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 «ЗАПИСНАЯ КНИЖКА»**

Студент группы ИС-3 М.М. Шаров

Руководитель И.Н. Морошкин

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

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

Рыбинск, 2024

График выполнения работы

«База данных «Записная книжка»»

|  |  |  |
| --- | --- | --- |
| №№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 Шарову Матвей Михайловичу

1 Тема курсового проекта – База данных «Записная книжка»

Содержание:

1. Разработать БД ЗАПИСНАЯ КНИЖКА для оптовой базы. В каждый момент времени должны иметься точные данные о имени и фамилии контакта, знать отношение к контакту, иметь в пользовании адрес контакта, состоящий из региона, города и улицы с домом, хранить информацию о месте работы или учебы контакта с его должностью, и быть в курсе об дне рождения контакта чтобы поздравить его в этот день. Требуются следующие операции: писать поздравление о дне рождения контактам которые попадают в определенный диапазон дат, обновлять дату изменения контакта если было изменено имя или фамилия контакта, обновлять дату изменения контакта в случае обновления адреса контакта, а так же обновлять в случае изменения номера телефона контакта, просмотреть информацию о контакте с его данными состоящими из адреса контакта его номера телефона, и имени с фамилией, просмотреть информацию о контакте с его днем рождения, и местом работы или учебы с должностью.

Система должна обеспечивать поиск и выдачу сведений по разным вопросам.

1. Срок сдачи студентом 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 6](#_Toc185805214)

[1 Проектная часть 8](#_Toc185805215)

[1.1 Описание предметной области 8](#_Toc185805216)

[1.2 Анализ атрибутов 8](#_Toc185805217)

[1.3 Создание информационно-логической модели предметной области в каноническом виде 12](#_Toc185805218)

[1.4 Создание даталогической модели реляционной базы данных 14](#_Toc185805219)

[1.5 Разработка данных контрольного примера 15](#_Toc185805220)

[2 Программные разработки 19](#_Toc185805221)

[2.1 Создание базы данных 19](#_Toc185805222)

[2.2 Заполнение данными 20](#_Toc185805223)

[2.3 Разработка триггеров 21](#_Toc185805224)

[2.4 Разработка представлений 22](#_Toc185805225)

[2.5 Разработка хранимых процедур 25](#_Toc185805226)

[Заключение 30](#_Toc185805227)

[Список используемых источников 31](#_Toc185805228)

[Приложение А 32](#_Toc185805229)

[Приложение Б 34](#_Toc185805230)

[Приложение В 36](#_Toc185805231)

[Приложение Г 37](#_Toc185805232)

[Приложение Д 39](#_Toc185805233)

# Введение

Изначально компьютеры были задуманы для выполнения сложных математических задач, включая создание инновационных технологий. Однако с течением времени основное внимание переключилось на обработку и хранение информации. Это логичное изменение приоритетов: гражданский сектор превзошел военные и научные потребности, и снижение стоимости компьютеров сделало их доступными даже для небольших компаний и частных лиц. Сегодня уже невозможно представить управление предприятием без компьютеров. Они стали неотъемлемой частью бухгалтерии, складского учета, и закупок. Однако современный бизнес требует более широкого использования информационных технологий для эффективного управления. Успех бизнеса сегодня зависит от точности управленческих решений. Интуиция и личный опыт руководителя уже недостаточны для лидерства. Чем крупнее компания, тем серьезнее нужно вкладываться в эти технологии. Они становятся жизненно важными в условиях жесткой конкуренции, и лишь компании с лучшей оснащенностью и эффективной организацией смогут достичь успеха.

Автоматизированная информационная система «Записная книжка» включает в себя данные о контакте, номере телефона, адресе и месте работы с датой рождения. База данных позволяет осуществлять добавление, изменение, поиск и удаление данных, а также просматривать эти данные. Актуальность данной темы в том, что в наш век информационных технологий, стало обычным делом обменятся контактами с любым человеков, и удобно структурировать информацию о новом контакте, такую как адрес проживания, место работы и т.д.

Целью курсового проекта является разработка базы данных «Записная книжка», в которой будет храниться информация о контакте, адресе, место работы, номер телефона и характер знакомства с контактом.

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 «Записная книжка».

# 1 Проектная часть

## 1.1 Описание предметной области

Концепция информационной системы для записной книжки представляет собой комплексный инструмент, позволяющий эффективно просматривать и изменять сведения о контакте.

Эта система предназначена для учета и управления данными о контактах, их наличии на в книжке, а также для оптимизации и облегчения процесса редактирования данных о контактах. Каждый контакт имеет свои уникальные характеристики, включая адрес проживания контакта, место учебы или работы представленного контакта. Основная задача информационной системы - обеспечить возможность добавления новых контактов с данными о них, корректировки или удаления не нужной информации. Она также обеспечивает учет актуальных данных о контакте в записной книжке. Важным аспектом работы системы является обеспечение актуальности данных и возможность оперативно реагировать на запросы пользователя по внесению данных о контактах.

