**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

Бондарев Е.Ю.

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.

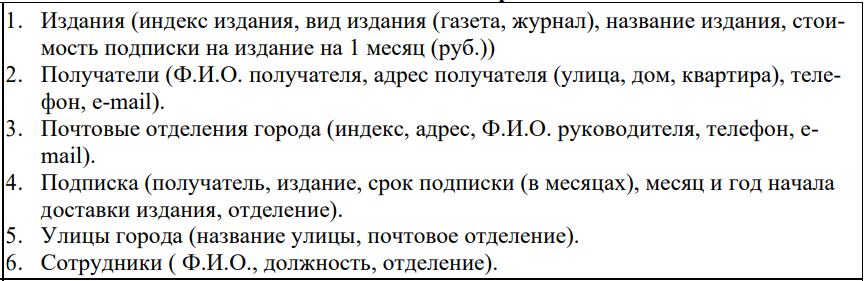
****

Рисунок 2 – Вариант задания

**Ход работы**

В начале процесса проектирования базы данных была создана таблица с названием *"PublicationType",* которая разработана с целью создания перечисления основных типов изданий (газеты и журналы). Это было сделано с учетом необходимости добавления новых типов изданий в будущем. Пример таблицы указан на рисунке 2. Листинг создания этой таблицы указана в приложении А.



Рисунок 2 – Пример таблицы *PublicationType*

Далее была создана таблица с названием *"Publication"*, в которой содержится информация о изданиях, такая как цена, название и тип. Эта таблица устанавливает связь таблицей *"PublicationType",* которая была описана выше. Чтобы заполнить таблицу *"Publication"* с данными, была создана виртуальная таблица, содержащую названия изданий. Эти названия были добавлены в таблицу *"Publication"* в случайном порядке. Пример таблицы указан на рисунке 3. Листинг создания этой таблицы указана в приложении А.

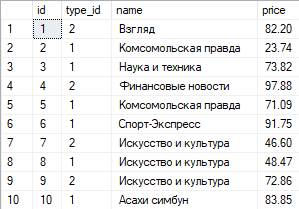


Рисунок 3 – Пример таблицы *Publication*

После была разработана таблица *"RecipientAddress"*, в которой хранилась информация о адресе получателя, включая улицу, дом и квартиру. Для заполнения этой таблицы были использованы случайные названиями улиц из виртуальной таблицы.

Далее была создана ещё одна таблица с названием *"Recipient"*, в которой была собрана полная информация о получателе, включая его ФИО, адрес, контактный телефон и адрес электронной почты. Эта таблица устанавливает связь между получателями и их адресами. Пример таблиц *RecipientAddress* и *Recipient* указан на рисунках 4 и 5. Листинг создания таблиц указана в приложении А.

****

Рисунок 4 – Пример таблицы *RecipientAddress*

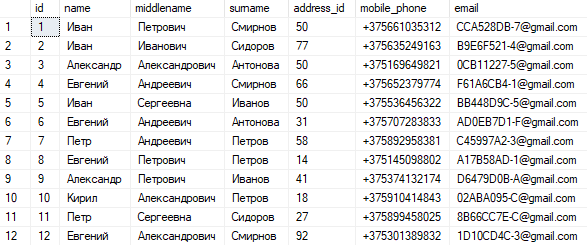
****

Рисунок 5 – Пример таблицы *Recipient*

В дальнейшей работе была создана таблица *"EmployeePosition"*. Эта таблица содержит информацию о должностях сотрудников в почтовом отделении и так же выступает перечислением как таблица *"PublicationType"*. В таблице *"EmployeePosition"* были перечислены различные должности сотрудников, такие как "почтальон", "кассир" и другие.

Это обеспечивает гибкость в управлении должностями сотрудников в почтовом отделении и позволяет легко добавлять новые должности в будущем. Пример таблицы *EmployeePosition* указан на рисунке 6. Листинг создания этой таблицы указана в приложении А.

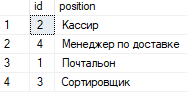


Рисунок 6 – Пример таблицы *EmployeePosition*

Далее, в ходе проектирования базы данных, была разработана таблица с названием *"Office"*, предназначенная для хранения информации о почтовых отделениях в разных городах. Пример таблицы *Office* указан на рисунке 7. Листинг создания этой таблицы указана в приложении А.

