**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине: «Разработка приложений баз данных для информационных систем»

на тему: «Разработка серверной части информационной системы

в СУБД *MS SQL Server*»

Выполнил: студент гр. ИТП-31

Демиденко Д. В.

Принял: ректор

Асенчик О. Д.

Гомель 2023

**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 указан на рисунке 1.

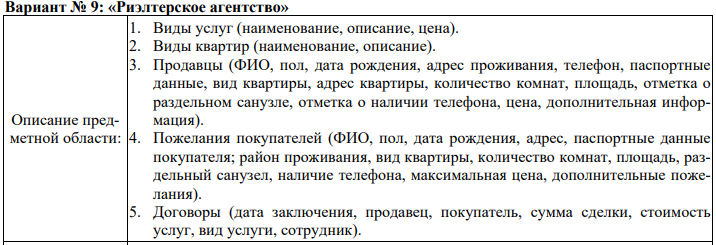


Рисунок 1 – Вариант задания

**Ход работы**

В начале выполнения задания была спроектирована база с условием выполнения трех форма нормализации. Первой была реализована таблица, которая хранит данные о квартирах. Для хранения информации о недвижимости была создана отдельная таблица *Apartments*.Эта таблица содержит данные о квартирах, такие как их наименование, описание, количество комнат, площадь, наличие раздельного санузла, наличие телефона, максимальная цена и дополнительные пожелания. Так же эта таблица содержит дополнительные пожелания клиентов. Пример схемы этой таблицы указан на рисунке 2. Листинг создания этой таблицы указана в приложении А.

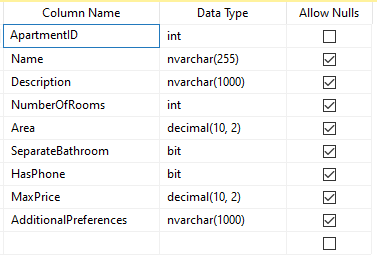


Рисунок 2 – Пример таблицы *Apartments*

Далее для хранения информации о заключенных договорах на продажу квартир была создана таблица *Contracts*. В этой таблице записываются данные о договорах, включая дату заключения, информацию о продавце и покупателе, сумму сделки, стоимость услуги и информацию о сотруднике, заключившем договор. Пример схемы таблицы *Contracts* указана на рисунке 3. Листинг создания этой таблицы указана в приложении А.

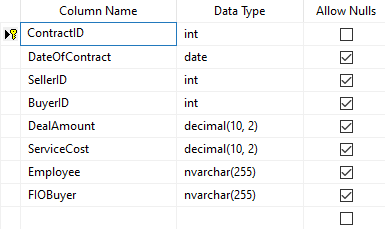


Рисунок 3 – Пример таблицы *Contracts*

Далее была реализована таблица *Sellers*. В этой таблице содержатся информация о продавцах квартир, включая их ФИО, пол, дату рождения, адрес проживания, контактный телефон и паспортные данные. Пример схемы таблицы *Sellers* указан на рисунке 4. Листинг создания этой таблицы указана в приложении А.

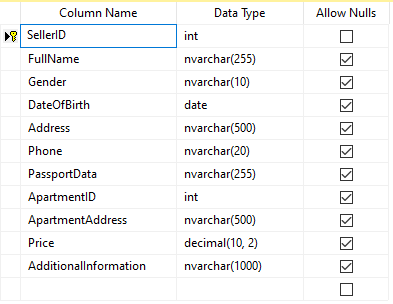


Рисунок 4 – Пример таблицы *Sellers*

Далее была разработана таблица *Services* которая хранит информацию о доступных услугах. Эта таблица содержит данные о различных услугах, такие как их наименование, описание и цена. Пример схемы таблицы *Services* указан на рисунке 5. Листинг создания этой таблицы указана в приложении А.

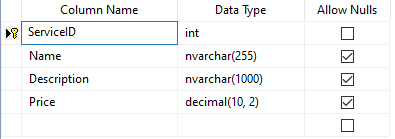


Рисунок 5 – Пример таблицы *Services*

Далее была разработана вспомогательная таблица *ContractServices* которая хранит в себе информация о клиентах. Для этой таблицы была создана вспомогательная таблица *Addresses*, которая хранит в себе данные об адресах клиентов. Первый столбец этой таблицы хранит город, второй улицу, третий хранит номер дома и четвертый столбец хранит в себе номер комнаты клиента. Пример схемы таблицы *Addresses* указан на рисунке 6. Листинг создания этой таблицы указана в приложении А.