## 1.2 Анализ атрибутов

Проектирование базы данных начинается с создания все покрывающего 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.

Для проектирования базы данных «Записная книжка» необходимо определить объекты, сведения о которых будут включены в БД.

База данных должна хранить следующую информацию: контакты, дни рождения, отношения контактов, телефоны, место работы или учебы, и адрес с городом и регион в отдельности.

Установим атрибуты, которые должны содержаться в БД, перечислим имена выделенных атрибутов и их краткие характеристики:

- контакт – имя контакта, дата изменения данных о нем;

- день рождения – хранит дни рождения контактов;

- отношения – информация кем приходится нам контакт;

- телефон – номер телефона контакта;

- место работы или учебы – информация о должности в этом месте;

- адрес – код района с названием улицы проживания связывается с городом и регионом;

- город – название города;

- регион – название региона.

В результате приведения исходного отношения к НФБК получаем следующие сущности: контакты, день рождения, отношения, телефоны, место работы или учебы, адреса, города, регионы.

Сущность «Контакты» предназначена для хранения данных о контакте. Атрибуты данной таблицы представлены в таблице 1.1

Таблица 1.1 – Атрибуты сущности «Контакты»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Контакт Код | INT | Идентификатор контакта |
| Имя | NVARCHAR(50) | Имя контакта |
| Фамилия | NVARCHAR(50) | Фамилия контакта |
| Когда изменён | DATATIME | Время последнего изменения контакта |

Сущность «День рождения» предназначена для хранения информации о днях рождения. Атрибуты данной таблицы представлены в таблице 1.2

Таблица 1.2 – Атрибуты сущности «День рождения»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| День Рождения Код | INT | Идентификатор день рождения |
| Контакт Код | INT | Идентификатор контакта |
| День рождения дата | DATE | Дата дня рождения |

Сущность «Отношения» предназначена для хранения информации о отношении к контакту. Атрибуты данной таблицы представлены в таблице 1.3

Таблица 1.3 – Атрибуты сущности «Отношения»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Отношения Код | INT | Идентификатор отношения |
| Контакт Код | INT | Идентификатор контакта |
| Тип отношения | NVARCHAR(50) | Характер отношения |

Сущность «Телефоны» предназначена для хранения данных о номерах телефона. Атрибуты данной таблицы представлены в таблице 1.4

Таблица 1.4 – Атрибуты сущности «Телефоны»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Телефон Код | INT | Идентификатор телефона |
| Контакт Код | INT | Идентификатор контакта |
| Номер телефона | NVARCHAR(20) | Номер телефона контакта |

Сущность «Место работы или учебы» предназначена для хранения данных о местах работы. Атрибуты данной таблицы представлены в таблице 1.5

Таблица 1.5 – Атрибуты сущности «Место работы или учебы»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Место работы или учебы Код | INT | Идентификатор телефона |
| Контакт Код | INT | Идентификатор контакта |
| Название организации | NVARCHAR(100) | Название организации контакта |
| Должность | NVARCHAR(50) | Должность в организации |

Сущность «Адреса» предназначена для хранения данных о адресах. Атрибуты данной таблицы представлены в таблице 1.6

Таблица 1.6 – Атрибуты сущности «Адреса»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Адрес Код | INT | Идентификатор адреса |
| Контакт Код | INT | Идентификатор контакта |
| Улица | NVARCHAR(100) | Название улицы и дома |
| Город Код | INT | Идентификатор города |
| Почтовый индекс | NVARCHAR(20) | Номер почтового индекса |

Сущность «Города» предназначена для хранения данных о городах. Атрибуты данной таблицы представлены в таблице 1.7

Таблица 1.7 – Атрибуты сущности «Города»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Города Код | INT | Идентификатор города |
| Города | NVARCHAR(50) | Название города |
| Регион Код | INT | Идентификатор региона |

Сущность «Регионы» предназначена для хранения данных о регионе. Атрибуты данной таблицы представлены в таблице 1.8

Таблица 1.8 – Атрибуты сущности «Регионы»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Характеристика |
| Регион Код | INT | Идентификатор города |
| Регион | NVARCHAR(50) | Название региона |

## 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 «Записная книжка». На рисунке 1.1 изображена инфологическая модель базы данных «Записная книжка» методом IDEF1X.