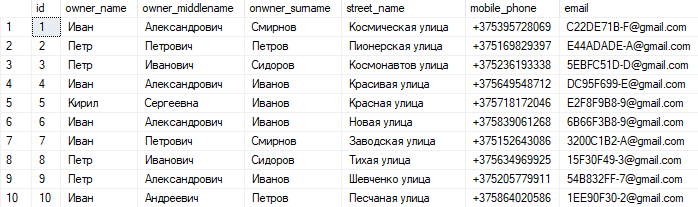


Рисунок 7 – Пример таблицы *Office*

Таблица *"Employee"* в базе данных содержит следующую информацию о сотрудниках почтового отделения: Ф.И.О, офис (ссылается на таблицу *"Office"*) и должность сотрудника (ссылается на таблицу *"EmployeePosition"*). Пример таблицы *Employee* указан на рисунке 8. Листинг создания этой таблицы указана в приложении А.

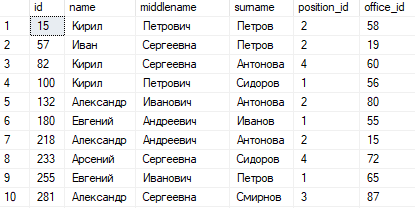


Рисунок 8 – Пример таблицы *Employee*

Таблица *"Subscription"* в базе данных содержит следующие поля: Получатель (ссылка на таблицу *"Recipient"*), издание (ссылается на таблицу *"Publication"*), срок подписки, месяц и год начала доставки, отделение (ссылается на таблицу *"Office"*). Пример таблицы *Subscription* указан на рисунке 8. Листинг создания этой таблицы указана в приложении А.

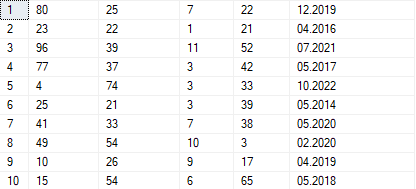


Рисунок 8 – Пример таблицы *Subscription*

После создания базы данных, была создана диаграмма, которая иллюстрирует зависимости между таблицами. Пример диаграммы базы данных указан на рисунке 9.

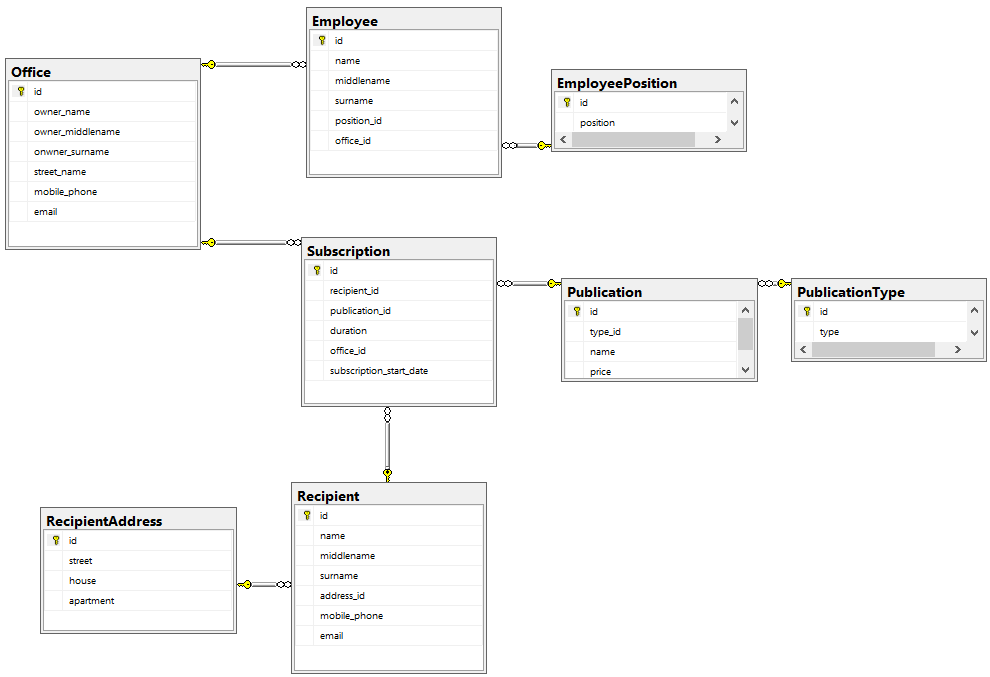


Рисунок 9 – Пример диаграммы базы данных

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 изданий. Пример работы данного представления указан на рисунке 10. Листинг кода по созданию этого представления указан в приложении А.

