**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

Ермолин Д.К.

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.

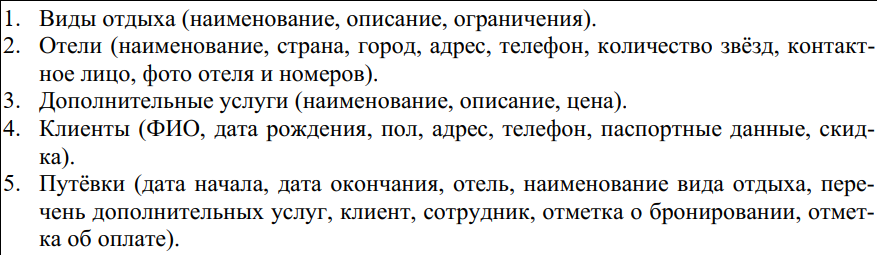
****

Рисунок 1 – Вариант задания

**Ход работы**

В начале процесса проектирования базы данных была создана таблица с названием *"* *AdditionalServices",* которая разработана с целью добавления дополнительных услуг для клиента в отеле. Пример таблицы указан на рисунке 2. Листинг создания этой таблицы указана в приложении А.



Рисунок 2 – Пример таблицы *AdditionalServices*

Далее была создана таблица с названием *"Hotels"*, в которой содержится информация о названии отеля, страна, улица, адрес, телефон, количество звёзд, контактное лицо и фото. Пример таблицы указан на рисунке 3. Листинг создания этой таблицы указана в приложении А.

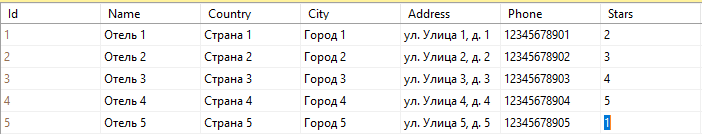


Рисунок 3 – Пример таблицы *Hotels*

После была разработана таблица *"Clients"*, в которой хранилась информация о ФИО, дата рождения, пол, адрес, телефон, паспортные данные, скидка. Пример таблицы указан на рисунке 4. Листинг создания этой таблицы указана в приложении А.

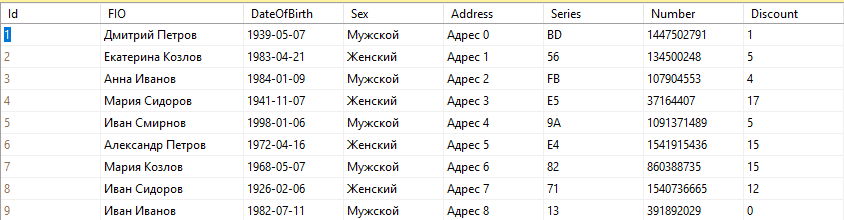
****

Рисунок 4 – Пример таблицы *Clients*

Далее была создана таблица с названием *"Employees"*, в которой содержится информация о ФИО, должности и возрасте. Пример таблицы указан на рисунке 5. Листинг создания этой таблицы указана в приложении А.

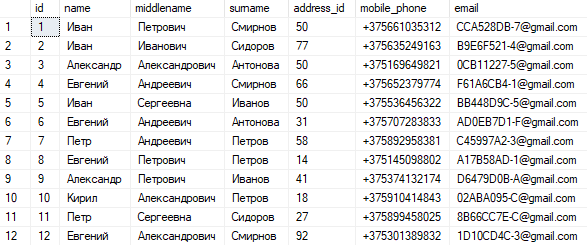
****

Рисунок 5 – Пример таблицы *Employees*

В дальнейшей работе была создана таблица *"TypesOfRecreation"*. Эта таблица содержит информацию о наименование, описание, ограничения. Пример таблицы указан на рисунке 6. Листинг создания этой таблицы указана в приложении А.

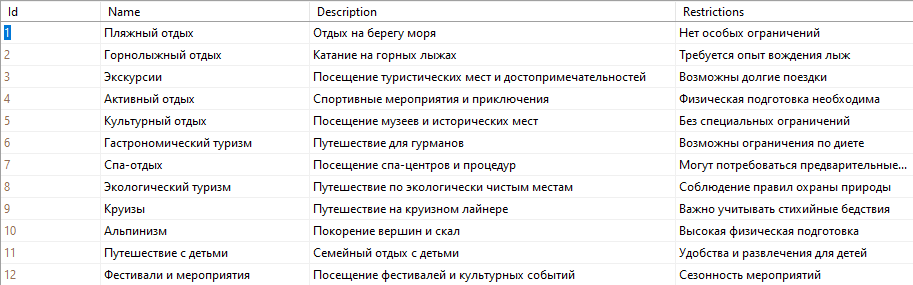


Рисунок 6 – Пример таблицы *TypesOfRecreation*

Далее, в ходе проектирования базы данных, была разработана таблица с названием *"Vouchers"*, предназначенная для хранения информации о дате начала, дате окончания, отель, наименование вида отдыха, перечень дополнительных услуг, клиент, сотрудник, отметка о бронировании, отметка об оплате. Пример таблицы указан на рисунке 7. Листинг создания этой таблицы указана в приложении А.

