МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Рыбинский государственный авиационный технический университет

имени П.А. Соловьева»

Авиационный колледж

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

МДК 11.01 Технология разработки и защита баз данных

на тему

**БАЗА ДАННЫХ «ПОЛИКЛИНИКА»**

Студент группы ИС-3 Р.Н. Шибалов

Руководитель И.Н. Морошкин

Работа допущена к защите «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рыбинск, 2024

График выполнения работы

«База данных "ПОЛИКЛИНИКА"»

|  |  |  |
| --- | --- | --- |
| №п/п | Наименование разделов (глав) курсового проекта | Дата окончания |
| 1 | Составление списка литературных источников |  |
| 2 | Описание предметной области |  |
| 3 | Построение логической модели |  |
| 4 | Создание БД |  |
| 5 | Заполнение БД |  |
| 6 | Создание триггеров к БД |  |
| 7 | Создание представлений |  |
| 8 | Создание хранимых процедур |  |
| 9 | Создание презентации |  |
| 10 | Оформление пояснительной записки |  |
| 11 | Защита курсового проекта |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Руководитель проекта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

Задание к исполнению принял\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рыбинский государственный авиационный технический университет

имени П.А. Соловьева»

Авиационный колледж

ЗАДАНИЕ

на курсовой проект

по дисциплине

МДК.11.01 Разработка администрирование и защита баз данных

по специальности

09.02.07. Информационные системы и программирование

студенту группы ИС-3 Шибалову Роману Николаевичу

1 Тема курсового проекта – База данных «ПОЛИКЛИНИКА»

Содержание:

1. Разработать БД ПОЛИКЛИНИКА. В базу данных медицинской поликлиники занесены имя, пол, дата рождения, паспортные данные, домашний адрес каждого пациента, а также сведения о врачах.

Всякий раз, когда врач осматривает больного, записываются данные пациента, имя врача, дата проведения осмотра, симптомы заболевания, диагноз и предписания больному. Если врач прописывает больному лекарство, записывается его название, способ приема и дозировка, описание предполагаемого действия лекарства и возможных побочных эффектов. Система должна обеспечивать поиск и выдачу сведений по разным вопросам. Разработать БД «ПОЛИКЛИНИКА».

1. Срок сдачи законченного курсового проекта «\_\_» \_\_\_\_\_\_ 20\_\_ г.
2. Исходные данные: отсутствуют.
3. Перечень подлежащих разработке задач/вопросов: изучение теоретического материала: атрибуты базы данных, даталогическая модель

Содержание

[Введение 2](#_Toc185351199)

[1 Проектная часть 2](#_Toc185351200)

[1.1 Описание предметной области 2](#_Toc185351201)

[1.2 Анализ атрибутов 2](#_Toc185351202)

[1.3 Создание информационно-логической модели предметной области в каноническом виде 2](#_Toc185351203)

[1.4 Создание даталогической модели реляционной базы данных 2](#_Toc185351204)

[1.5 Разработка данных контрольного примера 2](#_Toc185351205)

[2 Программные разработки 2](#_Toc185351206)

[2.1 Создание базы данных 2](#_Toc185351207)

[2.2 Заполнение данными 2](#_Toc185351208)

[2.3 Разработка триггеров 2](#_Toc185351209)

[2.4 Разработка представлений 2](#_Toc185351210)

[2.5 Разработка хранимых процедур 2](#_Toc185351211)

[Заключение 2](#_Toc185351212)

[Список используемых источников 2](#_Toc185351213)

[Приложение А 2](#_Toc185351214)

[Приложение Б 2](#_Toc185351215)

[Приложение В 2](#_Toc185351216)

[Приложение Г 2](#_Toc185351217)

[Приложение Д 2](#_Toc185351218)

# Введение

Современные медицинские учреждения сталкиваются с растущими требованиями к организации и ведению учета пациентов, что делает автоматизацию процесса управления особенно актуальной. Поликлиники играют ключевую роль в системе здравоохранения, обеспечивая первичную медицинскую помощь населению. В связи с увеличением объема информации, связанной с пациентами и врачами, разработка эффективной базы данных (БД) становится необходимостью.

Медицинская база данных поликлиники будет включать в себя обширную информацию о каждом пациенте: имя, пол, дата рождения, паспортные данные и домашний адрес. Также важна информация о врачах, что позволит составлять полные профили для каждого обращения. При каждом осмотре пациента фиксируются имя и данные больного, имя врача, дата и место проведения осмотра, а также симптомы, диагноз и предписания. В случае назначения лекарств будет записано их название, способ приема и дозировка, а также информация о предполагаемом действии и возможных побочных эффектах.

Эта система должна обеспечить удобный поиск и выдачу сведений по различным запросам, что значительно оптимизирует работу медицинского персонала. Разработка БД «ПОЛИКЛИНИКА» станет важным шагом к улучшению управления информацией, повышению эффективности поликлиник.

Целью курсового проекта является разработка базы данных «ПОЛИКЛИНИКА», в которой будет храниться информация о пациентах, врачах, медицинских осмотрах и назначениях.

Для достижения данной цели были поставлены следующие задачи:

- изучить язык Transact-SQL и Microsoft SQL Server;

- изучить предметную область;

- проанализировать атрибуты предметной области;

- создать инфологическую и даталогическую модели данных;

- создать базу данных «ПОЛИКЛИНИКА».

# 1 Проектная часть

## 1.1 Описание предметной области

Концепция информационной системы для управления медицинской поликлиникой представляет собой важный инструмент, который позволяет эффективно организовать учет и управление данными о пациентах и врачах. Эта система предназначена для хранения информации о каждом пациенте, включая его имя, пол, дату рождения, паспортные данные и домашний адрес, а также профиль каждого врача.

Каждый раз, когда врач принимает пациента или выезжает на дом, фиксируются ключевые данные: имя и информация о пациенте, имя врача, дата и место осмотра, симптомы, диагноз и предписания. Если врач назначает лекарственные средства, система обеспечивает запись информации, такой как название препарата, способ приема, дозировка, предполагаемое действие и возможные побочные эффекты. Основная задача данной информационной системы — обеспечить возможность простого и быстрого доступа к медицинской информации, позволяя врачам и медперсоналу эффективно управлять данными.

Система также предоставляет функции поиска и выдачи сведений по различным запросам, что облегчает работу медицинского персонала и улучшает качество обслуживания пациентов. Важно, чтобы база данных была актуальной и могла легко обновляться, включая добавление новых пациентов, врачей, а также корректировку и удаление старых записей. Это позволит поликлинике оптимизировать процессы приема, лечения и назначения медикаментов, обеспечивая максимально эффективное взаимодействие с пациентами и поддержание высокого уровня медицинской помощи.

## 1.2 Анализ атрибутов

Проектирование базы данных начинается с создания всепокрывающего отношения, включающего все ключевые атрибуты и содержащего необходимые данные для выполнения задачи. Атрибут базы данных — это название колонки и информация, хранящаяся в соответствующих полях таблицы базы данных. Таблица представляет собой экземпляр правильного отношения, которое называется универсальным отношением проектируемой базы данных. В универсальное отношение включаются все ключевые атрибуты, и оно способно содержать все данные, которые планируется хранить в базе данных.

Нормализация данных, в свою очередь, представляет собой процесс организации и структурирования базы данных с целью устранения избыточной информации. Проще говоря, это способ обеспечить логическую организацию каждого поля и записи, чтобы не только избежать избыточности, но и сделать использование любой реляционной базы данных более эффективным: предотвращение ошибок при вводе данных, случайного удаления или облегчение процесса обновления информации. Нормализация данных подчиняется определенным правилам, которые определяют, как должна быть организована база данных.

Первая нормальная форма диктует, что каждое поле базы данных должно хранить одно значение и что в одной базе данных не должно быть двух полей, одинаково хранящих информацию.

Вторая нормальная форма направлена на уменьшение избыточности, гарантируя, что каждое поле хранит информацию, которая говорит нам что-то о первичном ключе. Другими словами:

- Каждая база данных должна иметь только один первичный ключ;

- Все не первичные ключи должны полностью зависеть от первичного ключа.

Эти два принципа обеспечивают целостность данных в базе, гарантируя, что информация об одном элементе, хранящаяся в первичном ключе, представлена последовательно. База данных считается соответствующей третьей нормальной форме, если в ней отсутствуют переходные зависимости. Это означает, что данные организованы таким образом, чтобы значения одного столбца не зависели от другого столбца, который в свою очередь зависит от первичного ключа. Чтобы достичь соответствия базы данных третьей нормальной форме, необходимо убрать любые столбцы, которые зависят от данных, не связанных напрямую с первичным ключом, и хранить такие сведения в другой базе данных с собственным ключом. Помимо описанных нормальных форм, существуют четвертая и пятая нормальные формы, но их применение в реальной практике не так распространено.

Проектирование базы данных «ПОЛИКЛИНИКА» начинается с определения основных объектов и их атрибутов, которые должны храниться в системе. В базе данных учитываются пациенты, врачи, консультации, назначения лекарств, симптомы, диагнозы и сведения, обеспечивающие управление данными о медицинских осмотрах. Каждая таблица в БД представляет собой отдельную сущность, хранящую структурированную информацию с соблюдением принципов нормализации.