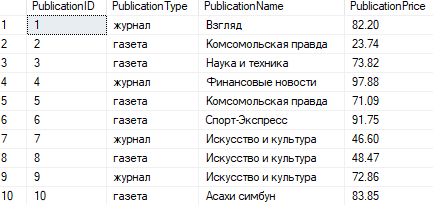


Рисунок 10 – Пример работы представления *PublicationView*

Вторым было создано представления для получения всех данных из таблицы получателей изданий. Пример работы данного представления указан на рисунке 11. Листинг кода по созданию этого представления указан в приложении А.



Рисунок 11 – Пример работы представления *RecipientView*

Третьим было создано представления для получения всех данных о подписках. Пример работы данного представления указан на рисунке 12. Листинг кода по созданию этого представления указан в приложении А.

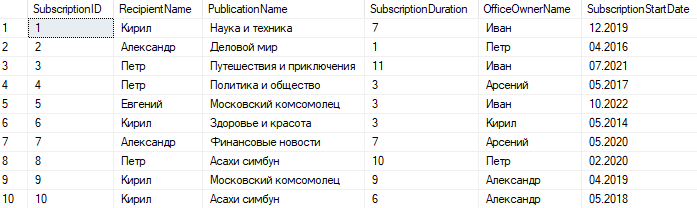


Рисунок 12 – Пример работы представления *SubscriptionView*

Далее было создано представления для получения всех данных из таблицы офисов. Пример работы данного представления указан на рисунке 13. Листинг кода по созданию этого представления указан в приложении А.

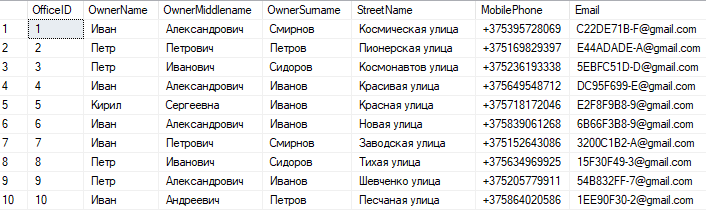


Рисунок 13 – Пример работы представления *OfficeView*

Далее были реализованы пять хранимых процедур для добавления новых записей в таблицы *Employee, Office, Publication, Recipient* и *Subscription*. Листинг данных хранимых процедур указан в приложении А.

**Вывод**: в процессе выполнения лабораторной работы была разработана и настроена база данных в среде MS SQL Server для использования в серверной части клиент-серверного приложения. Созданы необходимые таблицы баз данных и заполны данными. Также были разработаны хранимые процедуры, которые обеспечивают возможность вставки новых записей в таблицы. Кроме того, были созданы представления, упрощающие доступ к данным из базы и обеспечивающие более удобное представление информации.

**ПРИЛОЖНИЕ А**

Листинг скрипта для генерации базы данных

-- Создание базы с проверкой

USE master

IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = 'SubsCity1')

BEGIN

CREATE DATABASE SubsCity1

END

GO

ALTER DATABASE SubsCity1 SET RECOVERY SIMPLE

GO

USE [SubsCity1]

-- Создание таблиц с проверкой

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

BEGIN

CREATE TABLE [dbo].[PublicationType](

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

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

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

)

END

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

BEGIN

CREATE TABLE [dbo].[RecipientAddress](

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

[street] [nvarchar](50) NULL,

[house] [int] NULL,

[apartment] [int] NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

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

BEGIN

CREATE TABLE [dbo].[EmployeePosition](

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

[position] [nvarchar](50) NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[position] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

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

BEGIN

CREATE TABLE [dbo].[Publication](

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

[type\_id] [int] NOT NULL,

[name] [nvarchar](70) NOT NULL,

[price] [decimal](10, 2) NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Publication] WITH CHECK ADD FOREIGN KEY([type\_id])

REFERENCES [dbo].[PublicationType] ([id])

END

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

BEGIN

CREATE TABLE [dbo].[Recipient](

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

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

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

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

[address\_id] [int] NOT NULL,

[mobile\_phone] [nvarchar](20) NOT NULL,

[email] [nvarchar](255) NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[mobile\_phone] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Recipient] WITH CHECK ADD FOREIGN KEY([address\_id])

REFERENCES [dbo].[RecipientAddress] ([id])

END

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

BEGIN

CREATE TABLE [dbo].[Office](

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

[owner\_name] [nvarchar](20) NOT NULL,

[owner\_middlename] [nvarchar](20) NOT NULL,

[onwner\_surname] [nvarchar](20) NOT NULL,

[street\_name] [nvarchar](50) NOT NULL,

[mobile\_phone] [nvarchar](20) NOT NULL,

[email] [nvarchar](255) NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[mobile\_phone] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

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

BEGIN

CREATE TABLE [dbo].[Employee](

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

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

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

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

[position\_id] [int] NOT NULL,

[office\_id] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Employee] WITH CHECK ADD FOREIGN KEY([office\_id])

REFERENCES [dbo].[Office] ([id])

ALTER TABLE [dbo].[Employee] WITH CHECK ADD FOREIGN KEY([position\_id])

REFERENCES [dbo].[EmployeePosition] ([id])

END

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

BEGIN

CREATE TABLE [dbo].[Subscription](

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

[recipient\_id] [int] NOT NULL,

[publication\_id] [int] NOT NULL,

[duration] [int] NOT NULL,

[office\_id] [int] NOT NULL,

[subscription\_start\_date] [nvarchar](7) NOT NULL

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Subscription] WITH CHECK ADD FOREIGN KEY([office\_id])

REFERENCES [dbo].[Office] ([id])

ALTER TABLE [dbo].[Subscription] WITH CHECK ADD FOREIGN KEY([publication\_id])

REFERENCES [dbo].[Publication] ([id])

ALTER TABLE [dbo].[Subscription] WITH CHECK ADD FOREIGN KEY([recipient\_id])

REFERENCES [dbo].[Recipient] ([id])

END

-- Заполнение таблиц (если они пусты)

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

BEGIN

INSERT INTO [dbo].[PublicationType] ([type])

VALUES ('газета'), ('журнал')

END

DECLARE @StreetNames TABLE (Name NVARCHAR(50))

INSERT INTO @StreetNames (Name)