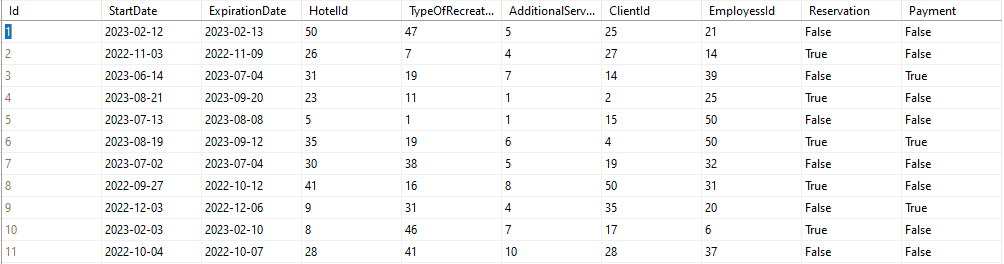


Рисунок 7 – Пример таблицы *Vouchers*

После создания базы данных, была создана диаграмма, которая иллюстрирует зависимости между таблицами. Пример диаграммы базы данных указан на рисунке 8.

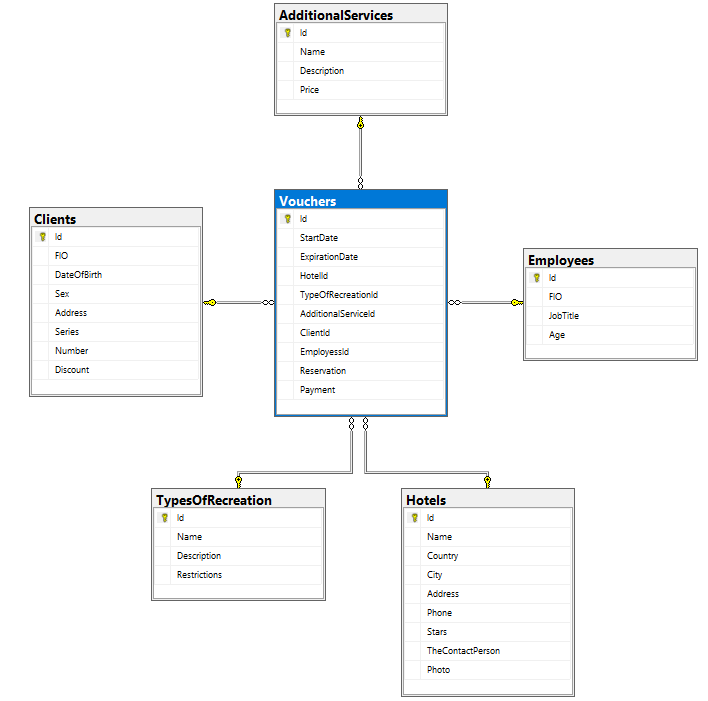


Рисунок 8 – Пример диаграммы базы данных

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 *"Hotels"*. Пример работы данного представления указан на рисунке 9. Листинг кода по созданию этого представления указан в приложении А.

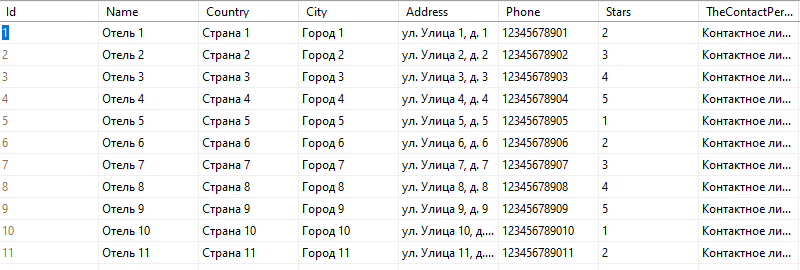


Рисунок 9 – Пример работы представления *HotelView*

Вторым было создано представления для получения данных из таблицы *"Employees"*, где находило работников старше 30 лет. Пример работы данного представления указан на рисунке 10. Листинг кода по созданию этого представления указан в приложении А.