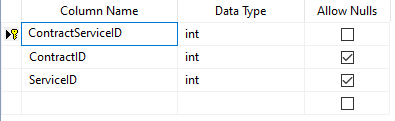


Рисунок 6 – Пример таблицы *ContractServices*

После создания всех таблиц баз данных были установлены все связи между ними. Также для необходимых столбцов базы были установлены необходимые ограничения. Пример диаграммы базы данных указан на рисунке 7.

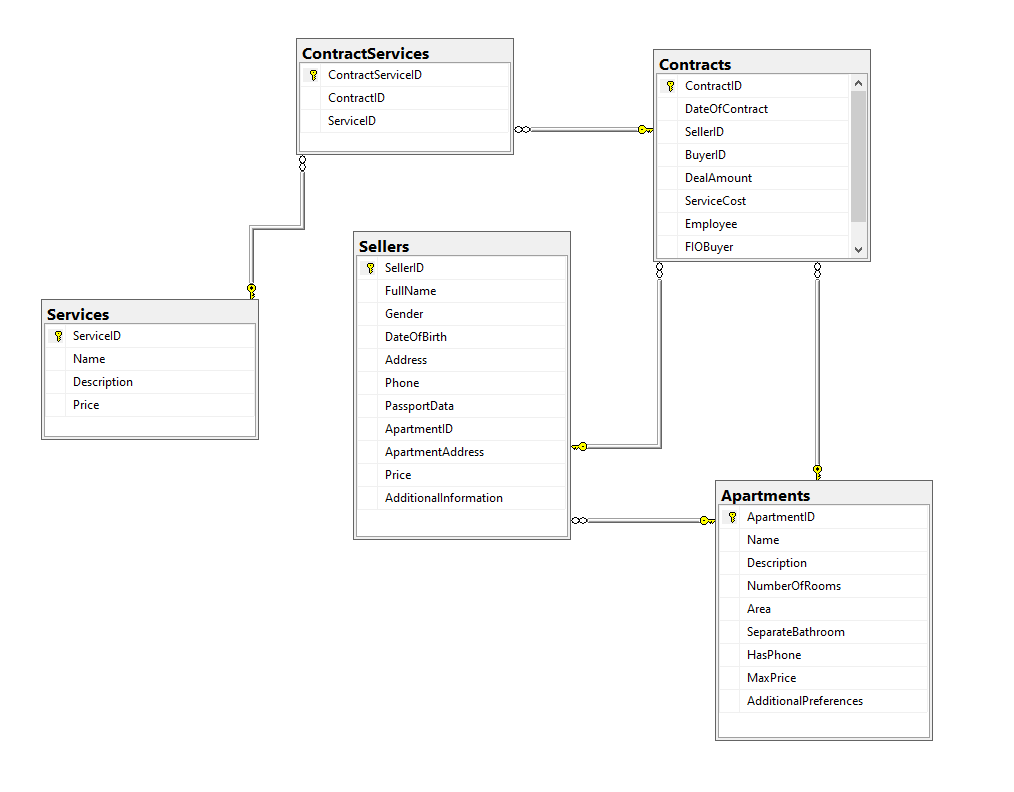


Рисунок 7 – Пример диаграммы базы данных

После написания всех скриптов для создания таблиц был написан скрипт для заполнения таблиц. Пример данного скрипта указан в приложение А. Далее были созданы три представления для удобного получения доступа к данным из таблицы.

Первым было создано представления для получения всех данных из таблицы *Contracts*. Пример работы данного представления указан на рисунке 8. Листинг кода по созданию этого представления указан в приложении А.

