**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 1**

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему: «Разработка серверной части информационной системы

в СУБД MS SQL Server»

Вариант 27

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**Цель работы:**

Разработать серверную часть клиент-серверной информационной системы, основанной на базе данных в заданной предметной области средствами СУБД MS SQL Server.

**Задание:**

1. Разработать логическую модель реляционной базы данных, моделирующую предметную область согласно своему варианту задания. Структура БД должна быть нормализована – таблицы должны удовлетворять требованиям третьей нормальной формы.

2. Создать базу данных и таблицы в СУБД MS SQL Server и заполнить их тестовым набором данных, для этого написать Transact SQL скрипт:

2.1. Создания базы данных и ее таблиц.

При создании таблиц должны быть назначены первичные и внешние ключи и установлены необходимые ограничения целостности данных. Наименования таблиц и полей давать в соответствии с соглашением об именовании этих объектов.

2.2. Заполнения не менее чем трех таблиц БД данными (см. пример внутри этого электронного курса).

При выборе таблиц для заполнения тестовыми наборами руководствоваться следующим:

выбранные таблицы должны содержать основную информацию, касающиеся предметной области приложения;

таблицы должны быть связаны непосредственно;

следует воздерживаться от выбора таблиц, характеризующих кадровую подсистему приложения (данные о сотрудниках и их должностях);

не менее, чем одна таблица должна находиться в схеме базы данных на стороне отношения «многие».

При этом заполнение осуществлять в следующем порядке: сначала генерируются данные для таблиц на стороне отношения «один» (таблицы- «справочники»), потом – в таблицы на стороне отношения «многие» («оперативные таблицы»).

БД заполнить записями в количестве, необходимом для отладки и демонстрации возможностей приложения. Таблицы на стороне отношения один должны содержать не менее 500 записей, таблице на стороне отношения многие должны содержать не менее 20000 записей.

3. Создать с использованием средств Transact SQL представления, позволяющие отображать данные в удобном для пользователя виде, и заменяющие часто используемые запросы на выборку из двух и более связанных таблиц.

4. Написать не менее трех хранимых процедур с параметрами для вставки и (или) обновление данных в таблицы базы данных.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

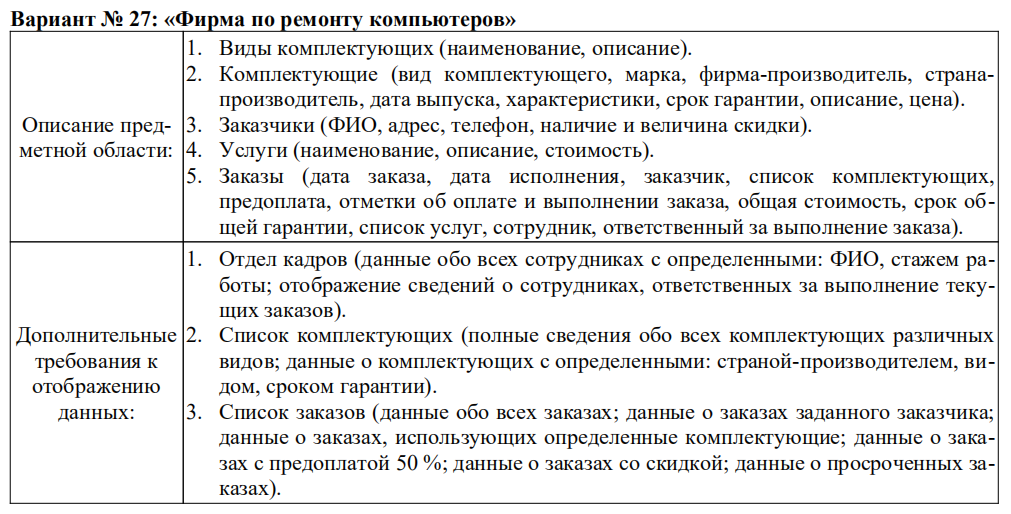
****

Рисунок 1 – Предметная область

**Ход работы:**

Задание 1

Первым делом была сформирована диаграмма базы данных, соответствующая заданному варианту и удовлетворяющая условиям задания.

Сформированная диаграмма изображена на рисунке 2.