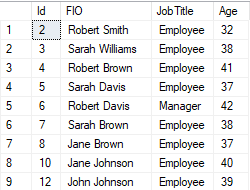


Рисунок 10 – Пример работы представления *EmployeesOver30*

Третьим было создано представления для получения всех данных о путёвках. Пример работы данного представления указан на рисунке 11. Листинг кода по созданию этого представления указан в приложении А.



Рисунок 11 – Пример работы представления *VoucherView*

Далее были реализованы пять хранимых процедур для добавления новых записей в таблицу *Hotels,* обновления записей в таблицу *Clients* и удаления данных по *Id* в таблице *AdditionalServices*. Листинг данных хранимых процедур указан в приложении А.

**Вывод**: в ходе выполнения лабораторной работы была успешно разработана и настроена база данных в среде *MS SQL Server*, предназначенная для использования в серверной части клиент-серверного приложения. Эта база данных включает в себя необходимые таблицы, которые были заполнены данными, а также были созданы хранимые процедуры, обеспечивающие возможность вставки новых записей в эти таблицы.

**ПРИЛОЖНИЕ А**

Листинг скрипта для генерации базы данных

USE master;

GO

IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = 'TouristAgency1')

BEGIN

CREATE DATABASE TouristAgency1;

END

GO

USE TouristAgency1;

GO

-- Таблица для дополнительных услуг

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'AdditionalServices')

BEGIN

CREATE TABLE [dbo].[AdditionalServices](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Description] [nvarchar](200) NOT NULL,

[Price] [money] NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

END

-- Таблица для сотрудников

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'Employees')

BEGIN

CREATE TABLE [dbo].[Employees](

[Id] [int] IDENTITY(1,1) NOT NULL,

[FIO] [nvarchar](150) NOT NULL,

[JobTitle] [nvarchar](50) NOT NULL,

[Age] [int] NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

END

-- Таблица для видов отдыха

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'TypesOfRecreation')

BEGIN

CREATE TABLE [dbo].[TypesOfRecreation](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Description] [nvarchar](100) NOT NULL,

[Restrictions] [nvarchar](50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

END

-- Таблица для клиентов

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'Clients')

BEGIN

CREATE TABLE [dbo].[Clients](

[Id] [int] IDENTITY(1,1) NOT NULL,

[FIO] [nvarchar](150) NOT NULL,

[DateOfBirth] [date] NOT NULL,

[Sex] [nvarchar](50) NOT NULL,

[Address] [nvarchar](100) NOT NULL,

[Series] [nvarchar](50) NOT NULL,

[Number] [bigint] NOT NULL,

[Discount] [bigint] NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

END

-- Таблица для отелей

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'Hotels')

BEGIN

CREATE TABLE [dbo].[Hotels](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Country] [nvarchar](50) NOT NULL,

[City] [nvarchar](50) NOT NULL,

[Address] [nvarchar](100) NOT NULL,

[Phone] [nvarchar](20) NOT NULL,

[Stars] [int] NOT NULL,

[TheContactPerson] [nvarchar](100) NOT NULL,

[Photo] [image] NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

END

-- Таблица для ваучеров

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo' AND TABLE\_NAME = 'Vouchers')

BEGIN

CREATE TABLE [dbo].[Vouchers](

[Id] [int] IDENTITY(1,1) NOT NULL,

[StartDate] [date] NOT NULL,

[ExpirationDate] [date] NOT NULL,

[HotelId] [int] NOT NULL,

[TypeOfRecreationId] [int] NOT NULL,

[AdditionalServiceId] [int] NOT NULL,

[ClientId] [int] NOT NULL,

[EmployessId] [int] NOT NULL,

[Reservation] [bit] NOT NULL,

[Payment] [bit] NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

ALTER TABLE [dbo].[Vouchers] WITH CHECK ADD CONSTRAINT [FK\_Vouchers\_AdditionalServices] FOREIGN KEY([AdditionalServiceId])

REFERENCES [dbo].[AdditionalServices] ([Id]);

ALTER TABLE [dbo].[Vouchers] WITH CHECK ADD CONSTRAINT [FK\_Vouchers\_Clients] FOREIGN KEY([ClientId])

REFERENCES [dbo].[Clients] ([Id]);

ALTER TABLE [dbo].[Vouchers] WITH CHECK ADD CONSTRAINT [FK\_Vouchers\_Employees] FOREIGN KEY([EmployessId])