VALUES

('Пролетарская улица'),

('Ленинская улица'),

('Гагарина улица'),

('Советская улица'),

('Пушкинская улица'),

('Московская улица'),

('Кировская улица'),

('Парковая улица'),

('Садовая улица'),

('Комсомольская улица'),

('Школьная улица'),

('Жукова улица'),

('Мичурина улица'),

('Свердлова улица'),

('Октябрьская улица'),

('Горького улица'),

('Красноармейская улица'),

('Рабочая улица'),

('Зеленая улица'),

('Трудовая улица'),

('Полярная улица'),

('Красная улица'),

('Строителей улица'),

('Молодежная улица'),

('Центральная улица'),

('Новая улица'),

('Солнечная улица'),

('Заречная улица'),

('Пионерская улица'),

('Речная улица'),

('Восточная улица'),

('Западная улица'),

('Южная улица'),

('Северная улица'),

('Цветочная улица'),

('Лесная улица'),

('Юбилейная улица'),

('Гранитная улица'),

('Маяковского улица'),

('Первомайская улица'),

('Коммунальная улица'),

('Чкалова улица'),

('Горная улица'),

('Сиреневая улица'),

('Сосновая улица'),

('Дружбы улица'),

('Озерная улица'),

('Заводская улица'),

('Вокзальная улица'),

('Партизанская улица'),

('Островская улица'),

('Городская улица'),

('Карла Маркса улица'),

('Железнодорожная улица'),

('Набережная улица'),

('Мирная улица'),

('Севастопольская улица'),

('Колхозная улица'),

('Совхозная улица'),

('Театральная улица'),

('Лермонтова улица'),

('Пушкинская улица'),

('Мичуринская улица'),

('Свердловская улица'),

('Щорса улица'),

('Смирнова улица'),

('Гусарская улица'),

('Петровская улица'),

('Космонавтов улица'),

('Суворова улица'),

('Фрунзе улица'),

('Толстого улица'),

('Горького улица'),

('Шевченко улица'),

('Греческая улица'),

('Воронцовская улица'),

('Скверная улица'),

('Сельская улица'),

('Холодильная улица'),

('Степная улица'),

('Заводская улица'),

('Космическая улица'),

('Речная улица'),

('Парковая улица'),

('Береговая улица'),

('Школьная улица'),

('Соседская улица'),

('Луговая улица'),

('Озерная улица'),

('Красивая улица'),

('Полевая улица'),

('Дачная улица'),

('Живописная улица'),

('Просторная улица'),

('Зеленая улица'),

('Чистая улица'),

('Тихая улица'),

('Песчаная улица')

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

BEGIN

DECLARE @House INT

DECLARE @Apartment INT

DECLARE @RandomStreet NVARCHAR(20)

WHILE (SELECT COUNT(\*) FROM [dbo].[RecipientAddress]) < 100

BEGIN

SELECT TOP 1 @RandomStreet = Name FROM @StreetNames ORDER BY NEWID()

SET @House = CAST(RAND() \* 100 AS INT) + 1

SET @Apartment = CAST(RAND() \* 20 AS INT) + 1

INSERT INTO [dbo].[RecipientAddress] ([street], [house], [apartment])

VALUES (@RandomStreet, @House, @Apartment)

END

END

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

BEGIN

INSERT INTO [dbo].[EmployeePosition] ([position])

VALUES

('Почтальон'),

('Кассир'),

('Сортировщик'),

('Менеджер по доставке')

END

-- Проверка, пуста ли таблица Publication

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

BEGIN

-- Заполнение таблицы Publication случайными данными

DECLARE @PublicationNames TABLE (Name NVARCHAR(70))

INSERT INTO @PublicationNames (Name)

VALUES

('Утренние новости'),

('Вечерний вестник'),

('Здоровье и красота'),

('Наука и техника'),

('Деловой мир'),

('Спортивные новости'),

('Искусство и культура'),

('Политика и общество'),

('Финансовые новости'),

('Путешествия и приключения'),

('Асахи симбун'),

('Спорт-Экспресс'),

('Взгляд'),

('Московский комсомолец'),

('Комсомольская правда')

DECLARE @PublicationType INT

DECLARE @PublicationName NVARCHAR(50)

DECLARE @PublicationPrice DECIMAL(10, 2)

WHILE (SELECT COUNT(\*) FROM [dbo].[Publication]) < 100

BEGIN

SET @PublicationType = CASE WHEN RAND() > 0.5 THEN 1 ELSE 2 END

SELECT TOP 1 @PublicationName = Name FROM @PublicationNames ORDER BY NEWID()

SET @PublicationPrice = CAST(RAND() \* 100 AS DECIMAL(10, 2))

INSERT INTO [dbo].[Publication] ([type\_id], [name], [price])

VALUES (@PublicationType, @PublicationName, @PublicationPrice)

END

END

DECLARE @Names TABLE (Name NVARCHAR(20))

DECLARE @MiddleNames TABLE (MiddleName NVARCHAR(20))

DECLARE @Surnames TABLE (Surname NVARCHAR(20))

INSERT INTO @Names (Name) VALUES ('Евгений'), ('Иван'), ('Петр'), ('Кирил'), ('Александр'), ('Арсений')

INSERT INTO @MiddleNames (MiddleName) VALUES ('Иванович'), ('Андреевич'), ('Петрович'), ('Сергеевна'), ('Александрович')

INSERT INTO @Surnames (Surname) VALUES ('Иванов'), ('Антонова'), ('Петров'), ('Смирнов'), ('Сидоров')

DECLARE @AddressID INT

DECLARE @MobilePhone NVARCHAR(20)

DECLARE @Email NVARCHAR(255)

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

BEGIN

-- Заполнение таблицы Recipient

WHILE (SELECT COUNT(\*) FROM [dbo].[Recipient]) < 100

BEGIN

SET @AddressID = CAST(RAND() \* 100 AS INT)

SET @MobilePhone = '+375' + CAST(100000000 + CAST(RAND() \* 899999999 AS INT) AS NVARCHAR(20))

SET @Email = LEFT(NEWID(), 10) + '@gmail.com'