Сущность «Passports» предназначена для хранения паспортных данных. Атрибуты данной таблицы представлены в таблице 1.1

Таблица 1.1 – Атрибуты сущности «Passports»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор паспорта |
| Series | NVARCHAR(10) | Серия паспорта |
| Number | NVARCHAR(20) | Номер паспорта |
| Surname | NVARCHAR(30) | Фамилия |
| Name | NVARCHAR(30) | Имя |
| FatherName | NVARCHAR(30) | Отчество |
| Sex | NCHAR(1) | Пол |
| DateOfBirth | DATE | Дата рождения |

Сущность «Patients» предназначена для хранения данных о пациентах. Атрибуты данной таблицы представлены в таблице 1.2

Таблица 1.2 – Атрибуты сущности «Patients»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор пациента |
| PassportId | INT | Идентификатор паспорта |
| CreatedAt | DATETIME2(7) | Дата создания записи |
| UpdatedAt | DATETIME2(7) | Дата обновления записи |

Сущность «Doctors» предназначена для хранения данных о докторах. Атрибуты данной таблицы представлены в таблице 1.3

Таблица 1.3 – Атрибуты сущности «Doctors»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор врача |
| Specialty | NVARCHAR(50) | Специализация врача |
| Phone | NVARCHAR(20) | Телефон врача |
| Email | NVARCHAR(100) | Электронная почта врача |
| ExperienceYears | INT | Количество лет опыта |
| Address | NVARCHAR(200) | Адрес врача |
| CreatedAt | DATETIME2(7) | Дата создания записи |
| UpdatedAt | DATETIME2(7) | Дата обновления записи |
| PassportId | INT | Идентификатор паспорта врача |

Таблица «WorkingHours» предназначена для хранения данных о рабочем графике докторов. Атрибуты данной таблицы представлены в таблице 1.4

Таблица 1.4 – Атрибуты сущности «WorkingHours»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор рабочего времени |
| StartTime | TIME(7) | Время начала работы |
| EndTime | TIME(7) | Время окончания работы |
| CreatedAt | DATETIME2(7) | Дата создания записи |
| UpdatedAt | DATETIME2(7) | Дата обновления записи |
| DoctorId | INT | Идентификатор врача |
| WorkDay | NVARCHAR(11) | Рабочий день |

Таблица «DozeUnits» предназначена для хранения мер дозировки медицинских препаратов. Атрибуты данной таблицы представлены в таблице 1.5

Таблица 1.5 – Атрибуты сущности «DozeUnits»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор единицы дозировки |
| Name | NVARCHAR(20) | Наименование единицы дозировки |

Таблица «MedicineUsingWays» предназначена для хранения путей приема медицинских препаратов. Атрибуты сущности «MedicineUsingWays» представлены в таблице 1.6

Таблица 1.6 – Атрибуты сущности «MedicineUsingWays»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор способа применения |
| Name | NVARCHAR(20) | Наименование способа применения |

Таблица «MedicineSideEffects» предназначена для хранения данных о эффектах медицинских препаратов. Ее атрибуты представлены в таблице 1.7

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор побочного эффекта |
| Name | NVARCHAR(50) | Наименование побочного эффекта |
| Description | NVARCHAR(200) | Описание побочного эффекта |
| MedicineId | INT | Идентификатор лекарства |

Таблица «Medicines» предназначена для хранения данных о медицинских препаратах. Атрибуты данной таблицы представлены в таблице 1.8

Таблица 1.8 – Атрибуты сущности «Medicines»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор лекарства |
| Name | NVARCHAR(100) | Наименование лекарства |
| DozeUnitId | INT | Идентификатор единицы дозировки |
| MedicineUsingWayId | INT | Идентификатор способа применения |

Таблица «Prescriptions» предназначена для хранения данных о выписанных рецептах. Атрибуты данной таблицы представлены в таблице 1.9

Таблица 1.9 – Атрибуты сущности «Prescriptions»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор назначения |
| Description | NVARCHAR(200) | Описание назначения |
| TimesInDay | INT | Частота применения в день |
| DozeCount | FLOAT | Количество дозировки |
| CreatedAt | DATETIME2(7) | Дата создания назначения |
| MedicineId | INT | Идентификатор лекарства |
| ConsultationId | INT | Идентификатор консультации |

Таблица «Consultations» предназначена для хранения информации о врачебных приемах. Атрибуты данной таблицы представлены в таблице 1.10

Таблица 1.10 – Атрибуты сущности «Consultations»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор консультации |
| Location | NVARCHAR(100) | Место проведения консультации |
| StartTime | DATETIME2(7) | Время начала консультации |
| EndTime | DATETIME2(7) | Время окончания консультации |
| DoctorId | INT | Идентификатор врача |
| PatientId | INT | Идентификатор пациента |

Таблица «Symptoms» предназначена для хранения симптомов. Атрибуты данной таблицы представлены в таблице 1.11

Таблица 1.11 – Атрибуты сущности «Symptoms»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор симптома |
| Name | NVARCHAR(100) | Наименование симптома |

Таблица «ConsultationSymptoms» предназначена для хранения выявленных симптомов на приеме. Атрибуты данной таблицы представлены в таблице 1.12

Таблица 1.12 – Атрибуты сущности «ConsultationSymptoms»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор записи симптома |
| SymptomId | INT | Идентификатор симптома |
| ConsultationId | INT | Идентификатор консультации |

Таблица «Roles» предназначена для хранения ролей пользователей. Атрибуты данной таблицы представлены в таблице 1.13

Таблица 1.13 – Атрибуты сущности «Roles»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор роли |
| Name | NVARCHAR(20) | Наименование роли |
| CreatedAt | DATETIME2(7) | Дата создания роли |

Таблица «Users» предназначена для хранения информации о пользователях. Атрибуты данной таблицы представлены в таблице 1.14

Таблица 1.14 – Атрибуты сущности «Users»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Id | INT | Идентификатор пользователя |
| Name | NVARCHAR(20) | Имя пользователя |
| Password | NCHAR(60) | Пароль |
| Email | NVARCHAR(100) | Электронная почта |
| Phone | NVARCHAR(20) | Телефон пользователя |
| CreatedAt | DATETIME2(7) | Дата создания записи |
| UpdatedAt | DATETIME2(7) | Дата обновления записи |
| RoleId | INT | Идентификатор роли |

## 1.3 Создание информационно-логической модели предметной области в каноническом виде

Инфологическая модель данных — это своего рода макет будущей базы данных, который представляет информацию о данных, используя различные инструменты, такие как текстовые описания, диаграммы, таблицы и графики. Она служит для описания информации и ее организации в базе данных, обеспечивая понимание структуры данных как для разработчиков, так и для конечных пользователей.

Концептуальная модель данных отражает сущности и связи между ними в базе данных. Сущности представляют объекты или события, которые моделируются в системе и имеют определенные характеристики, выраженные через атрибуты. Эти атрибуты могут быть использованы для однозначной идентификации каждого экземпляра сущности. Концептуальная модель фокусируется на ключевых аспектах данных и их взаимосвязях, предоставляя абстрактное представление информации.

Методология IDEF1X используется для моделирования реляционных баз данных. Она предлагает специальный синтаксис для создания концептуальных схем баз данных. В IDEF1X сущности описываются как наборы объектов с общими атрибутами. Связи между сущностями отражают отношения между ними и могут быть разных типов, например, один-ко-многим или многие-ко-многим.

Преимущества IDEF1X включают в себя строгие стандарты моделирования, что помогает уменьшить неоднозначность в интерпретации структуры базы данных. Этот подход способствует единообразному пониманию данных и облегчает коммуникацию между разработчиками и пользователями.

Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи – глаголы, которые показывают, как относятся сущности между собой. Связи могут быть нескольких видов: один ко одному, один ко многим, многие ко многим. Чаще используется связь вида один ко многим. Такие связи отображаются в виде линии между двумя сущностями с точкой на одном конце и глагольной фразой, отображаемой над линией. Отношение (связи) многие ко многим обычно используются на начальной стадии разработки диаграммы и отображаются в IDEF1X в виде сплошной линии с точками на обоих концах.

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построения модели.

На основе полученных сущностей и их атрибутов построим инфологическую модель базы данных «ПОЛИКЛИНИКА». На рисунке 1.1 изображена инфологическая модель базы данных «ПОЛИКЛИНИКА» методом IDEF1X.