REFERENCES [dbo].[Employees] ([Id]);

ALTER TABLE [dbo].[Vouchers] WITH CHECK ADD CONSTRAINT [FK\_Vouchers\_Hotels] FOREIGN KEY([HotelId])

REFERENCES [dbo].[Hotels] ([Id]);

ALTER TABLE [dbo].[Vouchers] WITH CHECK ADD CONSTRAINT [FK\_Vouchers\_TypesOfRecreation] FOREIGN KEY([TypeOfRecreationId])

REFERENCES [dbo].[TypesOfRecreation] ([Id]);

END

-- Проверяем, пуста ли таблица Hotels

IF NOT EXISTS (SELECT \* FROM [dbo].[Hotels])

BEGIN

-- Заполняем таблицу 50 примерами данных

DECLARE @HotelIndex INT;

SET @HotelIndex = 1;

WHILE @HotelIndex <= 50

BEGIN

INSERT INTO [dbo].[Hotels] ([Name], [Country], [City], [Address], [Phone], [Stars], [TheContactPerson], [Photo])

VALUES (

'Отель ' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальное название отеля

'Страна ' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальная страна

'Город ' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальный город

'ул. Улица ' + CAST(@HotelIndex AS NVARCHAR(10)) + ', д. ' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальный адрес

'1234567890' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальный телефон

@HotelIndex % 5 + 1, -- Рейтинг отеля (звезды)

'Контактное лицо ' + CAST(@HotelIndex AS NVARCHAR(10)), -- Уникальное контактное лицо

0x -- Пустое изображение (для столбца Photo)

);

SET @HotelIndex = @HotelIndex + 1;

END

END

-- Проверяем, пуста ли таблица AdditionalServices

IF NOT EXISTS (SELECT \* FROM [dbo].[AdditionalServices])

BEGIN

-- Проверяем, пуста ли таблица AdditionalServices

IF (SELECT COUNT(\*) FROM [dbo].[AdditionalServices]) = 0

BEGIN

-- Заполняем таблицу 50 примерами данных

INSERT INTO [dbo].[AdditionalServices] ([Name], [Description], [Price])