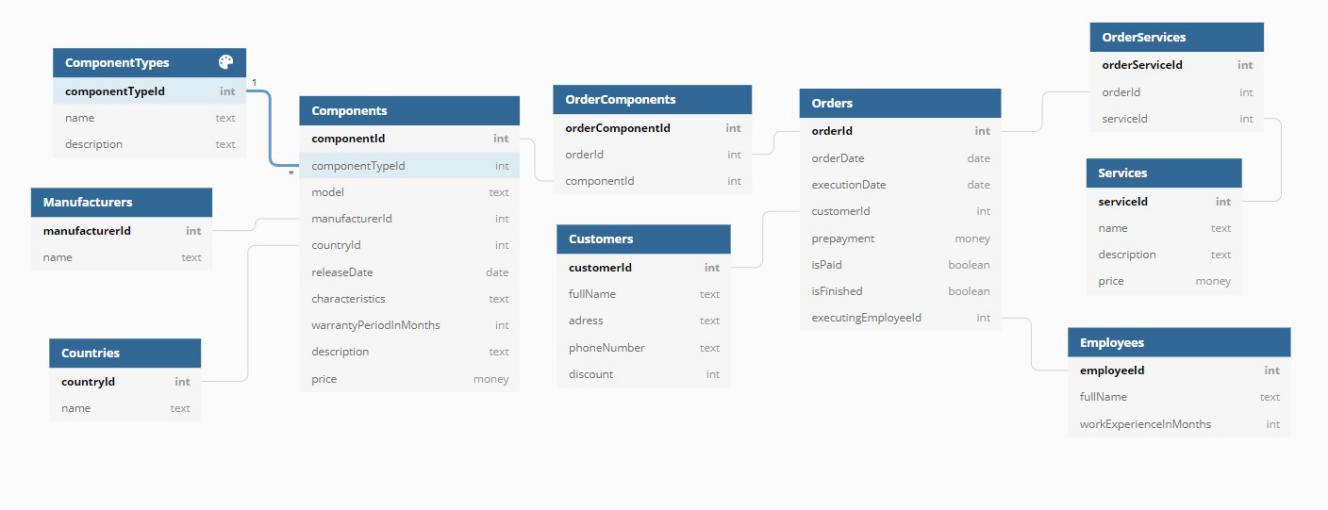


Рисунок 2 – Сформированная диаграмма базы данных

Задание 2

Далее была создана база данных и были написаны скрипты для создания таблиц базы данных.

Скриншот скрипта создания таблиц базы данных изображён на рисунке 3.

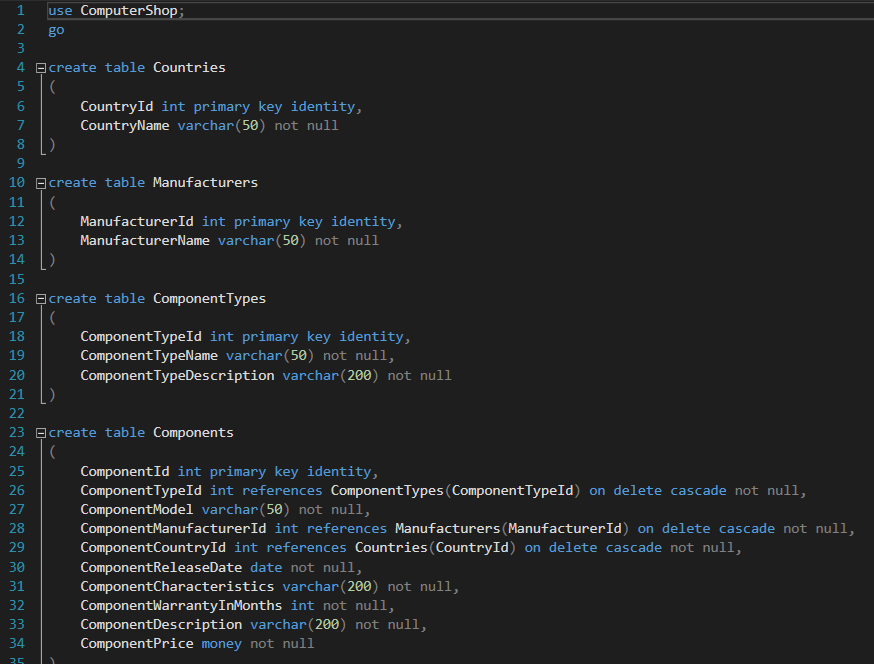


Рисунок 3 – Скрипт создания таблиц базы данных

Скриншот скрипта, выполняющего заполнение таблиц тестовыми данными изображён на рисунке 4.