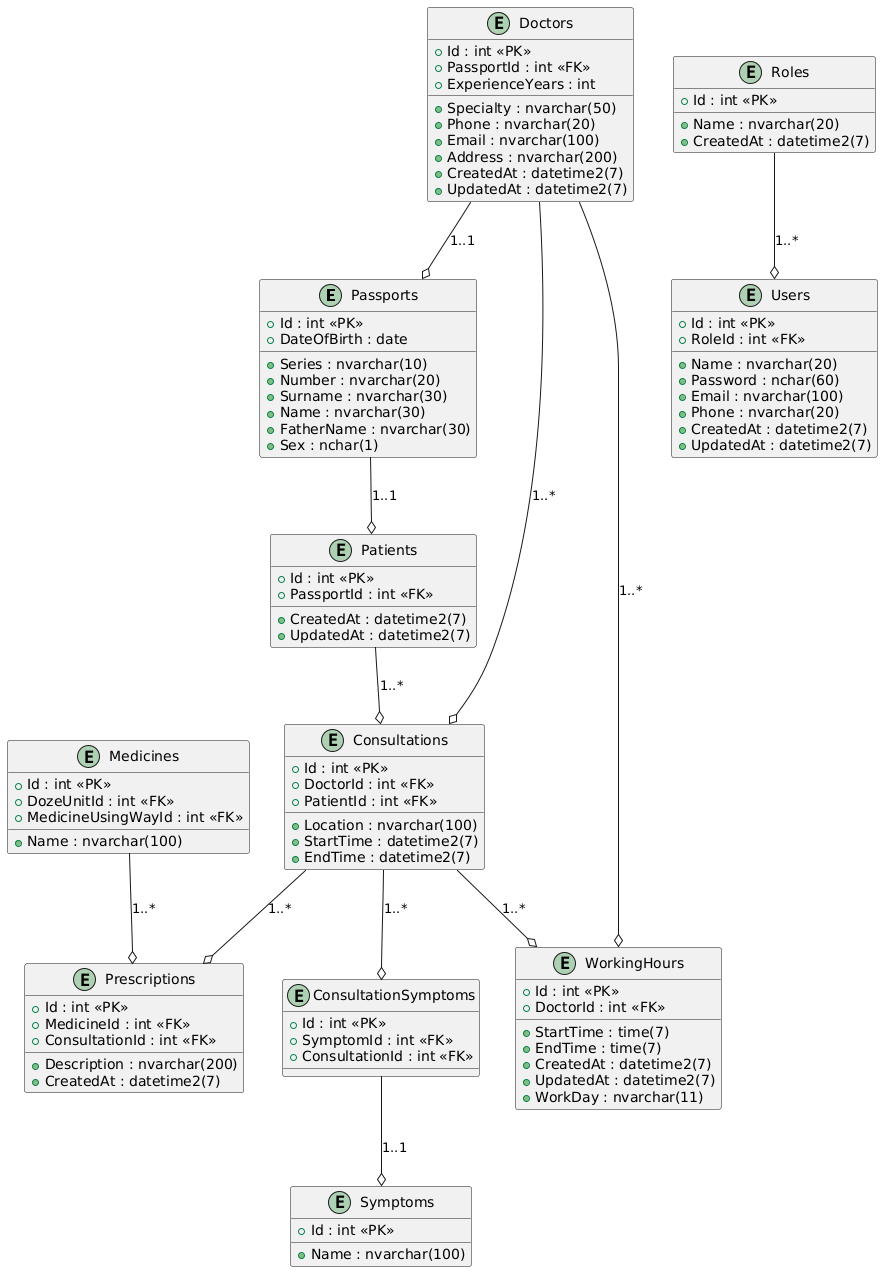


Рисунок 1.1 – Инфологическая модель базы данных «ПОЛИКЛИНИКА»

## 1.4 Создание даталогической модели реляционной базы данных

Даталогическая модель данных — это представление логической структуры базы данных в рамках конкретной системы управления базами данных (СУБД). Она отображает связи между элементами базы данных, учитывая особенности и возможности конкретной СУБД.

Для создания даталогической модели часто используют инфологическую модель, которая представляет собой описание структуры данных на уровне предметной области.

Проектирование даталогической модели для реляционной базы данных включает разделение всей информации на отдельные файлы или таблицы. В каждом файле определяется набор полей, которые отображаются как атрибуты в реляционной модели. Файлы соответствуют таблицам, а поля – атрибутам этих таблиц в базе данных.

На основе инфологической модели базы данных построена даталогическая модель базы данных «ПОЛИКЛИНИКА». Даталогическая модель базы данных «ПОЛИКЛИНИКА» представлена в приложении А.

## 1.5 Разработка данных контрольного примера

Под текстовым (контрольным) примером понимается совокупность исходных (текстовых) данных, для которых осуществляется полная апробация алгоритма. К исходным данным предъявляются следующие требования:

- небольшой объем, чтобы решение было как можно более быстрым;

- полнота, т.е. обязательно проверки всех предусмотренных разветвлений вычислительного процесса и обработки нестандартных ситуаций;

- реалистичной по форме и содержанию информации о той предметной области, для которой предназначена зада.

Таким образом, решение на основе контрольного примера является о определенным эталоном для проверки корректности алгоритма, реализованного в программном продукте. Если верный результат получается для контрольного примера, то алгоритм работает правильно и его можно применять для любой другой вариативности исходных данных.

Для каждой таблицы создадим CSV-файл, в котором будут храниться данные для последующей вставки базы данных.

CSV-файл – это обычный текстовый файл, в котором хранятся таблицы и данные электронных таблиц. Содержимое представляет собой набор строк, чисел или дат. CSV-файл можно легко импортировать и экспортировать с помощью программ, хранящих данные в таблицах.

Контрольные данные для сущности «Passports» представлены в таблице 1.15.

Таблица 1.15 – Контрольные данные примера для сущности «Passports»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | Series | Number | Surname | Name | FatherName | Sex | DateOfBirth |
| 1 | 1234 | 567890 | Иванов | Иван | Иванович | М | 1990-05-10 |
| 2 | 1234 | 567891 | Петрова | Мария | Сергеевна | Ж | 1985-03-22 |
| 3 | 1234 | 567892 | Сидоров | Алексей | Михайлович | М | 1988-07-30 |
| 4 | 1234 | 567893 | Кузнецова | Анна | Дмитриевна | Ж | 1992-09-19 |
| 5 | 1234 | 567894 | Семёнов | Дмитрий | Николаевич | М | 1987-12-25 |
| 6 | 1234 | 567895 | Леонов | Сергей | Александрович | М | 1991-04-12 |
| 7 | 1234 | 567896 | Морозова | Елена | Игоревна | Ж | 1994-11-11 |
| 8 | 1234 | 567897 | Ковалёв | Александр | Валерьевич | М | 1986-02-05 |
| 9 | 1234 | 567898 | Степанова | Татьяна | Петровна | Ж | 1993-08-30 |
| 10 | 1234 | 567899 | Григорьев | Виктор | Николаевич | М | 1982-10-17 |

Контрольные данные для сущности «Patients» представлены в таблице 1.16

Таблица 1.16 – Контрольные данные примера для сущности «Patients»

|  |  |  |  |
| --- | --- | --- | --- |
| Id | PassportId | CreatedAt | UpdatedAt |
| 1 | 1 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 2 | 2 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 3 | 3 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 4 | 4 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 5 | 5 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |

Контрольные данные для сущности «Doctors» представлены в таблице 1.17

Таблица 1.17 – Контрольные данные примера для сущности «Doctors»

|  |  |  |  |
| --- | --- | --- | --- |
| Id | PassportId | CreatedAt | UpdatedAt |
| 1 | 6 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 2 | 7 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |

Окончание таблицы 1.17

|  |  |  |  |
| --- | --- | --- | --- |
| Id | PassportId | CreatedAt | UpdatedAt |
| 3 | 8 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 4 | 9 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 5 | 10 | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |

Контрольные данные для «WorkingHours» представлены в таблице 1.18.

Таблица 1.18 – Контрольные данные примера для сущности «WorkingHours»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | DoctorId | StartTime | | EndTime | WorkDay | CreatedAt | UpdatedAt |
| 1 | 1 | 08:00:00 | 16:00:00 | | Понедельник | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 |
| 2 | 2 | 09:00:00 | 17:00:00 | | Вториик | 2023-01-02 12:00:00 | 2023-01-02 12:00:00 |
| 3 | 3 | 10:00:00 | 18:00:00 | | Среда | 2023-01-03 12:00:00 | 2023-01-03 12:00:00 |
| 4 | 4 | 08:00:00 | 16:00:00 | | Четверг | 2023-01-04 12:00:00 | 2023-01-04 12:00:00 |
| 5 | 5 | 09:00:00 | 17:00:00 | | Пятница | 2023-01-05 12:00:00 | 2023-01-05 12:00:00 |

Контрольные данные для сущности «DozeUnits» представлены в таблице 1.19.

Таблица 1.19 – Контрольные данные примера для сущности «DozeUnits»

|  |  |
| --- | --- |
| Id | Name |
| 1 | мг. |
| 2 | мл. |
| 3 | г. |
| 4 | л. |
| 5 | шт. |
| 6 | табл. |
| 7 | кап. |

Контрольные данные для сущности «MedicineUsingWays» представлены в таблице 1.20.

Таблица 1.20 – Контрольные данные примера для сущности «MedicineUsingWays»

|  |  |
| --- | --- |
| Id | Name |
| 1 | Перорально |
| 2 | Инъекция |
| 3 | Топически |

Контрольные данные для сущности «Medicines» представлены в таблице 1.21.

Таблица 1.21 – Контрольные данные примера для сущности «Medicines»

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | DozeUnitId | MedicineUsingWayId |
| 1 | Парацетамол | 1 | 1 |
| 2 | Ибупрофен | 2 | 2 |
| 3 | Амоксициллин | 1 | 3 |
| 4 | Инсулин | 3 | 1 |
| 5 | Диклофенак | 2 | 2 |
| 6 | Омепразол | 1 | 3 |
| 7 | Лизиноприл | 2 | 1 |
| 8 | Аторвастатин | 1 | 1 |
| 9 | Аспирин | 2 | 2 |
| 10 | Метформин | 3 | 3 |

Контрольные данные для сущности «Consultations» представлены в таблице 1.22.