VALUES

('Услуга 1', 'Описание услуги 1', 10.99),

('Услуга 2', 'Описание услуги 2', 25.50),

('Услуга 3', 'Описание услуги 3', 5.75),

('Услуга 4', 'Описание услуги 4', 30.00),

('Услуга 5', 'Описание услуги 5', 15.25),

('Услуга 6', 'Описание услуги 6', 12.99),

('Услуга 7', 'Описание услуги 7', 8.50),

('Услуга 8', 'Описание услуги 8', 40.00),

('Услуга 9', 'Описание услуги 9', 18.75),

('Услуга 10', 'Описание услуги 10', 22.50),

('Услуга 11', 'Описание услуги 11', 13.99),

('Услуга 12', 'Описание услуги 12', 7.25),

('Услуга 13', 'Описание услуги 13', 9.99),

('Услуга 14', 'Описание услуги 14', 11.50),

('Услуга 15', 'Описание услуги 15', 35.00),

('Услуга 16', 'Описание услуги 16', 28.75),

('Услуга 17', 'Описание услуги 17', 17.50),

('Услуга 18', 'Описание услуги 18', 21.99),

('Услуга 19', 'Описание услуги 19', 6.25),

('Услуга 20', 'Описание услуги 20', 14.50),

('Услуга 21', 'Описание услуги 21', 31.00),

('Услуга 22', 'Описание услуги 22', 19.75),

('Услуга 23', 'Описание услуги 23', 27.99),

('Услуга 24', 'Описание услуги 24', 23.50),

('Услуга 25', 'Описание услуги 25', 20.25),

('Услуга 26', 'Описание услуги 26', 45.00),

('Услуга 27', 'Описание услуги 27', 33.75),

('Услуга 28', 'Описание услуги 28', 12.50),

('Услуга 29', 'Описание услуги 29', 15.99),

('Услуга 30', 'Описание услуги 30', 8.25),

('Услуга 31', 'Описание услуги 31', 24.50),

('Услуга 32', 'Описание услуги 32', 29.00),

('Услуга 33', 'Описание услуги 33', 9.75),

('Услуга 34', 'Описание услуги 34', 10.99),

('Услуга 35', 'Описание услуги 35', 22.50),

('Услуга 36', 'Описание услуги 36', 13.25),

('Услуга 37', 'Описание услуги 37', 16.50),

('Услуга 38', 'Описание услуги 38', 32.00),

('Услуга 39', 'Описание услуги 39', 18.75),

('Услуга 40', 'Описание услуги 40', 25.99),

('Услуга 41', 'Описание услуги 41', 7.25),

('Услуга 42', 'Описание услуги 42', 11.50),

('Услуга 43', 'Описание услуги 43', 35.00),

('Услуга 44', 'Описание услуги 44', 28.75),

('Услуга 45', 'Описание услуги 45', 17.50),

('Услуга 46', 'Описание услуги 46', 21.99),

('Услуга 47', 'Описание услуги 47', 6.25),

('Услуга 48', 'Описание услуги 48', 14.50),

('Услуга 49', 'Описание услуги 49', 31.00),

('Услуга 50', 'Описание услуги 50', 19.75);

END

END

-- Проверяем, пуста ли таблица Clients

IF NOT EXISTS (SELECT \* FROM [dbo].[Clients])

BEGIN

-- Переменные для генерации реалистичных данных

DECLARE @FirstNames TABLE (Name NVARCHAR(50));

DECLARE @LastNames TABLE (Name NVARCHAR(50));

INSERT INTO @FirstNames (Name) VALUES

('John'), ('Jane'), ('Robert'), ('Emily'), ('Michael'), ('Sarah');

INSERT INTO @LastNames (Name) VALUES

('Smith'), ('Johnson'), ('Williams'), ('Brown'), ('Davis');

-- Проверяем, пуста ли таблица Clients

IF NOT EXISTS (SELECT \* FROM [dbo].[Clients])

BEGIN

-- Переменные для генерации реалистичных данных

DECLARE @CustomFirstNames TABLE (Name NVARCHAR(50));

DECLARE @CustomLastNames TABLE (Name NVARCHAR(50));

INSERT INTO @CustomFirstNames (Name) VALUES

('Иван'), ('Мария'), ('Александр'), ('Екатерина'), ('Дмитрий'), ('Анна');

INSERT INTO @CustomLastNames (Name) VALUES

('Иванов'), ('Петров'), ('Сидоров'), ('Козлов'), ('Смирнов');

-- Генерируем случайные данные для клиентов

WHILE (SELECT COUNT(\*) FROM [dbo].[Clients]) < 50

BEGIN

DECLARE @CustomFirstName NVARCHAR(50);

DECLARE @CustomLastName NVARCHAR(50);

DECLARE @DateOfBirth DATE;

DECLARE @Sex NVARCHAR(50);

DECLARE @Address NVARCHAR(100);

DECLARE @Series NVARCHAR(50);

DECLARE @Number BIGINT;

DECLARE @Discount BIGINT;

SELECT TOP 1 @CustomFirstName = Name FROM @CustomFirstNames ORDER BY NEWID();

SELECT TOP 1 @CustomLastName = Name FROM @CustomLastNames ORDER BY NEWID();

SET @DateOfBirth = DATEADD(DAY, -1 \* (ABS(CHECKSUM(NEWID())) % 36525), GETDATE()); -- Генерируем дату рождения в пределах 100 лет

SET @Sex = CASE WHEN (SELECT COUNT(\*) FROM [dbo].[Clients]) % 2 = 0 THEN 'Мужской' ELSE 'Женский' END;

SET @Address = 'Адрес ' + CAST((SELECT COUNT(\*) FROM [dbo].[Clients]) AS NVARCHAR(10));

SET @Series = LEFT(CAST(NEWID() AS NVARCHAR(36)), 2); -- Случайная серия (первые два символа GUID)

SET @Number = ABS(CAST(CHECKSUM(NEWID()) AS BIGINT)) % 10000000000; -- Случайный номер в пределах 10 миллиардов

SET @Discount = ABS(CAST(CHECKSUM(NEWID()) AS INT)) % 20; -- Случайная скидка в пределах 20

-- Вставляем данные в таблицу

INSERT INTO [dbo].[Clients] ([FIO], [DateOfBirth], [Sex], [Address], [Series], [Number], [Discount])

VALUES (CONCAT(@CustomFirstName, ' ', @CustomLastName), @DateOfBirth, @Sex, @Address, @Series, @Number, @Discount);

END

END

-- Проверяем, пуста ли таблица Employees

IF NOT EXISTS (SELECT \* FROM [dbo].[Employees])

BEGIN

-- Переменные для генерации реалистичных данных

DECLARE @CusFirstNames TABLE (Name NVARCHAR(50));