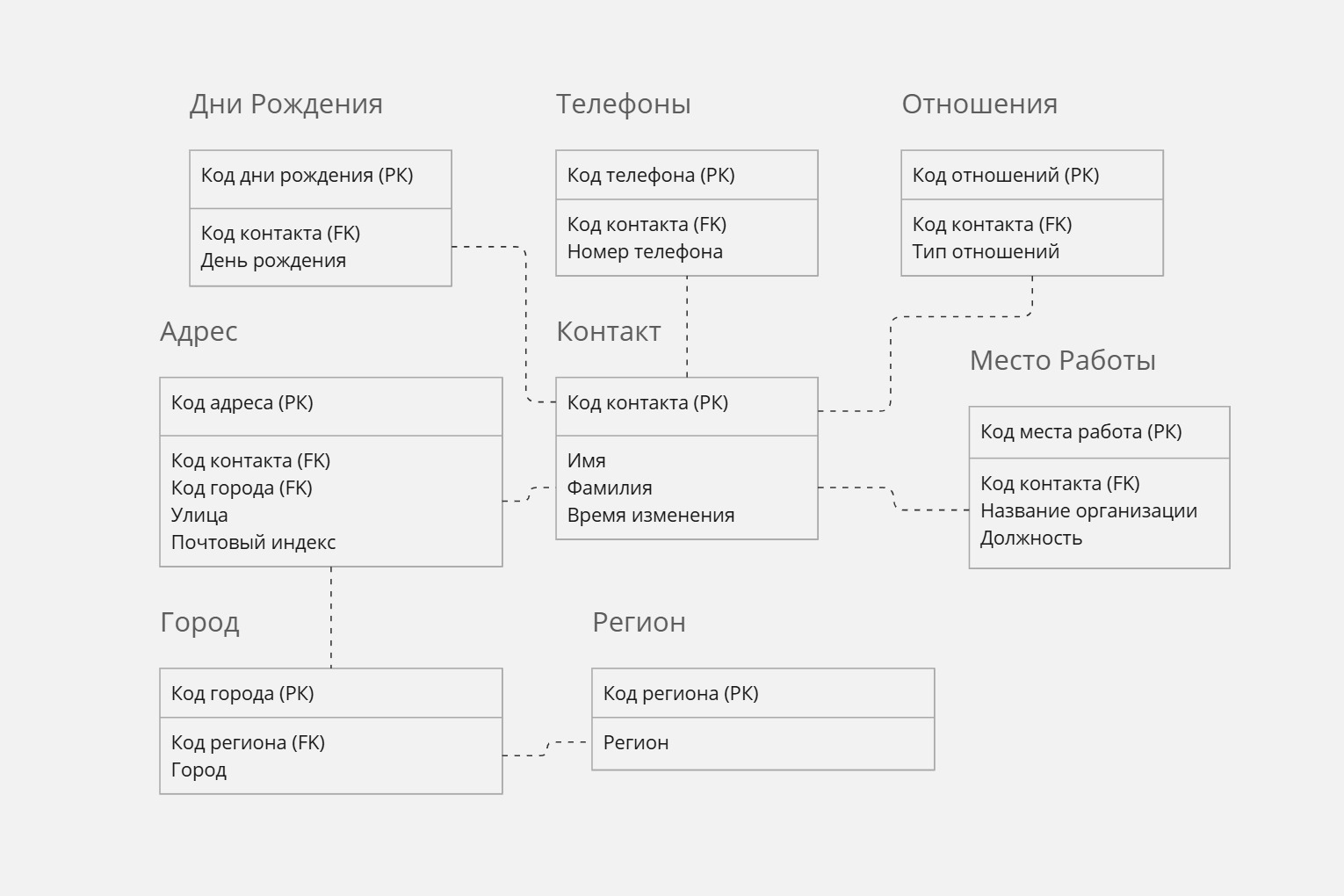


Рисунок 1.1 – Инфологическая модель базы данных «Записная книжка»

## 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 «Записная книжка».

## 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 список данных для заполнения базы данных информацией.

Контрольные данные для сущности «Контакты» представлены в таблице 1.12.

Таблица 1.12 – Контрольные данные примера для сущности «Контакты»

|  |  |  |  |
| --- | --- | --- | --- |
| Код контакта | Имя | Фамилия | Дата Изменения |
| 1 | Иван | Петров | 2024-12-22 16:56:22.850 |
| 2 | Сергей | Скворцов | 2024-12-22 15:45:44.520 |
| 3 | Инга | Платонова | 2024-12-22 15:45:44.520 |
| 4 | Мария | Аремова | 2024-12-22 15:45:44.520 |
| 5 | Иван | Гончаров | 2024-12-22 17:08:08.773 |
| 6 | Лена | Завещук | 2024-12-22 15:45:44.520 |
| 7 | Олеся | Верещук | 2024-12-22 15:45:44.520 |
| 8 | Роман | Лобачев | 2024-12-22 15:45:44.520 |

Окончание таблицы 1.12

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Кирилл | Букачев | 2024-12-22 18:23:22.840 |
| 10 | Александра | Никифова | 2024-12-22 15:45:44.520 |

Контрольные данные для сущности «Телефоны» представлены в таблице 1.13.