Таблица 1.22 – Контрольные данные примера для сущности «Consultations»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Location | StartTime | EndTime | DoctorId | PatientId |
| 1 | Кабинет 101 | 2023-01-15 08:00:00 | 2023-01-15 09:15:00 | 1 | 1 |
| 2 | Кабинет 102 | 2023-01-15 09:15:00 | 2023-01-15 10:30:00 | 2 | 2 |
| 3 | Кабинет 103 | 2023-01-15 10:30:00 | 2023-01-15 11:45:00 | 3 | 3 |
| 4 | Кабинет 104 | 2023-01-15 11:45:00 | 2023-01-15 12:00:00 | 4 | 4 |
| 5 | Кабинет 105 | 2023-01-15 12:00:00 | 2023-01-15 13:15:00 | 5 | 5 |
| 6 | Кабинет 106 | 2023-01-15 13:15:00 | 2023-01-15 14:30:00 | 6 | 6 |
| 7 | Кабинет 107 | 2023-01-15 14:30:00 | 2023-01-15 15:45:00 | 7 | 7 |
| 8 | Кабинет 108 | 2023-01-15 15:45:00 | 2023-01-15 16:00:00 | 8 | 8 |
| 9 | Кабинет 109 | 2023-01-15 16:00:00 | 2023-01-15 17:30:00 | 9 | 9 |
| 10 | Кабинет 110 | 2023-01-15 17:30:00 | 2023-01-15 19:00:00 | 10 | 10 |

Контрольные данные для сущности «Prescriptions» представлены в таблице 1.23.

Таблица 1.23 – Контрольные данные примера для сущности «Prescriptions»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Description | CreatedAt | MedicineId | ConsultationId |
| 1 | Парацетамол | 2023-01-15 08:00:00 | 1 | 1 |
| 2 | Ибупрофен | 2023-01-15 09:15:00 | 2 | 2 |
| 3 | Амоксициллин | 2023-01-15 10:30:00 | 3 | 3 |
| 4 | Инсулин | 2023-01-15 11:45:00 | 4 | 4 |
| 5 | Диклофенак | 2023-01-15 12:00:00 | 5 | 5 |
| 6 | Омепразол | 2023-01-15 13:15:00 | 6 | 6 |
| 7 | Лизиноприл | 2023-01-15 14:30:00 | 7 | 7 |
| 8 | Аторвастатин | 2023-01-15 15:45:00 | 8 | 8 |
| 9 | Аспирин | 2023-01-15 16:00:00 | 9 | 9 |
| 10 | Метформин | 2023-01-15 17:30:00 | 10 | 10 |

Контрольные данные для сущности «Symptoms» представлены в таблице 1.24

Таблица 1.24 – Контрольные данные примера для сущности «Symptoms»

|  |  |
| --- | --- |
| Id | Name |
| 1 | Головная боль |
| 2 | Жар |
| 3 | Кашель |
| 4 | Тошнота |
| 5 | Головокружение |
| 6 | Боль в теле |
| 7 | Боль в груди |
| 8 | Одышка |
| 9 | Боль в животе |
| 10 | Боль в горле |

Контрольные данные для сущности «ConsultationSymptoms» представлены в таблице 1.25.

Таблица 1.25 – Контрольные данные примера для сущности «ConsultationSymptoms»

|  |  |  |
| --- | --- | --- |
| Id | SymptomId | ConsultationId |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 2 |
| 4 | 4 | 2 |
| 5 | 5 | 3 |
| 6 | 6 | 3 |
| 7 | 7 | 4 |
| 8 | 8 | 4 |
| 9 | 9 | 5 |
| 10 | 10 | 5 |

Контрольные данные для сущности «Roles» представлены в таблице 1.26.

Таблица 1.26 – Контрольные данные примера для сущности «Roles»

|  |  |  |
| --- | --- | --- |
| Id | Name | CreatedAt |
| 1 | Администратор | 2023-01-01 08:00:00 |
| 2 | Главный врач | 2023-01-01 08:30:00 |
| 3 | Врач | 2023-01-01 09:00:00 |
| 4 | Интерн | 2023-01-01 09:30:00 |

Контрольные данные для сущности «Users» представлены в таблице 1.27.

Таблица 1.27 – Контрольные данные примера для сущности «Users»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Name | Password | Email | CreatedAt | UpdatedAt | RoleId |
| 1 | Джон Смит | pass1234 | john.smith@email.com | 2023-01-01 08:00:00 | 2023-01-01 08:00:00 | 1 |
| 2 | Эмили Джонсон | pass5678 | emily.johnson@email.com, | 2023-01-01 08:30:00 | 2023-01-01 08:30:00 | 2 |
| 3 | Майкл Браун | pass9101 | michael.brown@email.com | 2023-01-01 09:00:00 | 2023-01-01 09:00:00 | 2 |
| 4 | Оливия Уильямс | pass1122 | olivia.williams@email.com | 2023-01-01 09:30:00 | 2023-01-01 09:30:00 | 3 |
| 5 | Дэвид Джонс | pass3344 | david.jones@email.com | 2023-01-01 10:00:00 | 2023-01-01 10:00:00 | 2 |
| 6 | Анна Миллер | pass5566 | anna.miller@email.com | 2023-01-01 10:30:00 | 2023-01-01 10:30:00 | 3 |
| 7 | Джеймс Дэвис | pass7788 | james.davis@email.com | 2023-01-01 11:00:00 | 2023-01-01 11:00:00 | 2 |
| 8 | Изабелла Гарсия | pass9900 | isabella.garcia@email.com | 2023-01-01 11:30:00 | 2023-01-01 11:30:00 | 3 |
| 9 | Даниэль Мартинес | pass1010 | daniel.martinez@email.com | 2023-01-01 12:00:00 | 2023-01-01 12:00:00 | 2 |
| 10 | Шарлотта Эрнандес | pass1212 | charlotte.hernandez@email.com | 2023-01-01 12:30:00 | 2023-01-01 12:30:00 | 4 |

Контрольные данные для сущности «MedicineSideEffects» представлены в таблице 1.28.

Таблица 1.28 – Контрольные данные примера для сущности «MedicineSideEffects»

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Description | MedicineId |
| 1 | Тошнота | Тошнота | 1 |
| 2 | Головокружение | Ощущение легкости в голове | 2 |
| 3 | Сыпь | Кожное раздражение | 3 |
| 4 | Мигрень | Сильная головная боль | 4 |
| 5 | Усталость | Чрезмерная усталость и стресс | 5 |
| 6 | Боль в животе | Боль в животе | 6 |
| 7 | Кашель | Сухой кашель | 7 |
| 8 | Боль в мышцах | Боль в мышцах | 8 |
| 9 | Язва | Пептическая язва | 9 |
| 10 | Понос | Жидкий стул | 10 |

# 2 Программные разработки

## 2.1 Создание базы данных

Создание базы данных в SQL Server осуществляется с помощью команды CREATE DATABASE, где указывается уникальное имя для новой базы данных, соответствующее правилам идентификаторов SQL Server. Создание базы происходит в системной базе master, которая содержит необходимые системные таблицы для работы с самой системы баз данных.

Для создания таблиц используется инструкция CREATE TABLE в языке T-SQL. Эти инструкции могут быть встроены в процедуру.

Целостность данных обеспечивается с помощью ограничений целостности. Ограничение NOT NULL позволяет определить, можно ли хранить в столбце NULL значения или нет. PRIMARY KEY задает первичный ключ таблицы, а FOREIGN KEY устанавливает связь между данными в разных таблицах, обеспечивая ссылочную целостность.

Ограничение FOREIGN KEY предотвращает наличие записей в таблице, которые ссылаются на несуществующие записи в другой таблице. Оно гарантирует корректность данных, предотвращая некорректные ссылки между записями.

UNIQUE — это ограничение, которое гарантирует уникальность значений в столбце или комбинации столбцов.

Таблицы на языке Transact-SQL создаются при помощи инструкции CREATE TABLE. Листинг всех таблиц представлен в приложении Б.

На рисунке 2.1 изображена таблица «Medicines» в Microsoft SQL Server



Рисунок 2.1- Таблица «Medicines»

Аналогичные действия проводятся для создания всех таблиц БД «ПОЛИКЛИНИКА»

На рисунке 2.2 изображена диаграмма базы данных БД «ПОЛИКЛИНИКА».