DECLARE @CusLastNames TABLE (Name NVARCHAR(50));

INSERT INTO @CusFirstNames (Name) VALUES

('John'), ('Jane'), ('Robert'), ('Emily'), ('Michael'), ('Sarah');

INSERT INTO @CusLastNames (Name) VALUES

('Smith'), ('Johnson'), ('Williams'), ('Brown'), ('Davis');

-- Генерируем случайные данные для сотрудника

WHILE (SELECT COUNT(\*) FROM [dbo].[Employees]) < 50

BEGIN

DECLARE @FirstName NVARCHAR(50);

DECLARE @LastName NVARCHAR(50);

DECLARE @JobTitle NVARCHAR(50);

DECLARE @Age INT;

SELECT TOP 1 @FirstName = Name FROM @CusFirstNames ORDER BY NEWID();

SELECT TOP 1 @LastName = Name FROM @CusLastNames ORDER BY NEWID();

SET @JobTitle = CASE WHEN (SELECT COUNT(\*) FROM [dbo].[Employees]) % 5 = 0 THEN 'Manager' ELSE 'Employee' END;

SET @Age = RAND() \* 20 + 25; -- Генерируем возраст от 25 до 45 лет

-- Вставляем данные в таблицу

INSERT INTO [dbo].[Employees] ([FIO], [JobTitle], [Age])

VALUES (CONCAT(@FirstName, ' ', @LastName), @JobTitle, @Age);

END

END

-- Проверяем, пуста ли таблица TypesOfRecreation

IF NOT EXISTS (SELECT \* FROM [dbo].[TypesOfRecreation])

BEGIN

-- Заполняем таблицу 50 примерами данных

INSERT INTO [dbo].[TypesOfRecreation] ([Name], [Description], [Restrictions])

VALUES

('Пляжный отдых', 'Отдых на берегу моря', 'Нет особых ограничений'),

('Горнолыжный отдых', 'Катание на горных лыжах', 'Требуется опыт вождения лыж'),

('Экскурсии', 'Посещение туристических мест и достопримечательностей', 'Возможны долгие поездки'),

('Активный отдых', 'Спортивные мероприятия и приключения', 'Физическая подготовка необходима'),

('Культурный отдых', 'Посещение музеев и исторических мест', 'Без специальных ограничений'),

('Гастрономический туризм', 'Путешествие для гурманов', 'Возможны ограничения по диете'),

('Спа-отдых', 'Посещение спа-центров и процедур', 'Могут потребоваться предварительные записи'),

('Экологический туризм', 'Путешествие по экологически чистым местам', 'Соблюдение правил охраны природы'),

('Круизы', 'Путешествие на круизном лайнере', 'Важно учитывать стихийные бедствия'),

('Альпинизм', 'Покорение вершин и скал', 'Высокая физическая подготовка'),

('Путешествие с детьми', 'Семейный отдых с детьми', 'Удобства и развлечения для детей'),

('Фестивали и мероприятия', 'Посещение фестивалей и культурных событий', 'Сезонность мероприятий'),

('Путешествие в исторические эпохи', 'Посещение исторических реконструкций', 'Соблюдение исторической аутентичности'),

('Паломничество', 'Религиозное путешествие к святым местам', 'Духовное вдохновение и покаяние'),

('Треккинг', 'Пеший туризм по тропам и маршрутам', 'Специальное снаряжение и физическая подготовка'),

('Велотуризм', 'Путешествие на велосипеде', 'Безопасность на дороге'),

('Дайвинг', 'Подводное плавание и исследование морских глубин', 'Сертификация и оборудование'),

('Сафари', 'Путешествие на африканских сафари', 'Безопасность при встрече с дикой природой'),

('Городские прогулки', 'Ознакомление с историей города', 'Пешеходные маршруты и гиды'),

('Путешествие в маленькие деревни', 'Знакомство с местной культурой', 'Простые условия и общение с местными жителями'),

('Водные виды отдыха', 'Активности на воде: каякинг, серфинг и др.', 'Уровень подготовки и безопасность'),

('Гаражный туризм', 'Путешествие на автомобиле или мотоцикле', 'Транспорт и маршруты'),

('Экстремальный спорт', 'Путешествие для адреналин-зависимых', 'Экстремальные условия и безопасность'),

('Путешествие в лес', 'Оздоровление и медитация в природе', 'Выживание и безопасность'),

('Морская охота', 'Подводная охота и добыча морских биоресурсов', 'Законы и лицензии'),