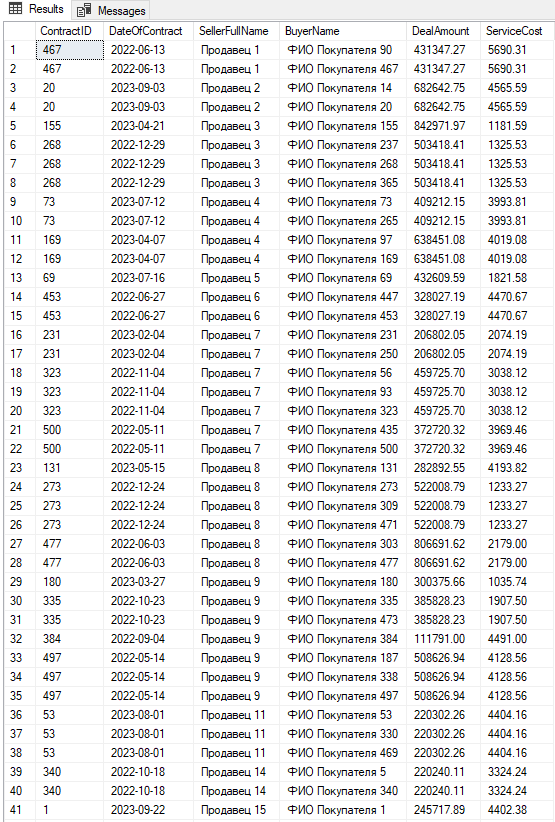


Рисунок 8 – Пример работы представления *ContractView*

Вторым было создано представления для получения всех данных из таблицы *Apartments*. Пример работы данного представления указан на рисунке 9. Листинг кода по созданию этого представления указан в приложении А.

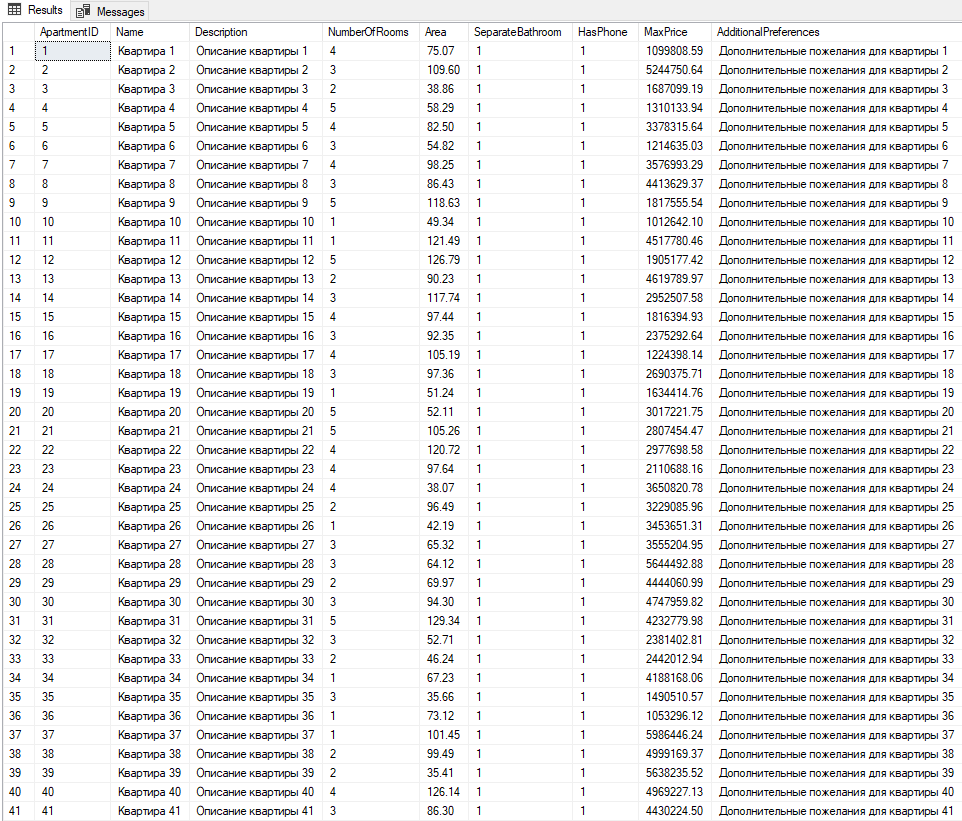


Рисунок 9 – Пример работы представления *ClientsView*

Третьим было создано представления для получения всех данных из таблицы *Services*. Пример работы данного представления указан на рисунке 10. Листинг кода по созданию этого представления указан в приложении А.

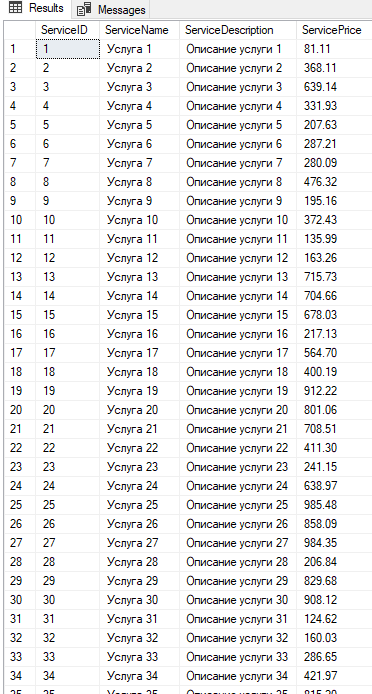


Рисунок 10 – Пример работы представления *ServiceView*

Далее были реализованы три хранимы процедуры для добавления новых записей в таблицы *Apartments*, *Sellers*, *Service*. Листинг данных хранимых процедур указан в приложении А.