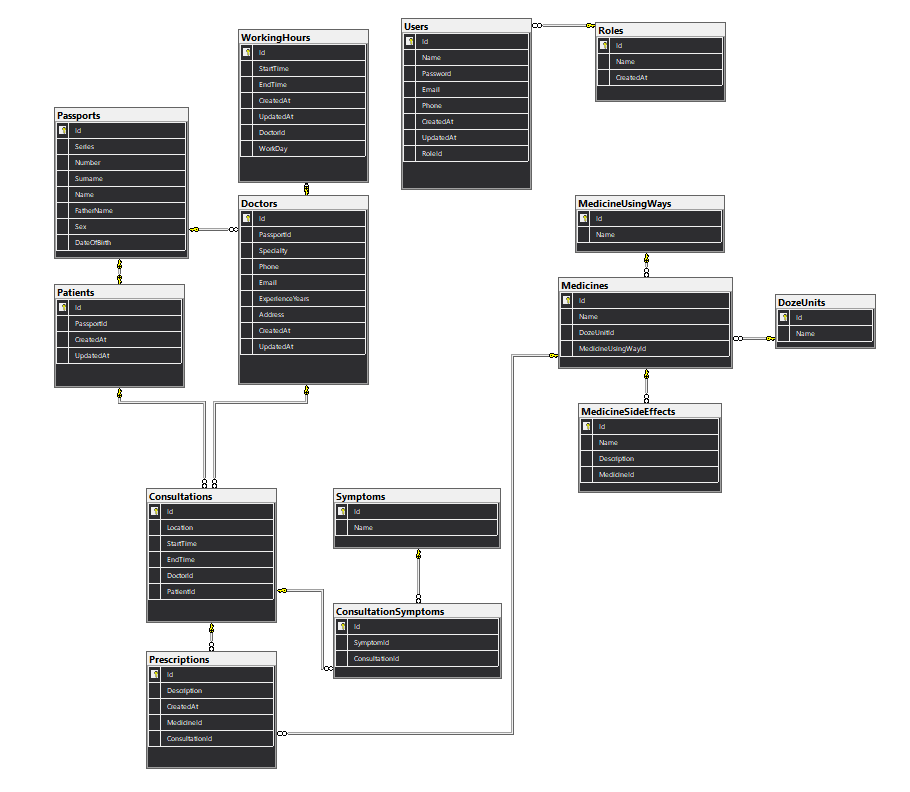


Рисунок 2.2 – Диаграмма базы данных «ПОЛИКЛИНИКА»

## 2.2 Заполнение данными

Созданные таблицы необходимо заполнить данными из CSV-файлов, которые были созданы ранее.

Чтобы импортировать данные из CSV-файла, создадим временные таблицы для каждой сущности и сохраним в них данные. Во временных таблицах данные будут сохранены в виде строк, а после этого перенесены в основные таблицы и приведены к нужному нам типу данных.

На рисунке 2.3 изображена таблица «Medicines» в CSV-файле.

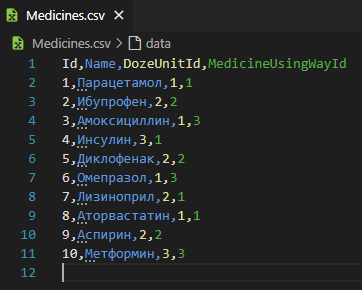


Рисунок 2.3 – Таблица «Medicines» в CSV-файле

Инструкция BULK INSERT выполняет импорт файла данных в таблицу или представление базы данных в формате, указанном пользователем. В предложении WITH для данной инструкции можно задавать множество опций.

На рисунке 2.4 изображена заполненная таблица «Medicines» в Microsoft SQL Server



Рисунок 2.4 – Заполненная таблица «Medicines»

Аналогичные действия проводятся для всех таблиц БД «ПОЛИКЛИНИКА»

Листинг импорта данных из CSV-файла в базу данных «ПОЛИКЛИНИКА» представлен в Приложении В.

## 2.3 Разработка триггеров

Триггер – это механизм, который вызывается, когда в указанной таблице происходит определенное действие. Каждый триггер имеет следующие основные составляющие:

- имя;

- действие;

- исполнение.

Триггер создается с помощью инструкции CREATE TRIGGER.

Можно задать тип триггера с помощью двух дополнительных параметров: AFTER и INSTEAD OF. Триггеры типа AFTER вызываются после выполнения действия, запускающего триггер, а триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггера.

Параметры INSERT, UPDATE и DELETE задают действия триггера. Под действием триггера имеются ввиду инструкция Transact-SQL, которая запускает триггер. Допускается любая комбинациях этих трех инструкций.

Триггер типа FOR можно использовать для выполнения следующих операций:

– создание журнала аудита действий в таблицах базы данных;

– реализация бизнес-правил;

– принудительное обеспечение ссылочной целостности.

Листинг создания триггеров для базы данных «ПОЛИКЛИНИКА» представлен в приложении Г.

На рисунке 2.5 представлена ошибка о неправильно заполненных данных в таблице «Consultations»

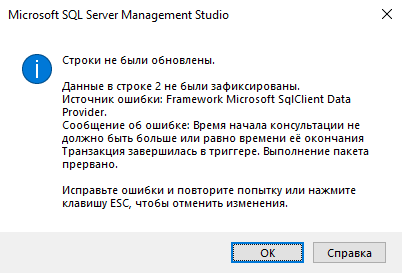


Рисунок 2.5 – Ошибка изменения поля «EndTime» в таблице «Consultations»

## 2.4 Разработка представлений

Представление – это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из ряда именованных столбцов и строк данных. Пока представление не будет проиндексировано, оно не существует в базе данных как хранимая совокупность значений. Строки и столбцы данных извлекаются из таблиц, указанных в определяющем представление запросе и динамически создаваемых при обращениях к представлению.

Представление выполняет функцию фильтра базовых таблиц, на которые оно ссылается. Определяющий представление запрос может быть инициирован в одной или нескольких таблицах или в других представлениях текущей или других баз данных. Кроме того, для определения представлений с данными из нескольких разнородных источников можно использовать распределенные запросы. Это полезно, например, если нужно объединить структурированные подобным образом данные, относящиеся к разным серверам, каждый из которых хранит данные конкретного отдела организации.

Представление обычно используется для направления, упрощения и настройки восприятия каждым пользователем информации базы данных. Представления могут использоваться как механизмы безопасности, давая возможность пользователям обращаться к данным через представления, но не предоставляя им разрешений на непосредственный доступ к базовым таблицам, лежащих на основе представлений. Представления могут использоваться для обеспечения интерфейса обратной совместимости, моделирующего таблицу, которая существует, но схема которой изменилась. Представления могут также использоваться при прямом и обратном копировании данных в SQL Server для повышения производительности и секционирования данных.

Общий синтаксис для создания представлений:

CREATE VIEW view\_name [(column\_list)]

[WITH {ENCRYPTION SCHEMABINDING | VIEW\_METADATA}]

AS select\_statement

[WITH CHECK OPTION]

Для базы данных "ПОЛИКЛИНИКА" необходимо создать представление текущих консультаций. Это представление будет отображать информацию о консультациях, которые проводятся в данный момент, включая данные о врачах и пациентах.

С помощью инструкции CREATE VIEW создадим представление CurrentConsultationsView. Вот код для его создания:

-- Представление текущих консультаций

CREATE OR ALTER VIEW CurrentConsultationsView AS

SELECT

C.Id [Код консультации],

C.StartTime [Начало],

C.EndTime [Конец],

DP.Name [Имя доктора],

DP.Surname [Фамилия доктора],

DP.FatherName [Отчество доктора],

PP.Name [Имя пациента],

PP.Surname [Фамилия пациента],

PP.FatherName [Отчество пациента]

FROM

dbo.Consultations C

JOIN

dbo.Doctors D ON D.Id = C.DoctorId

JOIN

dbo.Passports DP ON DP.Id = D.PassportId

JOIN

dbo.Patients P ON P.Id = C.PatientId

JOIN

dbo.Passports PP ON PP.Id = P.PassportId

WHERE

C.StartTime <= SYSDATETIME() AND C.EndTime >= SYSDATETIME()

После создания представления можно использовать инструкцию SELECT для выборки данных:

SELECT \* FROM CurrentConsultationsView

В окне результатов появится таблица с текущими консультациями, что показано на рисунке 2.6.

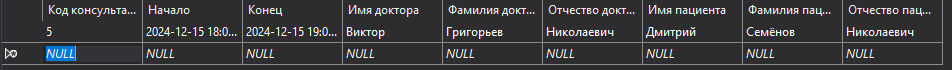


Рисунок 2.6 – представление для отображения текущих консультаций

Следующим шагом является создание представления для отображения всех рецептов, включая подробную информацию о лекарствах и связанных консультациях.

Используем следующую инструкцию для создания представления PrescriptionsDetailsView:

-- Представление всех рецептов

CREATE OR ALTER VIEW PrescriptionsDetailsView AS

SELECT

R.Id AS [Код рецепта],

R.Description [Описание рецепта],

M.Name [Наименование лекарства],

C.StartTime [Начало консультации],

C.EndTime [Конец консультации],

PP.Name [Имя пациента],

PP.Surname [Фамилия пациента],

PP.FatherName [Отчество пациента]

FROM

dbo.Prescriptions R

JOIN

dbo.Medicines M ON M.Id = R.MedicineId

JOIN

dbo.Consultations C ON C.Id = R.ConsultationId

JOIN

dbo.Patients P ON P.Id = C.PatientId

JOIN

dbo.Passports PP ON PP.Id = P.PassportId

Вызовем это представление следующим образом:

SELECT \* FROM PrescriptionsDetailsView

Далее в окне результатов появится таблица представления, это показано на рисунке 2.7.

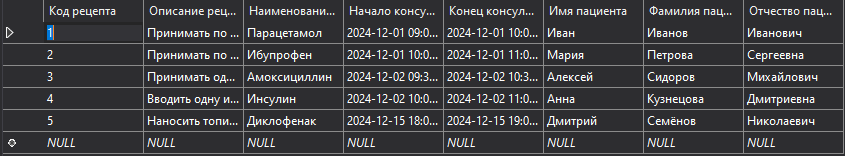


Рисунок 2.7 – представление для вывода информации о всех рецептах

Для более детального анализа симптомов, выявленных во время консультаций, создадим представление ConsultationSymptomsView. Это представление будет агрегировать симптомы, относящиеся к каждой консультации.