('Археологические раскопки', 'Исследование археологических памятников', 'Соблюдение законов и методологии'),

('Искусство и ремесла', 'Мастер-классы и творческие занятия', 'Творческий потенциал и интересы'),

('Астрономические наблюдения', 'Изучение звезд и планет', 'Телескоп и наблюдательные условия'),

('Походы с палатками', 'Переезды и ночевки в палатках', 'Подготовка и оборудование'),

('Термальные источники', 'Посещение горячих источников и курортов', 'Гигиенические меры и релаксация'),

('Фотосафари', 'Съемка дикой природы и животных', 'Фототехника и охрана природы'),

('Рыбалка', 'Охота на рыбу и ловля', 'Рыболовные удочки и законы'),

('Каякинг', 'Путешествие на байдарках и каяках', 'Безопасность на воде и маршруты'),

('Серфинг', 'Катание на волнах и серфбордах', 'Оборудование и безопасность'),

('Йога и здоровье', 'Занятия йогой и здоровый образ жизни', 'Инструкторы и практика'),

('Лошади и верховая езда', 'Путешествие на лошадях и тренировки', 'Лошади и седла'),

('Походы в горы', 'Покорение вершин и альпинизм', 'Высота и экстримальные условия'),

('Медицинский туризм', 'Путешествие для получения медицинских услуг', 'Медицинская документация'),

('Путешествия на воздушных шарах', 'Воздушные прогулки и экскурсии', 'Безопасность и погодные условия'),

('Семейный отдых', 'Отдых с семьей и детьми', 'Удобства для детей и развлечения'),

('Спортивные мероприятия', 'Посещение спортивных событий и соревнований', 'Билеты и график мероприятий'),

('Зоотуризм', 'Посещение зоопарков и заповедников', 'Экологические меры и безопасность'),

('Архитектурный туризм', 'Исследование архитектурных памятников и стилей', 'Архитектурные особенности'),

('Гастрономические туры', 'Путешествие для гурманов и дегустации', 'Местная кухня и рестораны'),

('Кино и культура', 'Посещение киносъемок и культурных мероприятий', 'Расписание и билеты'),

('Походы на водопады', 'Путешествие к водопадам и каскадам', 'Тропы и маршруты к водопадам'),

('Посещение термальных источников', 'Оздоровление в термальных источниках', 'Температурные режимы и лечебные свойства'),

('Путешествия на морских круизных лайнерах', 'Плавание на круизных кораблях', 'Маршруты и каюты на корабле'),

('Фотографический туризм', 'Съемка пейзажей и архитектуры', 'Фотоаппаратура и композиция'),

('Спелеотуризм', 'Исследование подземных пещер и карстов', 'Снаряжение и опасности в пещерах'),

('Индивидуальные туры', 'Организация индивидуальных маршрутов', 'Персональное обслуживание'),

('Исторические путешествия', 'Посещение исторических мест и памятников', 'Экскурсии и исторические факты'),

('Садоводство и ботанические сады', 'Уход за садами и изучение растений', 'Ботанические особенности');

END

END

-- Проверяем, пуста ли таблица Vouchers

IF NOT EXISTS (SELECT \* FROM [dbo].[Vouchers])

BEGIN

DECLARE @VoucherIndex INT;

SET @VoucherIndex = 1;

WHILE @VoucherIndex <= 50

BEGIN

DECLARE @StartDate DATE;

DECLARE @ExpirationDate DATE;

DECLARE @HotelId INT;

DECLARE @TypeOfRecreationId INT;

DECLARE @AdditionalServiceId INT;

DECLARE @ClientId INT;

DECLARE @EmployessId INT;

DECLARE @Reservation BIT;

DECLARE @Payment BIT;

-- Генерация случайных значений для записи в таблицу Vouchers

SET @StartDate = DATEADD(DAY, -1 \* (ABS(CHECKSUM(NEWID())) % 365), GETDATE());

SET @ExpirationDate = DATEADD(DAY, (ABS(CHECKSUM(NEWID())) % 30) + 1, @StartDate);

SET @HotelId = (ABS(CHECKSUM(NEWID())) % 50) + 1; -- Предполагается, что у вас есть 50 отелей

SET @TypeOfRecreationId = (ABS(CHECKSUM(NEWID())) % 50) + 1; -- Предполагается, что у вас есть 50 типов отдыха

SET @AdditionalServiceId = (ABS(CHECKSUM(NEWID())) % 10) + 1; -- Предполагается, что у вас есть 10 дополнительных услуг