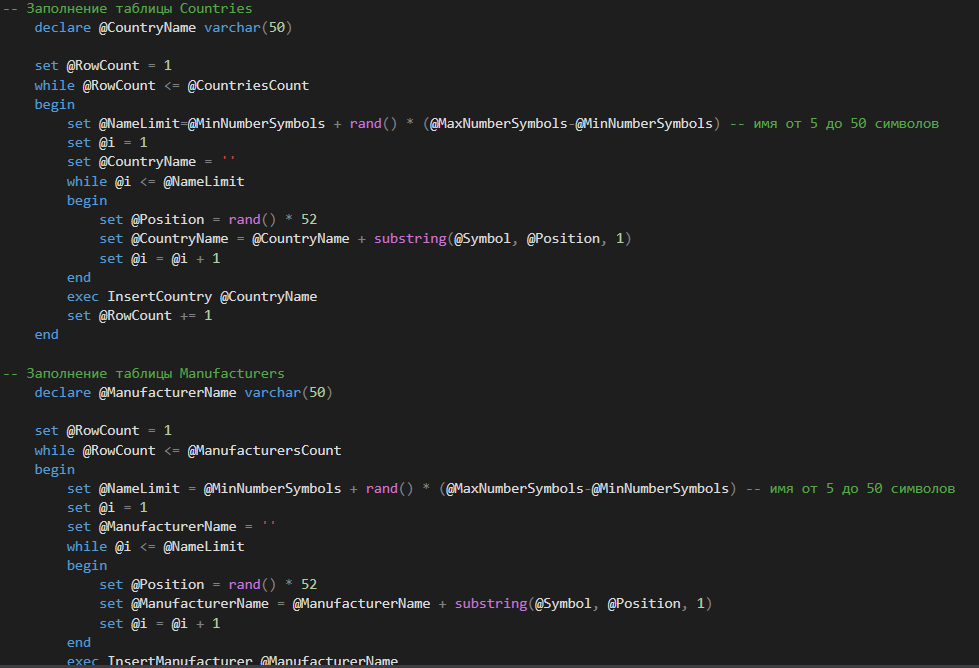


Рисунок 4 – Скрипт заполнения таблиц тестовыми данными

Задание 3

Далее был создан скрипт, создающий представление (рисунок 5).

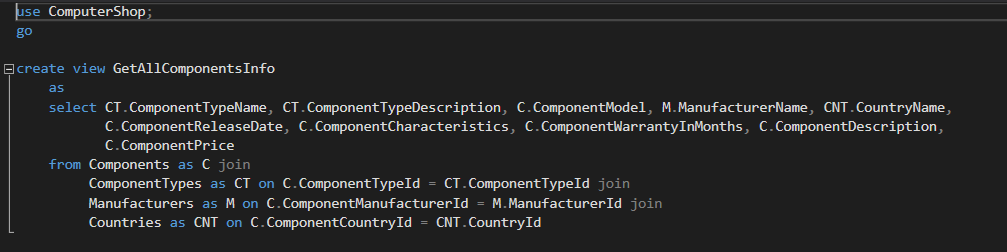


Рисунок 5 – Скрипт, создающий представление данных

Вывод представления выглядит следующим образом (рисунок 6).

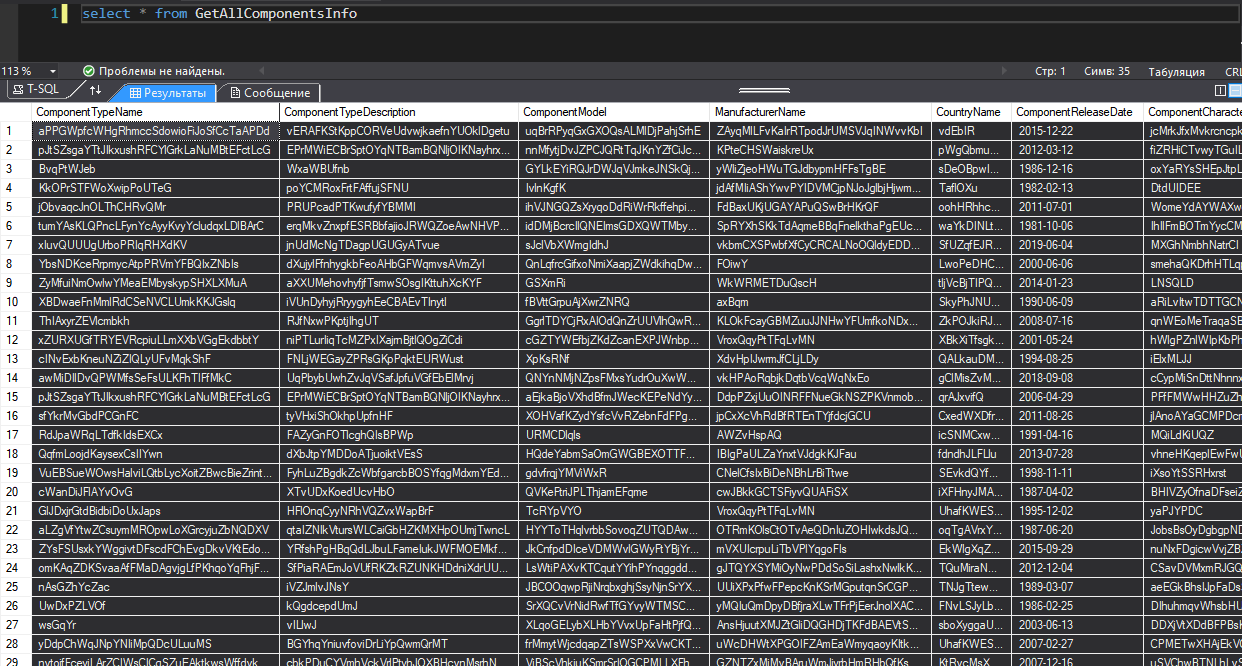


Рисунок 6 – Вывод представления на экран

Задание 4

Далее были созданы хранимые процедуры на вставку и обновление данных всех таблиц базы данных (рисунок 7 – 8).