Таблица 1.13 – Контрольные данные примера для сущности «Телефоны»

|  |  |  |
| --- | --- | --- |
| Код телефона | Код контакта | Номер телефона |
| 1 | 1 | 79159923014 |
| 2 | 2 | 79057922014 |
| 3 | 3 | 79156528719 |
| 4 | 4 | 79159356709 |
| 5 | 5 | 79156661234 |
| 6 | 6 | 79096826017 |
| 7 | 7 | 79117982341 |
| 8 | 8 | 79150913718 |
| 9 | 9 | 79153923311 |
| 10 | 10 | 79152428375 |

Контрольные данные для сущности «Дни рождения» представлены в таблице 1.14.

Таблица 1.14 – Контрольные данные примера для сущности «Дни рождения»

|  |  |  |
| --- | --- | --- |
| Код дней рождения | Код контакта | Дата дня рождения |
| 1 | 1 | 2000-01-09 |
| 2 | 2 | 1999-04-03 |
| 3 | 3 | 2001-08-15 |
| 4 | 4 | 2004-02-23 |
| 5 | 5 | 2004-01-11 |
| 6 | 6 | 2003-09-19 |
| 7 | 7 | 2003-10-17 |
| 8 | 8 | 2002-11-24 |
| 9 | 9 | 2002-12-29 |
| 10 | 10 | 2003-03-04 |

Контрольные данные для сущности «Место работы» представлены в таблице 1.15.

Таблица 1.15 – Контрольные данные примера для сущности «Место работы»

|  |  |  |  |
| --- | --- | --- | --- |
| Код место работы | Код контакта | Название организации | Должность |
| 1 | 1 | ООО НПО Криста | Программист |
| 2 | 2 | ООО НПО Криста | Программист |
| 3 | 3 | МУ МВД России | Оперативник |
| 4 | 4 | МУ МВД России | Оперативник |
| 5 | 5 | ПАО «ОДК-САТУРН» | Инженер |
| 6 | 6 | Рыбинский авиационный колледж | Студентка |
| 7 | 7 | Рыбинский авиационный колледж | Студентка |
| 8 | 8 | Рыбинский авиационный колледж | Студент |
| 9 | 9 | Рыбинский авиационный колледж | Студент |
| 10 | 10 | ПАО «ОДК-САТУРН» | Бухгалтер |

Контрольные данные для сущности «Отношения» представлены в таблице 1.16.

Таблица 1.16 – Контрольные данные примера для сущности «Отношения»

|  |  |  |
| --- | --- | --- |
| Код отношения | Код контакта | Тип отношения |
| 1 | 1 | Брат |
| 2 | 2 | Друг |
| 3 | 3 | Подруга |
| 4 | 4 | Подруга |
| 5 | 5 | Друг |
| 6 | 6 | Однокурсница |
| 7 | 7 | Однокурсница |
| 8 | 8 | Однокурсник |
| 9 | 9 | Однокурсник |
| 10 | 10 | Подруга |

Контрольные данные для сущности «Адреса» представлены в таблице 1.17.

Таблица 1.17 – Контрольные данные примера для сущности «Адреса»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код Адрес | Код контакта | Улица | Код Города | Почтовый индекс |
| 1 | 1 | д. 12, улица 50 Лет ВЛКСМ | 1 | 152925 |
| 2 | 2 | д.3, улица Айвазовского | 2 | 156678 |
| 3 | 3 | д. 1, улица Боткина | 1 | 152939 |
| 4 | 4 | д. 25, улица Герцена | 1 | 152645 |
| 5 | 5 | д. 36, улица Блюхера | 2 | 156547 |
| 6 | 6 | д. 31, улица Гоголя | 3 | 176123 |
| 7 | 7 | д. 7, улица Глазурная | 3 | 176275 |
| 8 | 8 | д. 15 а, улица Зои Космодемьянской | 3 | 176076 |

Окончание таблицы 1.17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | 9 | д. 21, улица Карла Маркса | 1 | 152783 |
| 10 | 10 | д. 8, улица 50 Лет ВЛКСМ | 1 | 152928 |

Контрольные данные для сущности «Город» представлены в таблице 1.18.

Таблица 1.18 – Контрольные данные примера для сущности «Город»

|  |  |  |
| --- | --- | --- |
| Код города | Города | Код региона |
| 1 | Москва | 2 |
| 2 | Рыбинск | 1 |
| 3 | