\_personal\_business\_number

and post\_id = p\_post\_id;

if have\_record>0 then select 'Такой сотрудник уже есть';

else update Employee set

First\_Name = p\_First\_Name,

Name\_Employee = p\_Name\_Employee,

Middle\_Name = p\_Middle\_Name,

date\_of\_birth = p\_date\_of\_birth,

passport\_seria = p\_passport\_seria,

passport\_number = p\_passport\_number,

living\_address = p\_living\_address,

personal\_business\_number = p\_personal\_business\_number,

post\_id = p\_post\_id

where ID\_Employee = p\_ID\_Employee;

end if;

end;

-- Создание процедуры таблицы "Должность" на обновление данных

DELIMITER //

create or replace procedure Post\_Update(p\_ID\_Post int, p\_Post\_Name varchar(50),

p\_Post\_Price decimal(38,2))

begin

DECLARE have\_record int;

select count(\*) into have\_record from Post

where Post\_Name = p\_Post\_Name and Post\_Price = p\_Post\_Price;

if have\_record>0 then select 'Такая должность уже есть';

else update Post set

Post\_Name = p\_Post\_Name,

Post\_Price = p\_Post\_Price

where ID\_Post = p\_ID\_Post;

end if;

end;

-- Создание процедуры таблицы "Поставщик" на обновление данных

DELIMITER //

create or replace procedure Provider\_Update(p\_ID\_Provider int, p\_Company varchar(30),

p\_Telephone varchar(30), p\_Mail varchar(30), p\_Zakaz\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Provider

where ID\_Provider = p\_ID\_Provider and Company = p\_Company and Telephone = p\_Telephone and Mail = p\_Mail and

Zakaz\_ID = p\_Zakaz\_ID;

if have\_record>0 then select 'Такой поставщик уже существует';

else update Provider set

ID\_Provider = p\_ID\_Provider,

Company = p\_Company,

Telephone = p\_Telephone,

Mail = p\_Mail,

Zakaz\_ID = p\_Zakaz\_ID

where ID\_Provider = p\_ID\_Provider;

end if;

end;

-- Создание процедуры таблицы "Вид Продукта" на обновление данных

DELIMITER //

create or replace procedure Product\_type\_Update(p\_ID\_Product\_type int, p\_Type\_name varchar(30))

begin

DECLARE have\_record int;

select count(\*) into have\_record from Product\_type

where Type\_name = p\_Type\_name;

if have\_record>0 then select 'Такой тип продукта уже есть';

else update Product\_type set

Type\_name = p\_Type\_name

where ID\_Product\_type = p\_ID\_Product\_type;

end if;

end;

-- Создание процедуры таблицы "Продукт" на обновление данных

DELIMITER //

create or replace procedure Product\_Update(p\_ID\_Product int, p\_Name\_of\_product varchar(30), p\_Product\_type\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Product

where Name\_of\_product = p\_Name\_of\_product and Product\_type\_ID = p\_Product\_type\_ID;

if have\_record>0 then select 'Такой продукт уже есть';

else update Product set

Name\_of\_product = p\_Name\_of\_product,

Product\_type\_ID = p\_Product\_type\_ID

where ID\_Product = p\_ID\_Product;

end if;

end;

-- Создание процедуры таблицы "Заказ" на обновление данных

DELIMITER //

create or replace procedure Zakaz\_Update(p\_ID\_Zakaz int, p\_Order\_number varchar(12),

p\_Number\_of\_product varchar(6), p\_Product\_Price decimal(38,2), p\_Product\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Zakaz

where Order\_number = p\_Order\_number and Number\_of\_product = p\_Number\_of\_product and

Product\_Price = p\_Product\_Price and Product\_ID = p\_Product\_ID;

if have\_record>0 then select 'Такой заказ уже есть';

else update Zakaz set

Order\_number = p\_Order\_number,

Number\_of\_product = p\_Number\_of\_product,

Product\_Price = p\_Product\_Price,

Product\_ID = p\_Product\_ID

where ID\_Zakaz = p\_ID\_Zakaz;

end if;

end;

-- Создание процедуры таблицы "Чек" на обновление данных

DELIMITER //

create or replace procedure Chek\_Update(p\_ID\_Chek int, p\_Chek\_number varchar(10),

p\_Date\_of\_printing\_check date, p\_INN varchar(10), p\_Employee\_ID int,

p\_Zakaz\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Chek

where Chek\_number = p\_Chek\_number and Date\_of\_printing\_check = p\_Date\_of\_printing\_check and

INN = p\_INN and Employee\_ID = p\_Employee\_ID and Zakaz\_ID = p\_Zakaz\_ID;

if have\_record>0 then select 'Такой чек уже есть';

else update Chek set

Chek\_number = p\_Chek\_number,

Date\_of\_printing\_check = p\_Date\_of\_printing\_check,

INN = p\_INN,

Employee\_ID = p\_Employee\_ID,

Zakaz\_ID = p\_Zakaz\_ID

where ID\_Chek = p\_ID\_Chek;

end if;

end;

-- Создание процедуры таблицы "Расчёт заработной платы" на обновление данных

DELIMITER //

create or replace procedure Payroll\_preparation\_Update(p\_ID\_Payroll\_preparation int, p\_Date\_of\_payroll date,

p\_Premium decimal(38,2), p\_Vacations decimal(38,2), p\_Post\_ID int)

begin

DECLARE have\_record int;

select count(\*) into have\_record from Payroll\_preparation

where Date\_of\_payroll = p\_Date\_of\_payroll and Premium = p\_Premium and

Vacations = p\_Vacations and Post\_ID = p\_Post\_ID;

if have\_record>0 then select 'Такой счёт уже есть';

else update Payroll\_preparation set

Date\_of\_payroll = p\_Date\_of\_payroll,

Premium = p\_Premium,

Vacations = p\_Vacations,

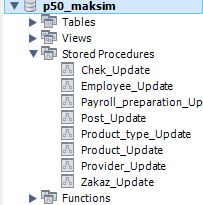
Post\_ID = p\_Post\_ID

where ID\_Payroll\_preparation = p\_ID\_Payroll\_preparation;

end if;

end;

* + - 1. Результат



* + 1. Реализация представлений
       1. Скрипт базы данных

-- Создание таблицы для представления "Чек"

create table Chek\_Combination (

ID\_Chek\_Combination int not null auto\_increment primary key,

Chek\_ID int not null,

foreign key (Chek\_ID) references Chek (ID\_Chek),

Employee\_ID int not null,

foreign key (Employee\_ID) references Employee (ID\_Employee),

Post\_ID int not null,

foreign key (Post\_ID) references Post (ID\_Post),

Zakaz\_ID int not null,

foreign key (Zakaz\_ID) references Zakaz (ID\_Zakaz),

Post\_Part decimal(38,1) null default 0.1 check(Post\_Part >0.0)

);

-- Добавление данных в созданную таблицу для представления "Чек"

insert into Chek\_Combination (Chek\_ID, Employee\_ID, Post\_ID, Zakaz\_ID, Post\_Part)

values (1,1,1,1,1),

(2,2,2,2,1),

(3,3,3,3,0.5),

(4,4,4,4,3),

(5,5,5,5,6),

(6,6,6,6,8),

(7,7,7,7,1),

(8,8,8,8,2),

(9,9,9,9,4),

(10,10,10,10,5);

select \* from Chek\_Combination;

-- Создание представления "Чек"

create or replace view Chek\_cheking

as

select

concat(First\_Name, ' ', substring(Name\_Employee,1,1), ' ', substring(Middle\_Name,1,1),'.') as "Фамилия и инициалы сотрудника",

concat('Номер чека: ', Chek\_Number, ', дата печати чека: ', Date\_of\_printing\_check, ', инн чека: ', INN) as "Информация чека",

concat('Название должности: ', Post\_Name) as "Занимаемая должность сотруднкиа",

concat('Номер заказа: ', Order\_number, ', кол-во товаров: ', Number\_of\_product, ', цена продукта: ', Product\_price) as "Информация заказа"

from Chek\_Combination

inner join Post on Post\_ID = ID\_Post

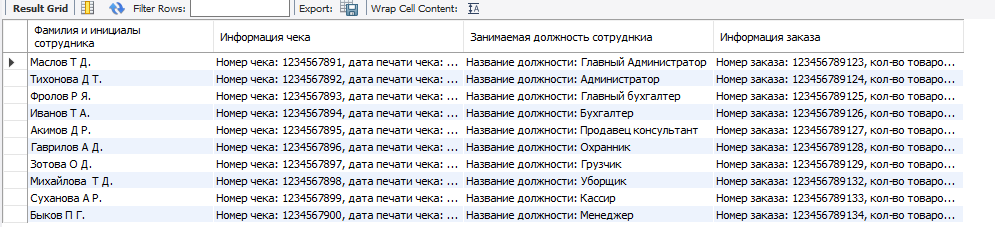
inner join Employee on Employee\_ID = ID\_Employee

inner join Zakaz on Zakaz\_ID = ID\_Zakaz

inner join Chek on Chek\_ID = ID\_Chek;

select \* from Chek\_cheking;

* + - 1. Результат



* + 1. Реализация функций
       1. Скрипт базы данных

-- Создание функции "Заработанная премия от оклада"

DELIMITER //

create or replace function Employee\_Premium(p\_id\_employee int)

returns decimal(38,2)

begin

declare count\_money decimal(38,2);

select (post\_price \* 30)/100 into count\_money

from Chek\_Combination

inner join post on post\_id = id\_post

where employee\_id = p\_id\_employee;

return count\_money;

end;

-- Создание функции "Заработанная премия от оклада"

DELIMITER //

create function Employee\_Amount(p\_Employee\_ID int, p\_Worked\_Hours decimal(38,1))

returns decimal(38,2)

begin

declare count\_money decimal(38,2);

select (Post\_Part\*Post\_Price\*0.87)\*(p\_Worked\_Hours/32) into count\_money

from Chek\_Combination

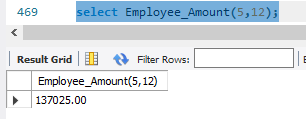
inner join post on Post\_ID = ID\_Post

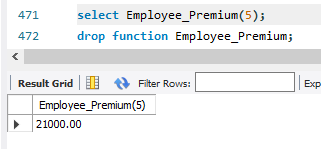
where Employee\_ID = p\_Employee\_ID;

return count\_money;

end;

* + - 1. Результат





* + 1. Реализация триггеров
       1. Скрипт базы данных

-- Создание таблицы для триггера

create table Employee\_History

(

ID\_Employee\_History int not null auto\_increment primary key,

Employee\_Info varchar(50) not null,

Post\_Info varchar(50) not null,

Chek\_Info varchar(50) not null,

Zakaz\_Info varchar(50) not null,

Date\_Create date not null

);

-- Создание триггера на обновление

DELIMITER //

create or replace trigger fc\_History\_Update

after update on Chek\_Combination

for each row

begin

if NEW.ID\_Chek\_Combination then

insert into Employee\_History(Employee\_Info,Post\_Info,Chek\_Info,Zakaz\_Info,Date\_create)

values ('Запись изменена',

(select concat(First\_Name,' ',Name\_Employee,' ',Middle\_Name) from employee where ID\_Employee = new.Employee\_ID),

(select concat(Post\_name,', оклад: ',cast(Post\_Price as varchar(100))) from post where ID\_Post=new.Post\_ID),

(select concat('Изменен чек - ', chek\_number, ' ', date\_of\_printing\_check, ' ', inn) from chek where ID\_Chek=new.Chek\_ID),

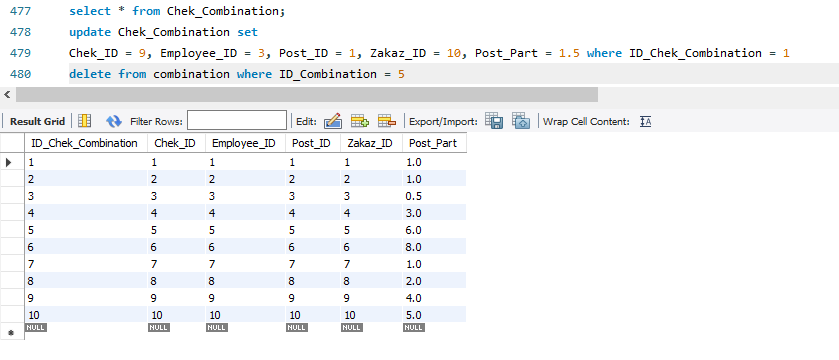
(select concat('Изменен заказ - ', order\_number, ' ', number\_of\_product, ' ', product\_price) from zakaz where ID\_Zakaz=new.Zakaz\_ID),

current\_date);

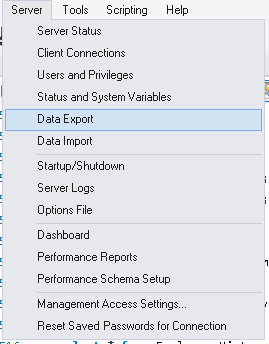
end if;

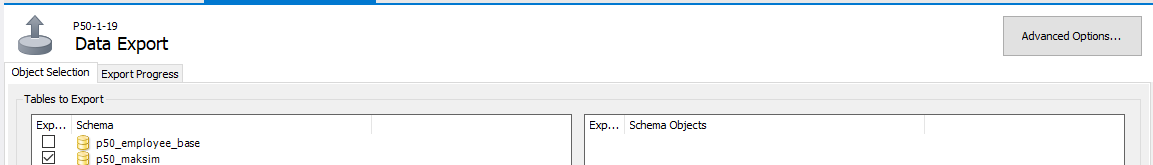
end;

* + - 1. Результат



* + 1. Резервное копирование и восстановление БД
       1. Создание резервной копии

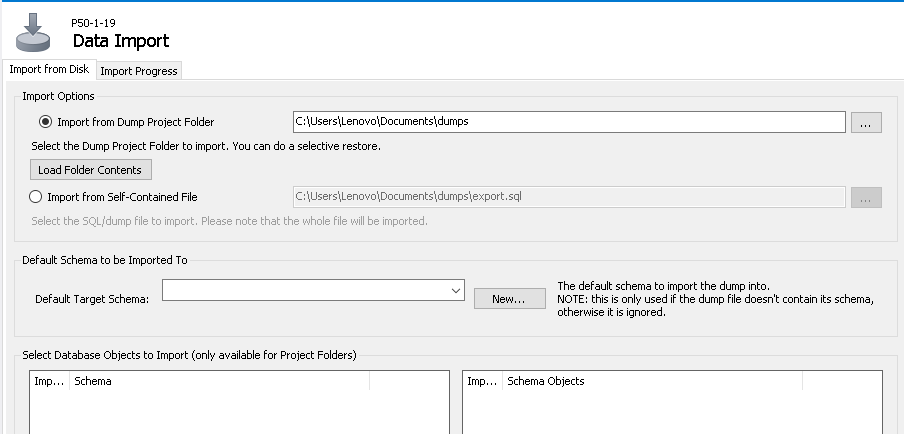




* + - 1. Восстановление БД

Изображение выглядит как текст

Автоматически созданное описание



1. СРАВНЕНИЕ СУБД

В данном разделе в виде таблицы необходимо представить, сравнение двух систем управления базами данных, с указанием всех достоинств, удобств и недостатков.

Таблица 4.1 – Сравнение СУБД

|  |  |  |
| --- | --- | --- |
|  | Достоинства | Недостатки |
| PostgreSQL | поддержка БД неограниченного размера  мощные и надёжные механизмы транзакций и репликации | нет сертифицированного дистрибутива ни для Linux ни для Windows.  Отсутствие примеров кода и литературы на русском |
| MySQL | Простота в использовании.  Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта. | Недостаточная надежность.  Низкая скорость разработки. |
| PGAdmin | Простота интерфейса.  Корректный вывод ошибок | Неоптимизированный запуск среды.  Отсутствие вывода таблиц |
| MySQL Workbench | Подсказки  Сохранение открытых вкладок | Неудобный вывод ошибок  Неудобный запуск кода |

1. ВЫВОД

В работе были разобраны функции и триггеры. Так же были созданы связи внутренних и внешних ключей. Была создана база данных «Велосипедный магазин». Работа делалась в PgAdmin и MySQL Workbench.

1. СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1) - <https://metanit.com/> - руководство по MySQL Workbench и PG Admin