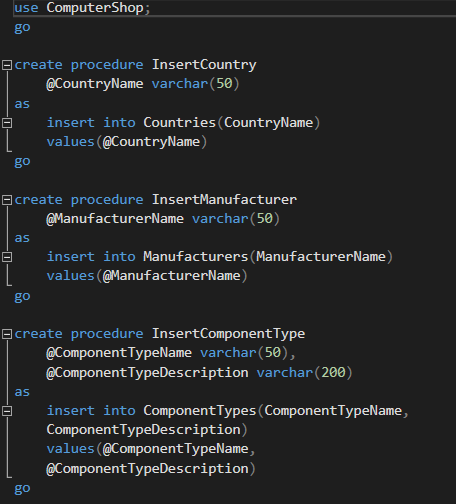


Рисунок 7 – Скрипт для создания хранимых процедур для вставки данных в таблицы

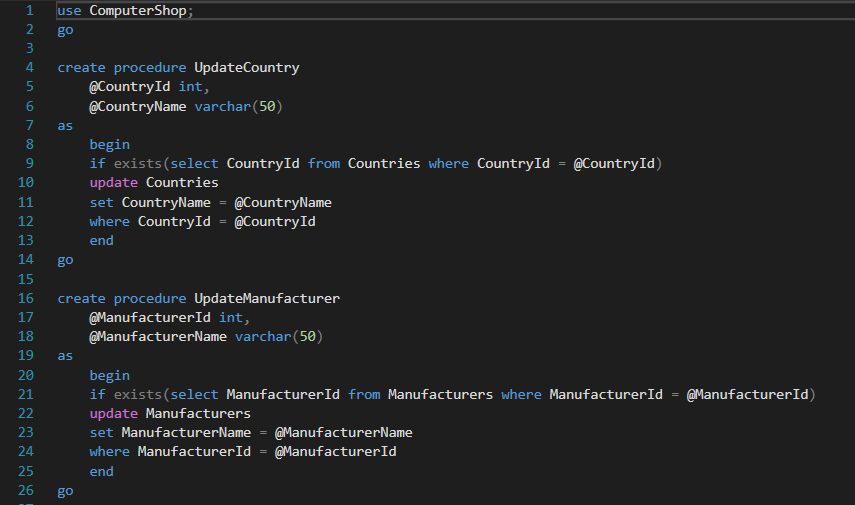


Рисунок 8 – Скрипт для создания хранимых процедур для обновления данных таблиц

**Вывод:**

В ходе работы была разработана база данных, соответствующая заданной предметной области. Написаны скрипты для взаимодействия с данными таблиц этой базы данных.

**ПРИЛОЖЕНИЕ А**

Листинг созданных скриптов

CreateTables.sql

use ComputerShop;

go

create table Countries

(

CountryId int primary key identity,

CountryName varchar(50) not null

)

create table Manufacturers

(

ManufacturerId int primary key identity,

ManufacturerName varchar(50) not null

)

create table ComponentTypes

(

ComponentTypeId int primary key identity,

ComponentTypeName varchar(50) not null,

ComponentTypeDescription varchar(200) not null

)

create table Components

(

ComponentId int primary key identity,

ComponentTypeId int references ComponentTypes(ComponentTypeId) on delete cascade not null,

ComponentModel varchar(50) not null,

ComponentManufacturerId int references Manufacturers(ManufacturerId) on delete cascade not null,

ComponentCountryId int references Countries(CountryId) on delete cascade not null,

ComponentReleaseDate date not null,

ComponentCharacteristics varchar(200) not null,

ComponentWarrantyInMonths int not null,

ComponentDescription varchar(200) not null,

ComponentPrice money not null

)

create table Customers

(

CustomerId int primary key identity,

CustomerFullName varchar(50) not null,

CustomerAddress varchar(100) not null,

CustomerPhoneNumber varchar(10) not null,

CustomerDiscount int not null

)

create table Employees

(

EmployeeId int primary key identity,

EmployeeFullName varchar(50) not null,

EmployeeWorkExperienceInMonth int not null

)

create table Services

(

ServiceId int primary key identity,

ServiceName varchar(50) not null,

ServiceDescription varchar(200) not null,

ServicePrice money not null

)

create table Orders

(

OrderId int primary key identity,

OrderStartDate date not null,

OrderExecutionDate date not null,

OrderCustomerId int references Customers(CustomerId) on delete cascade not null,

OrderPrepayment money not null,

OrderPaid bit not null,

OrderFinished bit not null,

OrderExecutingEmployeeId int references Employees(EmployeeId) on delete cascade not null

)

create table OrderServices

(

OrderServiceId int primary key identity,

OrderId int references Orders(OrderId) on delete cascade not null,

ServiceId int references Services(ServiceId) on delete cascade not null

)

create table OrderComponents

(

OrderComponentId int primary key identity,

OrderId int references Orders(OrderId) on delete cascade not null,

ComponentId int references Components(ComponentId) on delete cascade not null

)

CreateInsertProcedures.sql

use ComputerShop;

go

create procedure InsertCountry

@CountryName varchar(50)

as

insert into Countries(CountryName)

values(@CountryName)

go

create procedure InsertManufacturer

@ManufacturerName varchar(50)

as

insert into Manufacturers(ManufacturerName)

values(@ManufacturerName)

go

create procedure InsertComponentType

@ComponentTypeName varchar(50),

@ComponentTypeDescription varchar(200)

as

insert into ComponentTypes(ComponentTypeName,

ComponentTypeDescription)

values(@ComponentTypeName,

@ComponentTypeDescription)

go

create procedure InsertComponent

@ComponentTypeId int,

@ComponentModel varchar(50),

@ComponentManufacturerId int,

@ComponentCountryId int,

@ComponentReleaseDate date,

@ComponentCharacteristics varchar(200),

@ComponentWarrantyInMonths int,

@ComponentDescription varchar(200),

@ComponentPrice money

as

begin

if exists(select ComponentTypeId from ComponentTypes where ComponentTypeId = @ComponentTypeId) and

exists(select ManufacturerId from Manufacturers where ManufacturerId = @ComponentManufacturerId) and

exists(select CountryId from Countries where CountryId = @ComponentCountryId)

insert into Components(ComponentTypeId,

ComponentModel,

ComponentManufacturerId,

ComponentCountryId,

ComponentReleaseDate,

ComponentCharacteristics,

ComponentWarrantyInMonths,

ComponentDescription,

ComponentPrice)

values(@ComponentTypeId,

@ComponentModel,

@ComponentManufacturerId,

@ComponentCountryId,

@ComponentReleaseDate,

@ComponentCharacteristics,

@ComponentWarrantyInMonths,

@ComponentDescription,

@ComponentPrice)

end

go

create procedure InsertCustomer

@CustomerFullName varchar(50),

@CustomerAddress varchar(100),

@CustomerPhoneNumber varchar(10),

@CustomerDiscount int

as

insert into Customers(CustomerFullName,

CustomerAddress,

CustomerPhoneNumber,

CustomerDiscount)

values(@CustomerFullName,

@CustomerAddress,

@CustomerPhoneNumber,

@CustomerDiscount)

go

create procedure InsertEmployee

@EmployeeFullName varchar(50),

@EmployeeWorkExperienceInMonth int

as

insert into Employees(EmployeeFullName,

EmployeeWorkExperienceInMonth)

values(@EmployeeFullName,

@EmployeeWorkExperienceInMonth)

go

create procedure InsertService

@ServiceName varchar(50),

@ServiceDescription varchar(200),

@ServicePrice money

as

insert into Services(ServiceName,

ServiceDescription,

ServicePrice)

values(@ServiceName,

@ServiceDescription,

@ServicePrice)

go

create procedure InsertOrder

@OrderStartDate date,

@OrderExecutionDate date,

@OrderCustomerId int,

@OrderPrepayment money,

@OrderPaid bit,

@OrderFinished bit,

@OrderExecutingEmployeeId int

as

begin

if exists(select CustomerId from Customers where CustomerId = @OrderCustomerId) and

exists(select EmployeeId from Employees where EmployeeId = @OrderExecutingEmployeeId)

insert into Orders(OrderStartDate,

OrderExecutionDate,

OrderCustomerId,

OrderPrepayment,

OrderPaid,

OrderFinished,

OrderExecutingEmployeeId)

values(@OrderStartDate,

@OrderExecutionDate,

@OrderCustomerId,

@OrderPrepayment,

@OrderPaid,

@OrderFinished,

@OrderExecutingEmployeeId)

end

go

create procedure InsertOrderComponent

@OrderId int,

@ComponentId int

as

begin

if exists(select OrderId from Orders where OrderId = @OrderId) and

exists(select ComponentId from Components where ComponentId = @ComponentId)

insert into OrderComponents(OrderId,

ComponentId)

values(@OrderId,

@ComponentId)

end

go

create procedure InsertOrderService

@OrderId int,

@ServiceId int

as

begin

if exists(select OrderId from Orders where OrderId = @OrderId) and

exists(select ServiceId from Services where ServiceId = @ServiceId)

insert into OrderServices(OrderId,

ServiceId)

values(@OrderId,

@ServiceId)

End

CreateUpdateProcedures.sql

use ComputerShop;

go

create procedure UpdateCountry

@CountryId int,

@CountryName varchar(50)

as

begin

if exists(select CountryId from Countries where CountryId = @CountryId)

update Countries

set CountryName = @CountryName

where CountryId = @CountryId

end

go

create procedure UpdateManufacturer

@ManufacturerId int,

@ManufacturerName varchar(50)

as

begin

if exists(select ManufacturerId from Manufacturers where ManufacturerId = @ManufacturerId)

update Manufacturers

set ManufacturerName = @ManufacturerName

where ManufacturerId = @ManufacturerId

end

go

create procedure UpdateComponentType

@ComponentTypeId int,

@ComponentTypeName varchar(50),

@ComponentTypeDescription varchar(200)

as

begin

if exists(select ComponentTypeId from ComponentTypes where ComponentTypeId = @ComponentTypeId)

update ComponentTypes

set ComponentTypeName = @ComponentTypeName,

ComponentTypeDescription = @ComponentTypeDescription

where ComponentTypeId = @ComponentTypeId

end

go

create procedure UpdateComponent

@ComponentId int,

@ComponentTypeId int,

@ComponentModel varchar(50),

@ComponentManufacturerId int,

@ComponentCountryId int,

@ComponentReleaseDate date,

@ComponentCharacteristics varchar(200),

@ComponentWarrantyInMonths int,

@ComponentDescription varchar(200),

@ComponentPrice money

as

begin

if exists(select ComponentId from Components where ComponentId = @ComponentId) and

exists(select ComponentTypeId from ComponentTypes where ComponentTypeId = @ComponentTypeId) and

exists(select ManufacturerId from Manufacturers where ManufacturerId = @ComponentManufacturerId) and

exists(select CountryId from Countries where CountryId = @ComponentCountryId)

update Components

set ComponentTypeId = @ComponentTypeId,

ComponentModel = @ComponentModel,

ComponentManufacturerId = @ComponentManufacturerId,

ComponentCountryId = @ComponentCountryId,

ComponentReleaseDate = @ComponentReleaseDate,

ComponentCharacteristics = @ComponentCharacteristics,

ComponentWarrantyInMonths = @ComponentWarrantyInMonths,

ComponentDescription = @ComponentDescription,

ComponentPrice = @ComponentPrice

where ComponentId = @ComponentId

end

go

create procedure UpdateCustomer

@CustomerId int,

@CustomerFullName varchar(50),

@CustomerAddress varchar(100),

@CustomerPhoneNumber varchar(10),

@CustomerDiscount int

as

begin

if exists(select CustomerId from Customers where CustomerId = @CustomerId)

update Customers

set CustomerFullName = @CustomerFullName,

CustomerAddress = @CustomerAddress,

CustomerPhoneNumber = @CustomerPhoneNumber,

CustomerDiscount = @CustomerDiscount

where CustomerId = @CustomerId

end

go

create procedure UpdateEmployee

@EmployeeId int,

@EmployeeFullName varchar(50),

@EmployeeWorkExperienceInMonth int

as

begin

if exists(select EmployeeId from Employees where EmployeeId = @EmployeeId)

update Employees

set EmployeeFullName = @EmployeeFullName,

EmployeeWorkExperienceInMonth = @EmployeeWorkExperienceInMonth

where EmployeeId = @EmployeeId

end

go

create procedure UpdateService

@ServiceId int,

@ServiceName varchar(50),

@ServiceDescription varchar(200),

@ServicePrice money

as

begin

if exists(select ServiceId from Services where ServiceId = @ServiceId)

update Services

set ServiceName = @ServiceName,

ServiceDescription = @ServiceDescription,

ServicePrice = @ServicePrice

where ServiceId = @ServiceId

end

go

create procedure UpdateOrder

@OrderId int,

@OrderStartDate date,

@OrderExecutionDate date,

@OrderCustomerId int,

@OrderPrepayment money,

@OrderPaid bit,

@OrderFinished bit,

@OrderExecutingEmployeeId int

as

begin

if exists(select OrderId from Orders where OrderId = @OrderId) and

exists(select CustomerId from Customers where CustomerId = @OrderCustomerId) and

exists(select EmployeeId from Employees where EmployeeId = @OrderExecutingEmployeeId)

update Orders

set OrderStartDate = @OrderStartDate,

OrderExecutionDate = @OrderExecutionDate,

OrderCustomerId = @OrderCustomerId,

OrderPrepayment = @OrderPrepayment,

OrderPaid = @OrderPaid,

OrderFinished = @OrderFinished,

OrderExecutingEmployeeId = @OrderExecutingEmployeeId

where OrderId = @OrderId

end

go

create procedure UpdateOrderComponent

@OrderComponentId int,

@OrderId int,

@ComponentId int

as

begin

if exists(select OrderComponentId from OrderComponents where OrderComponentId = @OrderComponentId) and

exists(select OrderId from Orders where OrderId = @OrderId) and

exists(select ComponentId from Components where ComponentId = @ComponentId)

update OrderComponents

set OrderId = @OrderId,

ComponentId = @ComponentId

where OrderComponentId = @OrderComponentId

end

go

create procedure UpdateOrderService

@OrderServiceId int,

@OrderId int,

@ServiceId int

as

begin

if exists(select OrderServiceId from OrderServices where OrderServiceId = @OrderServiceId) and

exists(select OrderId from Orders where OrderId = @OrderId) and

exists(select ServiceId from Services where ServiceId = @ServiceId)

update OrderServices

set OrderId = @OrderId,

ServiceId = @ServiceId

where OrderServiceId = @OrderServiceId

end

CreateViews.sql

use ComputerShop;

go

create view GetAllComponentsInfo

as

select CT.ComponentTypeName, CT.ComponentTypeDescription, C.ComponentModel, M.ManufacturerName, CNT.CountryName,

C.ComponentReleaseDate, C.ComponentCharacteristics, C.ComponentWarrantyInMonths, C.ComponentDescription,

C.ComponentPrice

from Components as C join

ComponentTypes as CT on C.ComponentTypeId = CT.ComponentTypeId join

Manufacturers as M on C.ComponentManufacturerId = M.ManufacturerId join

Countries as CNT on C.ComponentCountryId = CNT.CountryId

InitializeTables.sql

use ComputerShop;

go

declare @ReferenceRecordsCount int = 500,

@OperationalRecordsCount int = 20000

declare @Symbol char(52)= 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',

@Digit char(10) = '0123456789',

@Position int,

@i int,

@NameLimit int,

@RowCount int,

@MinNumberSymbols int = 5,

@MaxNumberSymbols int = 50

declare @ComponentTypesCount int = @ReferenceRecordsCount,

@ManufacturersCount int = @ReferenceRecordsCount,

@CountriesCount int = @ReferenceRecordsCount,

@ComponentsCount int = @OperationalRecordsCount,

@CustomersCount int = @ReferenceRecordsCount,

@EmployeesCount int = @ReferenceRecordsCount,

@ServicesCount int = @ReferenceRecordsCount,

@OrdersCount int = @OperationalRecordsCount,

@OrderComponentsCount int = @ReferenceRecordsCount,

@OrderServicesCount int = @ReferenceRecordsCount

begin tran

-- Заполнение таблицы Countries

declare @CountryName varchar(50)

set @RowCount = 1

while @RowCount <= @CountriesCount

begin

set @NameLimit=@MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @CountryName = ''

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @CountryName = @CountryName + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertCountry @CountryName

set @RowCount += 1

end

-- Заполнение таблицы Manufacturers

declare @ManufacturerName varchar(50)

set @RowCount = 1

while @RowCount <= @ManufacturersCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @ManufacturerName = ''

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @ManufacturerName = @ManufacturerName + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertManufacturer @ManufacturerName

set @RowCount += 1

end

-- Заполнение таблицы ComponentTypes

declare @ComponentTypeName varchar(50),

@ComponentTypeDescription varchar(200)

set @RowCount = 1

while @RowCount <= @ComponentTypesCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @ComponentTypeName = ''

set @ComponentTypeDescription = ''

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @ComponentTypeName = @ComponentTypeName + substring(@Symbol, @Position, 1)

set @Position = rand() \* 52

set @ComponentTypeDescription = @ComponentTypeDescription + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertComponentType @ComponentTypeName, @ComponentTypeDescription

set @RowCount += 1

end

-- Заполнение таблицы Components

declare @ComponentTypeId int,

@ComponentModel varchar(50),

@ComponentManufacturerId int,

@ComponentCountryId int,

@ComponentReleaseDate date,

@ComponentCharacteristics varchar(50),

@ComponentWarrantyInMonths int,

@ComponentDescription varchar(50),

@ComponentPrice money

set @RowCount = 1

while @RowCount <= @ComponentsCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @ComponentTypeId = rand() \* @ComponentTypesCount

set @ComponentModel = ''

set @ComponentManufacturerId = rand() \* @ManufacturersCount

set @ComponentCountryId = rand() \* @CountriesCount

set @ComponentReleaseDate = dateadd(day,-RAND()\*15000,GETDATE())

set @ComponentCharacteristics = ''

set @ComponentWarrantyInMonths = 6 + rand() \* 30 -- от 6 до 36

set @ComponentDescription = ''

set @ComponentPrice = 100 + rand() \* 2900 -- от 100 до 3000

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @ComponentModel = @ComponentModel + substring(@Symbol, @Position, 1)

set @Position = rand() \* 52

set @ComponentCharacteristics = @ComponentCharacteristics + substring(@Symbol, @Position, 1)

set @Position = rand() \* 52

set @ComponentDescription = @ComponentDescription + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertComponent @ComponentTypeId, @ComponentModel, @ComponentManufacturerId, @ComponentCountryId,