Код для создания данного представления:

-- Представление симптомов по консультациям

CREATE OR ALTER VIEW ConsultationSymptomsView AS

SELECT

C.Id [Код консультации],

STRING\_AGG(S.Name, ', ') [Симптомы],

PP.Name [Имя пациента],

PP.Surname [Фамилия пациента],

PP.FatherName [Отчество пациента],

DP.Name [Имя доктора],

DP.Surname [Фамилия доктора],

DP.FatherName [Отчество доктора],

C.StartTime [Начало консультации],

C.EndTime [Конец консультации]

FROM

dbo.ConsultationSymptoms CS

JOIN

dbo.Consultations C ON C.Id = CS.ConsultationId

JOIN

dbo.Symptoms S ON S.Id = CS.SymptomId

JOIN

dbo.Patients P ON P.Id = C.PatientId

JOIN

dbo.Passports PP ON PP.Id = P.PassportId

JOIN

dbo.Doctors D ON C.DoctorId = D.Id

JOIN

dbo.Passports DP ON DP.Id = D.PassportId

GROUP BY

C.Id, PP.Name, PP.Surname, PP.FatherName, DP.Name, DP.Surname, DP.FatherName, C.StartTime, C.EndTime

Для выборки данных обращаемся к представлению с помощью:

SELECT \* FROM ConsultationSymptomsView

Результаты запроса отобразят симптомы и соответствующие данные о консультациях, как показано на рисунке 2.8.

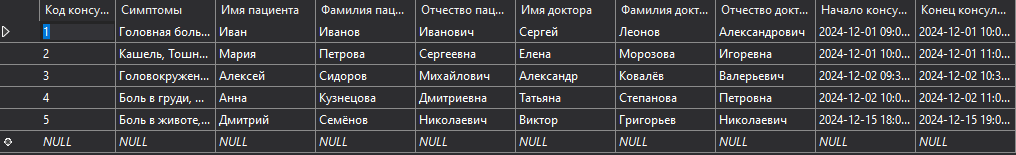


Рисунок 2.8 – представление с симптомами по консультациям

Заключительное представление будет отображать количество врачей по каждой специальности, что поможет в управлении кадрами и планировании.

Код для создания представления DoctorsBySpecialtyView:

-- Представление врачей по специальности

CREATE OR ALTER VIEW DoctorsBySpecialtyView AS

SELECT

D.Specialty [Специальность],

COUNT(D.Id) [Количество докторов]

FROM

dbo.Doctors D

GROUP BY

D.Specialty

Для выборки данных обращаемся к представлению с помощью:

SELECT \* FROM DoctorsBySpecialtyView

В результате выполнения данного запроса отобразится таблица с количеством врачей, сгруппированных по специальностям, что иллюстрировано на рисунке 2.9.

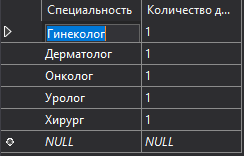


Рисунок 2.9 – представление для вывода кол-ва врачей по специальности

## 2.5 Разработка хранимых процедур

Хранимая процедура в SQL Server представляет собой группу из одного или нескольких операторов Transact-SQL или ссылку на метод CLR Microsoft .NET Framework. Процедуры аналогичны конструкциям в других языках программирования, поскольку обеспечивают следующее:

- обрабатывают входные параметры и возвращают вызывающей программе значения в виде выходных параметров;

- содержат программные инструкции, которые выполняют операции в базе данных, включая вызов других процедур;

- возвращают значение состояния вызывающей программе, таким образом передавая сведения об успешном или неуспешном завершении (и причины последнего).

Преимущества использования хранимых процедур:

- снижение сетевого трафика между клиентами и сервером Команды в процедуре выполняются как один пакет кода. Это позволяет существенно сократить сетевой трафик между сервером и клиентом, поскольку по сети отправляется только вызов на выполнение процедуры. Без инкапсуляции кода, предоставляемой процедурой, по сети бы пришлось пересылать все отдельные строки кода;

- большая безопасность. Многие пользователи и клиентские программы могут выполнять операции с базовыми объектами базы данных посредством процедур, даже если у них нет прямых разрешений на доступ к базовым объектам. Процедура проверяет, какие из процессов и действий могут выполняться, и защищает базовые объекты базы данных. Это устраняет необходимость предоставлять разрешения на уровне индивидуальных объектов и упрощает формирование уровней безопасности;

- при вызове процедуры через сеть виден только вызов на выполнение процедуры. Таким образом, злоумышленники не могут просматривать имена объектов таблиц и баз данных, внедрять собственные инструкции Transsact-SQL или выполнять поиск критически важных данных;

- повторное использование кода. Если какой-то код многократно используется в операции базы данных, то отличным решением будет произвести его инкапсуляцию в процедуры. Это устранит необходимость излишнего копирование того же кода, снизит уровень несогласованности когда и позволит осуществлять доступ к коду любым пользователем или приложением, имеющим необходимые разрешения.

- более легкое обслуживание. Если клиентские приложения вызывают процедуры, а операции без данных остаются лишь на уровне данных, то для внесения изменений в основную базу данных будет достаточно обновить только процедуры. Уровень приложения остается незатронутым изменениями в схемах баз данных, связях или процессах;

- повышение производительности. По умолчанию компиляция процедуры и создание плана выполнения, используемого для последующих выполнений, производится при ее первом запуске. Поскольку обработчику запросов не нужно создавать новый план, обычно обработка процедур занимает меньше времени.

Хранимые процедуры создаются посредством инструкции CREATE PROCEDURE, которая имеет следующий синтаксис:

CREATE PROC[EDURE] [schema\_name.]proc\_name

[({@param} type1 [VARYING] [=default] [OUTPUT])] {, …}

[WITH {RECOMPILE | ENCRYPTION | EXECUTE AS ‘user\_name’}]

[FOR REPLICATION]

AS batch | EXTERNAL NAME method\_name

Жизненный цикл хранимой процедуры состоит из двух этапов: ее создания и ее выполнения. Каждая процедура создается один раз, а выполняется многократно. Хранимая процедура выполняется посредством инструкции EXECUTE пользователем, который является владельцем процедуры или обладает правом EXECUTE для доступа к этой процедуре (см. главу 12). Инструкция EXECUTE имеет следующий синтаксис:

[[EXEC[UTE]] [@return\_status =] {proc\_name

l@proc\_name\_var}

{[[@parameter1=]value[[@parameter1=]@variable[OUTPUT]][DEFAULT}… [WITH RECOMPILE]

Для удобства использования базы данных нужны хранимые процедуры для поиска информации в базе, вставку данных в таблицы и создания архива данных.

С помощью инструкции CREATE PROCEDURE необходимо создать несколько хранимых процедур для базы данных "Healthcare". Далее представлен код создания каждой из процедур, которые обеспечивают эффективное управление данными.

С помощью инструкции CREATE OR ALTER PROC нужно создать хранимую процедуру InsertDataFromCSV, которая будет ответственна за вставку данных в указанную таблицу из CSV-файла. Код создания данной процедуры представлен ниже:

CREATE OR ALTER PROC InsertDataFromCSV

@TableName NVARCHAR(128),

@AbsoluteFilePath NVARCHAR(500),

@FirstRowContainsHeaders BIT = FALSE,

@FieldTerminator NCHAR(1) = ',',

@RowTerminator NVARCHAR(4) = '0x0a',

@ShowSQL BIT = FALSE

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX)

BEGIN TRAN

SET @SQL = 'SET IDENTITY\_INSERT ' + QUOTENAME(@TableName) + ' ON'

EXEC sp\_executesql @SQL

-- Формируем команду BULK INSERT

SET @SQL = N'BULK INSERT ' + QUOTENAME(@TableName) +

N'FROM ' + QUOTENAME(@AbsoluteFilePath, '''') +

N'WITH (

CODEPAGE = 65001,

FIELDTERMINATOR = ''' + @FieldTerminator + ''',

ROWTERMINATOR = ''' + @RowTerminator + ''',

FIRSTROW = ' + IIF(@FirstRowContainsHeaders = 1, '2', '1') + '

)'

BEGIN TRY

IF @ShowSQL = 1 PRINT @SQL

EXEC sp\_executesql @SQL

COMMIT

PRINT 'Данные успешно вставлены.'

END TRY

BEGIN CATCH

PRINT 'Произошла ошибка: ' + ERROR\_MESSAGE()

ROLLBACK

END CATCH

-- Отключаем IDENTITY\_INSERT обратно

SET @SQL = 'SET IDENTITY\_INSERT ' + QUOTENAME(@TableName) + ' OFF'

EXEC sp\_executesql @SQL

END

GO

Далее нужно создать хранимую процедуру SelectPatientConsultations, которая выводит все консультации для указанного пациента по его идентификатору:

CREATE OR ALTER PROC SelectPatientConsultations

@PatientId INT

AS

SELECT \* FROM dbo.Consultations

WHERE PatientId = @PatientId

GO

Следом нужно создать хранимую процедуру SelectDoctorsBySpecialty, которая выводит всех докторов на основе их специальности:

CREATE OR ALTER PROC SelectDoctorsBySpecialty

@Specialty NVARCHAR(50)

AS

SELECT \* FROM dbo.Doctors

WHERE Specialty = @Specialty

GO

Финальная хранимая процедура SelectRecentPrescriptions, которая возвращает рецепты, выданные указанным доктором за последние 30 дней:

CREATE OR ALTER PROC SelectRecentPrescriptions

@DoctorId INT

AS

SELECT

R.Id AS PrescriptionId,

R.Description,

PP.Name AS PatientName,

PP.Surname AS PatientSurname,

C.StartTime

FROM

dbo.Prescriptions R

JOIN

dbo.Consultations C ON C.Id = R.ConsultationId

JOIN

dbo.Patients P ON P.Id = C.PatientId

JOIN

dbo.Passports PP ON PP.Id = P.PassportId

WHERE

C.DoctorId = @DoctorId AND C.StartTime >= DATEADD(DAY, -30, GETDATE())

GO

После успешного выполнения создания вышеперечисленных хранимых процедур они станут доступны в разделе «Хранимые процедуры» вашей базы данных в MS SQL Server (см. рисунок 2.10).