INSERT INTO [dbo].[Recipient] ([name], [middlename], [surname], [address\_id], [mobile\_phone], [email])

SELECT TOP 1

(SELECT TOP 1 Name FROM @Names ORDER BY NEWID()),

(SELECT TOP 1 MiddleName FROM @MiddleNames ORDER BY NEWID()),

(SELECT TOP 1 Surname FROM @Surnames ORDER BY NEWID()),

@AddressID,

@MobilePhone,

@Email

END

END

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

BEGIN

WHILE (SELECT COUNT(\*) FROM [dbo].[Office]) < 100

BEGIN

SET @MobilePhone = '+375' + CAST(100000000 + CAST(RAND() \* 899999999 AS INT) AS NVARCHAR(20))

SET @Email = LEFT(NEWID(), 10) + '@gmail.com'

INSERT INTO [dbo].[Office] ([owner\_name], [owner\_middlename], [onwner\_surname], [street\_name], [mobile\_phone], [email])

SELECT TOP 1

(SELECT TOP 1 Name FROM @Names ORDER BY NEWID()),

(SELECT TOP 1 MiddleName FROM @MiddleNames ORDER BY NEWID()),

(SELECT TOP 1 Surname FROM @Surnames ORDER BY NEWID()),

(SELECT TOP 1 Name FROM @StreetNames ORDER BY NEWID()),

@MobilePhone,

@Email

END

END

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

BEGIN

DECLARE @PositionID INT

DECLARE @OfficeID INT

WHILE (SELECT COUNT(\*) FROM [dbo].[Employee]) < 100

BEGIN

SET @PositionID = CAST(RAND() \* 100 + 1 AS INT)

SET @OfficeID = CAST(RAND() \* 100 + 1 AS INT)

INSERT INTO [dbo].[Employee] ([name], [middlename], [surname], [position\_id], [office\_id])

SELECT TOP 1

(SELECT TOP 1 Name FROM @Names ORDER BY NEWID()),

(SELECT TOP 1 MiddleName FROM @MiddleNames ORDER BY NEWID()),

(SELECT TOP 1 Surname FROM @Surnames ORDER BY NEWID()),

@PositionID,

@OfficeID

END

END

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

BEGIN

DECLARE @RecipientID INT

DECLARE @PublicationID INT

DECLARE @Duration INT

DECLARE @StartDate NVARCHAR(7)

WHILE (SELECT COUNT(\*) FROM [dbo].[Subscription]) < 100

BEGIN

SET @RecipientID = CAST(RAND() \* 100 + 1 AS INT)

SET @PublicationID = CAST(RAND() \* 100 + 1 AS INT)

SET @Duration = CAST(RAND() \* 12 + 1 AS INT)

SET @OfficeID = CAST(RAND() \* 100 + 1 AS INT)

SET @StartDate = RIGHT('0' + CAST(ROUND(RAND() \* 11 + 1, 0) AS NVARCHAR(2)), 2) + '.' + CAST(ROUND(RAND() \* 9 + 2014, 0) AS NVARCHAR(4))

INSERT INTO [dbo].[Subscription] ([recipient\_id], [publication\_id], [duration], [office\_id], [subscription\_start\_date])

VALUES (@RecipientID, @PublicationID, @Duration, @OfficeID, @StartDate)

END

END

-- Создание хранимых процедур

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

DROP PROCEDURE dbo.InsertPublication;

GO

CREATE PROCEDURE InsertPublication

@type\_id INT,

@name NVARCHAR(70),

@price DECIMAL(10, 2)

AS

BEGIN

INSERT INTO [dbo].[Publication] ([type\_id], [name], [price])

VALUES (@type\_id, @name, @price)

END

GO

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

DROP PROCEDURE dbo.InsertRecipient;

GO

CREATE PROCEDURE InsertRecipient

@name NVARCHAR(20),

@middlename NVARCHAR(20),

@surname NVARCHAR(20),

@street NVARCHAR(50),

@house INT,

@apartment INT,

@mobile\_phone NVARCHAR(20),

@email NVARCHAR(255)

AS

BEGIN

DECLARE @AddressID INT

INSERT INTO [dbo].[RecipientAddress] ([street], [house], [apartment])

VALUES (@street, @house, @apartment)

SET @AddressID = SCOPE\_IDENTITY()