@ComponentReleaseDate, @ComponentCharacteristics, @ComponentWarrantyInMonths,

@ComponentDescription, @ComponentPrice

set @RowCount += 1

end

-- Заполнение таблицы Customers

declare @CustomerFullName varchar(50),

@CustomerAddress varchar(100),

@CustomerPhoneNumber varchar(10),

@CustomerDiscount int

set @RowCount = 1

while @RowCount <= @CustomersCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @CustomerFullName = ''

set @CustomerAddress = ''

set @CustomerPhoneNumber = ''

set @CustomerDiscount = rand() \* 50

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @CustomerFullName = @CustomerFullName + substring(@Symbol, @Position, 1)

set @Position = rand() \* 52

set @CustomerAddress = @CustomerAddress + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

set @i = 1

while @i <= 10

begin

set @Position = rand() \* 10

set @CustomerPhoneNumber = @CustomerPhoneNumber + substring(@Digit, @Position, 1)

set @i = @i + 1

end

exec InsertCustomer @CustomerFullName, @CustomerAddress, @CustomerPhoneNumber, @CustomerDiscount

set @RowCount += 1

end

-- Заполнение таблицы Employees

declare @EmployeeFullName varchar(50),

@EmployeeWorkExperienceInMonth int

set @RowCount = 1

while @RowCount <= @EmployeesCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @EmployeeFullName = ''

set @EmployeeWorkExperienceInMonth = rand() \* 480

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @EmployeeFullName = @EmployeeFullName + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertEmployee @EmployeeFullName, @EmployeeWorkExperienceInMonth

set @RowCount += 1

end

-- Заполнение таблицы Services

declare @ServiceName varchar(50),

@ServiceDescription varchar(200),

@ServicePrice money

set @RowCount = 1

while @RowCount <= @ServicesCount

begin

set @NameLimit = @MinNumberSymbols + rand() \* (@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

set @i = 1

set @ServiceName = ''

set @ServiceDescription = ''

set @ServicePrice = 300 + rand() \* 2700 -- от 300 до 3000

while @i <= @NameLimit

begin

set @Position = rand() \* 52

set @ServiceName = @ServiceName + substring(@Symbol, @Position, 1)

set @Position = rand() \* 52

set @ServiceDescription = @ServiceDescription + substring(@Symbol, @Position, 1)

set @i = @i + 1

end

exec InsertService @ServiceName, @ServiceDescription, @ServicePrice

set @RowCount += 1

end

-- Заполнение таблицы Orders

declare @OrderStartDate date,

@OrderExecutionDate date,

@OrderCustomerId int,

@OrderPrepayment money,

@OrderPaid bit,

@OrderFinished bit,

@OrderExecutingEmployeeId int

set @RowCount = 1

while @RowCount <= @OrdersCount

begin

set @OrderStartDate = dateadd(day,-(15000 + RAND()\*5000),GETDATE()) -- 15000 - 20000

set @OrderExecutionDate = dateadd(day,-(5000 + RAND()\*10000),GETDATE()) -- 5000 - 15000

set @OrderCustomerId = rand() \* @CustomersCount

set @OrderPrepayment = 300 + rand() \* 2700

set @OrderPaid = cast(round(rand(), 0) as bit)

set @OrderFinished = cast(round(rand(), 0) as bit)

set @OrderExecutingEmployeeId = rand() \* @EmployeesCount

exec InsertOrder @OrderStartDate, @OrderExecutionDate, @OrderCustomerId, @OrderPrepayment, @OrderPaid, @OrderFinished,

@OrderExecutingEmployeeId

set @RowCount += 1

end

-- Заполнение таблицы OrderComponents

declare @OrderId int,

@ComponentId int

set @RowCount = 1

while @RowCount <= @OrderComponentsCount

begin

set @OrderId = rand() \* @OrdersCount

set @ComponentId = rand() \* @ComponentsCount

exec InsertOrderComponent @OrderId, @ComponentId

set @RowCount += 1

end

-- Заполнение таблицы OrderServices

declare @ServiceId int

set @RowCount = 1

while @RowCount <= @OrderServicesCount

begin

set @OrderId = rand() \* @OrdersCount

set @ServiceId = rand() \* @ServicesCount

exec InsertOrderService @OrderId, @ServiceId

set @RowCount += 1

end

commit tran