**Вывод**: в ходе выполнения лабораторной работы была спроектирована и разработана база данных на платформе *MS SQL Server* для серверной части клиент-серверного приложения. Были созданы таблицы баз данных, которые далее были заполнены значениями. Также были разработаны три хранимые процедуры для вставки новых данных в таблицы. И три представления для получения удобного доступа к данным из базы. Ссылка на *GitHub* репозиторий - https://github.com/ddemidenko/RPBDIS.git.

**ПРИЛОЖНИЕ А**

Листинг создание базы данных и таблиц

CREATE DATABASE RealEstateAgency;

USE RealEstateAgency;

CREATE TABLE Apartments (

ApartmentID INT PRIMARY KEY,

Name NVARCHAR(255),

Description NVARCHAR(1000),

NumberOfRooms INT,

Area DECIMAL(10, 2),

SeparateBathroom BIT,

HasPhone BIT,

MaxPrice DECIMAL(10, 2),

AdditionalPreferences NVARCHAR(1000)

);

CREATE TABLE Sellers (

SellerID INT PRIMARY KEY,

FullName NVARCHAR(255),

Gender NVARCHAR(10),

DateOfBirth DATE,

Address NVARCHAR(500),

Phone NVARCHAR(20),

PassportData NVARCHAR(255),

ApartmentID INT,

ApartmentAddress NVARCHAR(500),

Price DECIMAL(10, 2),

AdditionalInformation NVARCHAR(1000),

FOREIGN KEY (ApartmentID) REFERENCES Apartments(ApartmentID)

);

CREATE TABLE Contracts (

ContractID INT PRIMARY KEY,

DateOfContract DATE,

SellerID INT,

BuyerID INT,

DealAmount DECIMAL(10, 2),

ServiceCost DECIMAL(10, 2),

Employee NVARCHAR(255),

FOREIGN KEY (SellerID) REFERENCES Sellers(SellerID),

FOREIGN KEY (BuyerID) REFERENCES Apartments(ApartmentID)

);

CREATE TABLE Services (

ServiceID INT PRIMARY KEY,

Name NVARCHAR(255),

Description NVARCHAR(1000),

Price DECIMAL(10, 2)

);

CREATE TABLE ContractServices (

ContractServiceID INT PRIMARY KEY,

ContractID INT,

ServiceID INT,

FOREIGN KEY (ContractID) REFERENCES Contracts(ContractID),

FOREIGN KEY (ServiceID) REFERENCES Services(ServiceID)

);

Листинг заполнения таблиц записями

Таблица *Apartments*

DECLARE @i INT = 1;

WHILE @i <= 20000

BEGIN

INSERT INTO Apartments (ApartmentID, Name, Description, NumberOfRooms, Area, SeparateBathroom, HasPhone, MaxPrice, AdditionalPreferences)

VALUES

(@i, 'Квартира ' + CAST(@i AS NVARCHAR(10)), 'Описание квартиры ' + CAST(@i AS NVARCHAR(10)),

RAND() \* 5 + 1, RAND() \* 100 + 30, CAST(RAND() \* 1 AS BIT), CAST(RAND() \* 1 AS BIT), RAND() \* 5000000 + 1000000, 'Дополнительные пожелания для квартиры ' + CAST(@i AS NVARCHAR(10)));

SET @i = @i + 1;

END;

Таблица *Contracts*

DECLARE @k INT = 1;

WHILE @k <= 500

BEGIN

INSERT INTO Contracts (ContractID, DateOfContract, SellerID, BuyerID, DealAmount, ServiceCost, Employee)

VALUES

(@k, DATEADD(DAY, -@k, GETDATE()),

CAST(RAND() \* 500 + 1 AS INT),

CAST(RAND() \* 500 + 1 AS INT),

RAND() \* 500000 + 50000,

RAND() \* 5000 + 1000,

'Сотрудник ' + CAST(@k AS NVARCHAR(10)));

SET @k = @k + 1;