INSERT INTO [dbo].[Recipient] ([name], [middlename], [surname], [address\_id], [mobile\_phone], [email])

VALUES (@name, @middlename, @surname, @AddressID, @mobile\_phone, @email)

END

GO

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

DROP PROCEDURE dbo.InsertOffice;

GO

CREATE PROCEDURE InsertOffice

@owner\_name NVARCHAR(20),

@owner\_middlename NVARCHAR(20),

@owner\_surname NVARCHAR(20),

@street\_name NVARCHAR(50),

@mobile\_phone NVARCHAR(20),

@email NVARCHAR(255)

AS

BEGIN

INSERT INTO [dbo].[Office] ([owner\_name], [owner\_middlename], [onwner\_surname], [street\_name], [mobile\_phone], [email])

VALUES (@owner\_name, @owner\_middlename, @owner\_surname, @street\_name, @mobile\_phone, @email)

END

GO

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

DROP PROCEDURE dbo.InsertSubscription;

GO

CREATE PROCEDURE InsertSubscription

@recipient\_id INT,

@publication\_id INT,

@duration INT,

@office\_id INT,

@subscription\_start\_date NVARCHAR(7)

AS

BEGIN

INSERT INTO [dbo].[Subscription] ([recipient\_id], [publication\_id], [duration], [office\_id], [subscription\_start\_date])

VALUES (@recipient\_id, @publication\_id, @duration, @office\_id, @subscription\_start\_date)

END

GO

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

DROP PROCEDURE dbo.InsertEmployee;

GO

CREATE PROCEDURE InsertEmployee

@name NVARCHAR(20),

@middlename NVARCHAR(20),

@surname NVARCHAR(20),

@position\_id INT,

@office\_id INT

AS

BEGIN

INSERT INTO [dbo].[Employee] ([name], [middlename], [surname], [position\_id], [office\_id])

VALUES (@name, @middlename, @surname, @position\_id, @office\_id)

END

GO

-- Создание представлений

IF NOT EXISTS (SELECT \* FROM sys.views WHERE [name] = 'PublicationView')

BEGIN

EXEC('

CREATE VIEW PublicationView AS

SELECT

p.id AS PublicationID,

pt.type AS PublicationType,

p.name AS PublicationName,

p.price AS PublicationPrice

FROM

[dbo].[Publication] p

JOIN

[dbo].[PublicationType] pt ON p.type\_id = pt.id;

');

END

IF NOT EXISTS (SELECT \* FROM sys.views WHERE [name] = 'RecipientView')

BEGIN

EXEC('

CREATE VIEW RecipientView AS

SELECT

r.id AS RecipientID,

r.name AS RecipientName,

r.middlename AS RecipientMiddlename,

r.surname AS RecipientSurname,

ra.street AS RecipientStreet,

ra.house AS RecipientHouse,

ra.apartment AS RecipientApartment,

r.mobile\_phone AS RecipientMobilePhone,

r.email AS RecipientEmail

FROM

[dbo].[Recipient] r

JOIN

[dbo].[RecipientAddress] ra ON r.address\_id = ra.id;

');

END

IF NOT EXISTS (SELECT \* FROM sys.views WHERE [name] = 'SubscriptionView')

BEGIN

EXEC('

CREATE VIEW SubscriptionView AS

SELECT

s.id AS SubscriptionID,

r.name AS RecipientName,

p.name AS PublicationName,

s.duration AS SubscriptionDuration,

o.owner\_name AS OfficeOwnerName,

s.subscription\_start\_date AS SubscriptionStartDate

FROM

[dbo].[Subscription] s

JOIN

[dbo].[Recipient] r ON s.recipient\_id = r.id

JOIN

[dbo].[Publication] p ON s.publication\_id = p.id

JOIN

[dbo].[Office] o ON s.office\_id = o.id;

');

END

IF NOT EXISTS (SELECT \* FROM sys.views WHERE [name] = 'OfficeView')

BEGIN

EXEC('

CREATE VIEW OfficeView AS

SELECT

id AS OfficeID,

owner\_name AS OwnerName,

owner\_middlename AS OwnerMiddlename,

onwner\_surname AS OwnerSurname,

street\_name AS StreetName,

mobile\_phone AS MobilePhone,

email AS Email

FROM

[dbo].[Office];

');

END