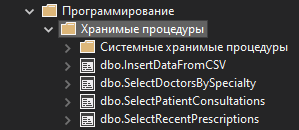


Рисунок 2.10 – созданные хранимые процедуры в MS SQL

# Заключение

В результате программных разработок была создана база данных для управления здравоохранением. Все таблицы заполнены корректными данными, и разработаны триггеры, представления и другие хранимые процедуры, что позволяет эффективно управлять данными о пациентах, докторах, консультациях и рецептах.

При этом решены следующие задачи:

* изучены синтаксис языка Transact-SQL и СУБД Microsoft SQL Server;
* проанализирована предметная область базы данных и ее атрибуты;
* выполнено построение информационно-логической и даталогической модели базы данных;
* разработаны входные данные для таблиц БД;
* выполнено создание и заполнение БД;
* созданы триггеры, необходимые для БД;
* созданы представления для БД;
* созданы хранимые процедуры для БД;

Для реализации основной цели курсовой работы была использована СУБД Miscoroft SQL Server.

Полученный программный продукт готов к эксплуатации и может быть использован различными поликлиниками для учета, хранения и вывода информации о пациентах, консультациях, врачах и рецептах. В настоящее время для оказания медицинских услуг поликлиники нуждаются в эффективных системах управления данными для координации потоков информации о пациентах и медицинских процедурах. В связи с этим всегда существует необходимость в создании новых элементов и операций, что позволяет вносить изменения в базу данных для улучшения ее функциональности.

# Список используемых источников

* 1. Кореньков В.В. Технологии баз данных. Проектирование реляционных баз данных – Москва : Курс, 2022. – 128 с.
  2. Уолтер Шилдс. SQL: быстрое погружение.
  3. Алан Бьюли. Изучаем SQL, 2007.
  4. Иванов А. Д. Основы проектирования реляционных баз данных.
  5. Петров С. Л. Основы SQL: Руководство по реляционным базам данных.
  6. Джо Селко. SQL для умников: Advanced SQL Programming.
  7. Учебник с теорией и задачами по изучению SQL Academy [Электронный ресурс] URL: <https://sql-academy.org/ru>
  8. Смирнов И. В. Реляционные базы данных: теория и практика.
  9. Сидорова Е. И. СУБД и язык SQL в проектировании информационных систем.
  10. Козлов А. Д. Основы работы с базами данных: практическое руководство.

# Приложение А

Даталогическая модель базы данных:

Таблица Passports (хранение паспортных данных)

Первичный ключ: Id

Поля:

Id (INT) – Идентификатор паспорта

Series (NVARCHAR(10)) – Серия паспорта

Number (NVARCHAR(20)) – Номер паспорта

Surname (NVARCHAR(30)) – Фамилия

Name (NVARCHAR(30)) – Имя

FatherName (NVARCHAR(30)) – Отчество

Sex (NCHAR(1)) – Пол

DateOfBirth (DATE) – Дата рождения

Таблица Patients (данные о пациентах)

Первичный ключ: Id

Внешний ключ: PassportId → Passports.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор пациента

PassportId (INT) – Идентификатор паспорта

CreatedAt (DATETIME2(7)) – Дата создания записи

UpdatedAt (DATETIME2(7)) – Дата обновления записи

Таблица Doctors (данные о врачах)

Первичный ключ: Id

Внешний ключ: PassportId → Passports.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор врача

Specialty (NVARCHAR(50)) – Специализация

Phone (NVARCHAR(20)) – Телефон врача

Email (NVARCHAR(100)) – Электронная почта

ExperienceYears (INT) – Количество лет опыта

Address (NVARCHAR(200)) – Адрес

CreatedAt (DATETIME2(7)) – Дата создания записи

UpdatedAt (DATETIME2(7)) – Дата обновления записи

PassportId (INT) – Идентификатор паспорта

Таблица WorkingHours (рабочее время врачей)

Первичный ключ: Id

Внешний ключ: DoctorId → Doctors.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор записи

StartTime (TIME(7)) – Время начала работы

EndTime (TIME(7)) – Время окончания работы

CreatedAt (DATETIME2(7)) – Дата создания записи

UpdatedAt (DATETIME2(7)) – Дата обновления записи

DoctorId (INT) – Идентификатор врача

WorkDay (NVARCHAR(11)) – Рабочий день

Таблица DozeUnits (единицы дозировки)

Первичный ключ: Id

Поля:

Id (INT) – Идентификатор дозировки

Name (NVARCHAR(20)) – Наименование дозировки

Таблица MedicineUsingWays (способы применения лекарств)

Первичный ключ: Id

Поля:

Id (INT) – Идентификатор способа применения

Name (NVARCHAR(20)) – Наименование способа применения

Таблица Medicines (данные о лекарствах)

Первичный ключ: Id

Внешние ключи:

DozeUnitId → DozeUnits.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

MedicineUsingWayId → MedicineUsingWays.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор лекарства

Name (NVARCHAR(100)) – Наименование лекарства

DozeUnitId (INT) – Идентификатор единицы дозировки

MedicineUsingWayId (INT) – Идентификатор способа применения

Таблица Consultations (врачебные приемы)

Первичный ключ: Id

Внешние ключи:

DoctorId → Doctors.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

PatientId → Patients.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор консультации

Location (NVARCHAR(100)) – Место проведения консультации

StartTime (DATETIME2(7)) – Время начала консультации

EndTime (DATETIME2(7)) – Время окончания консультации

DoctorId (INT) – Идентификатор доктора

PatientId (INT) – Идентификатор пациента

Таблица Prescriptions (выписанные рецепты)

Первичный ключ: Id

Внешние ключи:

MedicineId → Medicines.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

ConsultationId → Consultations.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор назначения

Description (NVARCHAR(200)) – Описание назначения

CreatedAt (DATETIME2(7)) – Дата создания назначения

MedicineId (INT) – Идентификатор лекарства

ConsultationId (INT) – Идентификатор осмотра

Таблица Symptoms (симптомы)

Первичный ключ: Id

Поля:

Id (INT) – Идентификатор симптома

Name (NVARCHAR(100)) – Наименование симптома

Таблица ConsultationSymptoms (выявленные симптомы)

Первичный ключ: Id

Внешние ключи:

SymptomId → Symptoms.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

ConsultationId → Consultations.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор записи

SymptomId (INT) – Идентификатор симптома

ConsultationId (INT) – Идентификатор осмотра

Таблица Roles (роли пользователей)

Первичный ключ: Id

Поля:

Id (INT) – Идентификатор роли

Name (NVARCHAR(20)) – Наименование роли

CreatedAt (DATETIME2(7)) – Дата создания роли

Таблица Users (пользователи)

Первичный ключ: Id

Внешний ключ: RoleId → Roles.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор пользователя

Name (NVARCHAR(20)) – Имя пользователя

Password (NCHAR(60)) – Пароль (хэшированный)

Email (NVARCHAR(100)) – Электронная почта

Phone (NVARCHAR(20)) – Телефон пользователя

CreatedAt (DATETIME2(7)) – Дата создания записи

UpdatedAt (DATETIME2(7)) – Дата обновления записи

RoleId (INT) – Идентификатор роли

Таблица MedicineSideEffects (данные о побочных эффектах лекарств)

Первичный ключ: Id

Внешние ключи:

MedicineId → Medicines.Id (NOT NULL, CASCADE UPDATE, RESTRICT DELETE)

Поля:

Id (INT) – Идентификатор побочного эффекта

Name (NVARCHAR(50)) – Название побочного эффекта

Description (NVARCHAR(200)) – Описание побочного эффекта

MedicineId (INT) – Идентификатор лекарства

# Приложение Б

Листинг создания таблиц для базы данных:

USE [Healthcare]

GO

CREATE FULLTEXT CATALOG [FT\_Catalog\_IdentityDocuments]

GO

CREATE TABLE [dbo].[Consultations](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Location] nvarchar(100) NOT NULL,

[StartTime] datetime2(7) NOT NULL,

[EndTime] datetime2(7) NOT NULL,

[DoctorId] int NULL FOREIGN KEY REFERENCES Doctors([Id]),

[PatientId] int NULL FOREIGN KEY REFERENCES Patients([Id]),

UNIQUE (StartTime, EndTime, DoctorId, PatientId)

)

GO

CREATE TABLE [dbo].[ConsultationSymptoms](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[SymptomId] int NULL FOREIGN KEY REFERENCES Symptoms([Id]),

[ConsultationId] int NULL FOREIGN KEY REFERENCES Consultations([Id])

)

GO

CREATE TABLE [dbo].[Doctors](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Specialty] nvarchar(50) NOT NULL,