END;

Таблица *ContractsService*

DECLARE @j INT = 1;

WHILE @j <= 2000

BEGIN

INSERT INTO ContractServices (ContractServiceID, ContractID, ServiceID)

VALUES

(@j, CAST(RAND() \* 500 + 1 AS INT), CAST(RAND() \* 500 + 1 AS INT));

SET @j = @j + 1;

END;

Таблица *Sellers*

DECLARE @i INT = 1;

WHILE @i <= 500

BEGIN

INSERT INTO Sellers (SellerID, FullName, Gender, DateOfBirth, Address, Phone, PassportData, ApartmentID, ApartmentAddress, Price, AdditionalInformation)

VALUES

(@i, 'Продавец ' + CAST(@i AS NVARCHAR(10)),

CASE WHEN @i % 2 = 0 THEN 'Мужской' ELSE 'Женский' END,

DATEADD(DAY, -(@i \* 10), GETDATE()),

'Адрес продавца ' + CAST(@i AS NVARCHAR(10)),

'123456789' + RIGHT('000' + CAST(@i AS NVARCHAR(3)), 3),

'ABCD' + RIGHT('0000' + CAST(@i AS NVARCHAR(4)), 4),

CAST(RAND() \* 20000 + 1 AS INT),

'Адрес квартиры продавца ' + CAST(@i AS NVARCHAR(10)),

RAND() \* 500000 + 50000,

'Дополнительная информация для продавца ' + CAST(@i AS NVARCHAR(10)));

SET @i = @i + 1;

END;

Таблица *Services*

DECLARE @i INT = 1;

WHILE @i <= 500

BEGIN

INSERT INTO Services (ServiceID, Name, Description, Price)

VALUES

(@i, 'Услуга ' + CAST(@i AS NVARCHAR(10)), 'Описание услуги ' + CAST(@i AS NVARCHAR(10)), RAND() \* 1000);

SET @i = @i + 1;

END;

Листинг хранимых процедур

*InsertApartment*

USE RealEstateAgency;

GO

CREATE PROCEDURE InsertApartment

@Name NVARCHAR(255),

@Description NVARCHAR(1000),

@NumberOfRooms INT,

@Area DECIMAL(10, 2),

@SeparateBathroom BIT,

@HasPhone BIT,

@MaxPrice DECIMAL(10, 2),

@AdditionalPreferences NVARCHAR(1000)

AS

BEGIN

INSERT INTO Apartments (Name, Description, NumberOfRooms, Area, SeparateBathroom, HasPhone, MaxPrice, AdditionalPreferences)

VALUES (@Name, @Description, @NumberOfRooms, @Area, @SeparateBathroom, @HasPhone, @MaxPrice, @AdditionalPreferences)

END;

*InsertSeller*

CREATE PROCEDURE InsertSeller

@FullName NVARCHAR(255),

@Gender NVARCHAR(10),

@Birthdate DATE,

@Address NVARCHAR(500),

@PhoneNumber NVARCHAR(15),

@PassportData NVARCHAR(255)

AS

BEGIN

INSERT INTO Sellers (FullName, Gender, DateOfBirth, Address, Phone, PassportData)

VALUES (@FullName, @Gender, @BirthDate, @Address, @PhoneNumber, @PassportData)

END;

*InsertService*

CREATE PROCEDURE InsertService

@Name NVARCHAR(255),

@Description NVARCHAR(1000),

@Price DECIMAL(10, 2)

AS

BEGIN

INSERT INTO Services (Name, Description, Price)

VALUES (@Name, @Description, @Price)

END;

Листинг представлений

*ApartmentView*

CREATE VIEW ApartmentView

AS

SELECT

ApartmentID,

Name,

Description,

NumberOfRooms,

Area,

SeparateBathroom,

HasPhone,

MaxPrice,

AdditionalPreferences

FROM

Apartments;

*ContractView*

CREATE VIEW ContractView

AS

SELECT

c.ContractID,

c.DateOfContract,

s.FullName AS SellerFullName,

b.FIOBuyer AS BuyerName,

c.DealAmount,

c.ServiceCost

FROM

Contracts c

JOIN

Sellers s ON c.SellerID = s.SellerID

JOIN

Contracts b ON c.BuyerID = b.BuyerID;

*ServiceView*

CREATE VIEW ServiceView

AS

SELECT

ServiceID,

Name AS ServiceName,

Description AS ServiceDescription,

Price AS ServicePrice

FROM

Services;