SET @ClientId = (ABS(CHECKSUM(NEWID())) % 50) + 1; -- Предполагается, что у вас есть 50 клиентов

SET @EmployessId = (ABS(CHECKSUM(NEWID())) % 50) + 1; -- Предполагается, что у вас есть 50 сотрудников

SET @Reservation = CASE WHEN @VoucherIndex % 2 = 0 THEN 1 ELSE 0 END; -- Альтернативно, меняйте значение бита для резервации

SET @Payment = CASE WHEN @VoucherIndex % 3 = 0 THEN 1 ELSE 0 END; -- Альтернативно, меняйте значение бита для оплаты

-- Вставляем данные в таблицу Vouchers

INSERT INTO [dbo].[Vouchers] ([StartDate], [ExpirationDate], [HotelId], [TypeOfRecreationId], [AdditionalServiceId], [ClientId], [EmployessId], [Reservation], [Payment])

VALUES (@StartDate, @ExpirationDate, @HotelId, @TypeOfRecreationId, @AdditionalServiceId, @ClientId, @EmployessId, @Reservation, @Payment);

SET @VoucherIndex = @VoucherIndex + 1;

END

END

-- Просмотр данных таблиц

GO

CREATE VIEW VoucherView AS

SELECT

v.[Id] AS [VoucherId],

v.[StartDate],

v.[ExpirationDate],

h.[Name] AS [HotelName],

c.[FIO] AS [ClientName],

e.[FIO] AS [EmployeeName],

tr.[Name] AS [RecreationType],

s.[Name] AS [AdditionalService],

v.[Reservation],

v.[Payment]

FROM [dbo].[Vouchers] v

JOIN [dbo].[Hotels] h ON v.[HotelId] = h.[Id]

JOIN [dbo].[Clients] c ON v.[ClientId] = c.[Id]

JOIN [dbo].[Employees] e ON v.[EmployessId] = e.[Id]

JOIN [dbo].[TypesOfRecreation] tr ON v.[TypeOfRecreationId] = tr.[Id]

JOIN [dbo].[AdditionalServices] s ON v.[AdditionalServiceId] = s.[Id];

-- Создаем представление для таблицы Hotels

GO

CREATE VIEW HotelView AS

SELECT \* FROM [dbo].[Hotels];

-- Представление для таблицы TypesOfRecreation

GO

CREATE VIEW RecreationView AS

SELECT \* FROM [dbo].[TypesOfRecreation];

-- Создаем представление с определенными данными

GO

CREATE VIEW EmployeesOver30 AS

SELECT \* FROM [dbo].[Employees]

WHERE [Age] > 30;

-- Создание хранимых процедур

IF OBJECT\_ID('dbo.InsertHotel', 'P') IS NOT NULL

DROP PROCEDURE dbo.InsertHotel;

GO

CREATE PROCEDURE InsertHotel

@Name NVARCHAR(100),

@Country NVARCHAR(100),

@City NVARCHAR(100),

@Address NVARCHAR(200),

@Phone NVARCHAR(20),

@Stars INT,

@TheContactPerson NVARCHAR(100)

AS

BEGIN

INSERT INTO [dbo].[Hotels] ([Name], [Country], [City], [Address], [Phone], [Stars], [TheContactPerson])

VALUES (@Name, @Country, @City, @Address, @Phone, @Stars, @TheContactPerson);

END

GO

IF OBJECT\_ID('dbo.UpdateClient', 'P') IS NOT NULL

DROP PROCEDURE dbo.UpdateClient;

GO

CREATE PROCEDURE UpdateClient

@ClientId INT,

@FIO NVARCHAR(150),

@DateOfBirth DATE,

@Sex NVARCHAR(50),

@Address NVARCHAR(100),

@Series NVARCHAR(50),

@Number BIGINT,

@Discount BIGINT

AS

BEGIN

UPDATE [dbo].[Clients]

SET

[FIO] = @FIO,

[DateOfBirth] = @DateOfBirth,

[Sex] = @Sex,

[Address] = @Address,

[Series] = @Series,

[Number] = @Number,

[Discount] = @Discount

WHERE

[Id] = @ClientId;

END

GO

IF OBJECT\_ID('dbo.DeleteAdditionalServiceById', 'P') IS NOT NULL

DROP PROCEDURE dbo.DeleteAdditionalServiceById;

GO

CREATE PROCEDURE DeleteAdditionalServiceById

@ServiceId INT

AS

BEGIN

DELETE FROM [dbo].[AdditionalServices]

WHERE [Id] = @ServiceId;

END

GO