[Phone] nvarchar(20) NOT NULL,

[Email] nvarchar(100) NOT NULL,

[ExperienceYears] int NULL,

[Address] nvarchar(200) NOT NULL,

[CreatedAt] datetime2(7) NULL,

[UpdatedAt] datetime2(7) NULL,

[PassportId] int NULL FOREIGN KEY REFERENCES Passports([Id])

)

GO

CREATE TABLE [dbo].[DozeUnits](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(20) NOT NULL

)

GO

CREATE TABLE [dbo].[Medicines](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(100) NOT NULL,

[DozeUnitId] int NULL FOREIGN KEY DozeUnits([Id]),

[MedicineUsingWayId] int NULL FOREIGN KEY REFERENCES MedicineUsingWays([Id])

)

GO

CREATE TABLE [dbo].[MedicineSideEffects](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(50) NOT NULL,

[Description] nvarchar(200) NULL,

[MedicineId] int NULL FOREIGN KEY REFERENCES Medicines([Id])

)

GO

CREATE TABLE [dbo].[MedicineUsingWays](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(20) NULL

)

GO

CREATE TABLE [dbo].[Passports](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Series] nvarchar(10) NOT NULL,

[Number] nvarchar(20) NOT NULL,

[Surname] nvarchar(30) NOT NULL,

[Name] nvarchar(30) NOT NULL,

[FatherName] nvarchar(30) NULL,

[Sex] nchar(1) NULL CHECK ([Sex]=N'Ж' OR [Sex]=N'М'),

[DateOfBirth] date NOT NULL

)

GO

CREATE TABLE [dbo].[Patients](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[PassportId] int NULL FOREIGN KEY REFERENCES Passports([Id]),

[CreatedAt] datetime2(7) NULL,

[UpdatedAt] datetime2(7) NULL

)

GO

CREATE TABLE [dbo].[Prescriptions](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Description] nvarchar(200) NULL,

[CreatedAt] datetime2(7) NULL,

[MedicineId] int NULL FOREIGN KEY REFERENCES Medicines([Id]),

[ConsultationId] int NULL FOREGIN KEY REFERENCES Consultations([Id])

)

GO

CREATE TABLE [dbo].[Roles](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(20) NOT NULL,

[CreatedAt] datetime2(7) NULL

]

GO

CREATE TABLE [dbo].[Symptoms](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(100) NOT NULL

)

GO

CREATE TABLE [dbo].[Users](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[Name] nvarchar(20) NOT NULL UNIQUE,

[Password] nchar(60) NOT NULL,

[Email] nvarchar(100) NOT NULL UNIQUE CHECK ([Email] LIKE '%\_@\_\_%.\_\_%'),

[Phone] nvarchar(20) NOT NULL UNIQUE,

[CreatedAt] datetime2(7) NULL,

[UpdatedAt] datetime2(7) NULL,

[RoleId] int NULL FOREIGN KEY REFERENCES Roles([Id])

)

GO

CREATE TABLE [dbo].[WorkingHours](

[Id] int IDENTITY NOT NULL PRIMARY KEY,

[StartTime] time(7) NOT NULL,

[EndTime] time(7) NOT NULL,

[CreatedAt] datetime2(7) NULL,

[UpdatedAt] datetime2(7) NULL,

[DoctorId] int NULL FOREIGN KEY REFERENCES Doctors([Id]),

[WorkDay] nvarchar(11) NULL

)

GO

# Приложение В

Листинг для импорта данных из CSV файлов в базу данных:

EXEC InsertDataFromCSV 'Passports', 'E:\CourseWork\csv\Passports.csv', TRUE

EXEC InsertDataFromCSV 'Patients', 'E:\CourseWork\csv\Patients.csv', TRUE

EXEC InsertDataFromCSV 'Doctors', 'E:\CourseWork\csv\Doctors.csv', TRUE

EXEC InsertDataFromCSV 'WorkingHours', 'E:\CourseWork\csv\WorkingHours.csv', TRUE

EXEC InsertDataFromCSV 'DozeUnits', 'E:\CourseWork\csv\DozeUnits.csv', TRUE

EXEC InsertDataFromCSV 'MedicineUsingWays', 'E:\CourseWork\csv\MedicineUsingWays.csv', TRUE

EXEC InsertDataFromCSV 'Medicines', 'E:\CourseWork\csv\Medicines.csv', TRUE

EXEC InsertDataFromCSV 'MedicineSideEffects', 'E:\CourseWork\csv\MedicineSideEffects.csv', TRUE

EXEC InsertDataFromCSV 'Consultations', 'E:\CourseWork\csv\Consultations.csv', TRUE

EXEC InsertDataFromCSV 'Prescriptions', 'E:\CourseWork\csv\Prescriptions.csv', TRUE

EXEC InsertDataFromCSV 'Symptoms', 'E:\CourseWork\csv\Symptoms.csv', TRUE

EXEC InsertDataFromCSV 'ConsultationSymptoms', 'E:\CourseWork\csv\ConsultationSymptoms.csv', TRUE

EXEC InsertDataFromCSV 'Roles', 'E:\CourseWork\csv\Roles.csv', TRUE

EXEC InsertDataFromCSV 'Users', 'E:\CourseWork\csv\Users.csv', TRUE

# Приложение Г

Листинг создания триггеров:

USE [Healthcare]

GO

-- Триггер для обновления поля [UpdatedAt] при UPDATE у dbo.Doctors

DROP TRIGGER IF EXISTS trg\_UpdateDoctorTimestamp

GO

CREATE TRIGGER trg\_UpdateDoctorTimestamp

ON dbo.Doctors

AFTER UPDATE

AS

BEGIN

UPDATE dbo.Doctors

SET UpdatedAt = SYSDATETIME()

WHERE Id IN (SELECT DISTINCT Id FROM inserted)

END

GO

-- Триггер для обновления поля [UpdatedAt] при UPDATE у dbo.Patients

DROP TRIGGER IF EXISTS trg\_UpdatePatientTimestamp

GO

CREATE TRIGGER trg\_UpdatePatientTimestamp

ON dbo.Patients

AFTER UPDATE

AS

BEGIN

UPDATE dbo.Patients

SET UpdatedAt = SYSDATETIME()

WHERE Id IN (SELECT DISTINCT Id FROM inserted)

END

GO

-- Триггер для обновления поля [UpdatedAt] при UPDATE у dbo.Users

DROP TRIGGER IF EXISTS trg\_UpdateUserTimestamp

GO

CREATE TRIGGER trg\_UpdateUserTimestamp

ON dbo.Users

AFTER UPDATE

AS

BEGIN

UPDATE dbo.Users

SET UpdatedAt = SYSDATETIME()

WHERE Id IN (SELECT DISTINCT Id FROM inserted)

END

GO

-- Триггер для обновления поля [UpdatedAt] при UPDATE у dbo.WorkingHours

DROP TRIGGER IF EXISTS trg\_UpdateWorkingHourTimestamp

GO

CREATE TRIGGER trg\_UpdateWorkingHourTimestamp

ON dbo.WorkingHours

AFTER UPDATE

AS

BEGIN

UPDATE dbo.WorkingHours

SET UpdatedAt = SYSDATETIME()

WHERE Id IN (SELECT DISTINCT Id FROM inserted)

END

GO

-- Проверка, чтобы в Consultations StartTime был раньше, чем EndTime

DROP TRIGGER IF EXISTS trg\_Consultations\_InsertUpdate\_DateTime\_Check

GO

CREATE TRIGGER [dbo].[trg\_Consultations\_InsertUpdate\_DateTime\_Check]

ON [dbo].[Consultations]

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE EndTime <= StartTime

)

BEGIN

RAISERROR (N'Время начала консультации не должно быть больше или равно времени её окончания', 16, 1)

ROLLBACK TRAN

END

END

# Приложение Д

Листинг создания хранимой процедуры для импорта данных из CSV файлов в базу данных:

-- Хранимая процедура для выполнения импорта

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

DROP PROC InsertDataFromCSV

GO

CREATE PROC InsertDataFromCSV

@TableName NVARCHAR(128),

@AbsoluteFilePath NVARCHAR(500),

@FirstRowContainsHeaders BIT = FALSE,

@FieldTerminator NCHAR(1) = ',',

@RowTerminator NVARCHAR(4) = '0x0a',

@ShowSQL BIT = FALSE

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX)

BEGIN TRAN

SET @SQL = 'SET IDENTITY\_INSERT ' + QUOTENAME(@TableName) + ' ON'

EXEC sp\_executesql @SQL

-- Формируем команду BULK INSERT

SET @SQL = N'BULK INSERT ' + QUOTENAME(@TableName) +

N'FROM ' + QUOTENAME(@AbsoluteFilePath, '''') +

N'WITH (

CODEPAGE = 65001,

FIELDTERMINATOR = ''' + @FieldTerminator + ''',

ROWTERMINATOR = ''' + @RowTerminator + ''',

FIRSTROW = ' + IIF(@FirstRowContainsHeaders = 1, '2', '1') + '

)'

BEGIN TRY

IF @ShowSQL = 1 PRINT @SQL

EXEC sp\_executesql @SQL

COMMIT

PRINT 'Данные успешно вставлены.'

END TRY

BEGIN CATCH

PRINT 'Произошла ошибка: ' + ERROR\_MESSAGE()

ROLLBACK

END CATCH

-- Отключаем IDENTITY\_INSERT обратно

SET @SQL = 'SET IDENTITY\_INSERT ' + QUOTENAME(@TableName) + ' OFF'

EXEC sp\_executesql @SQL

END

GO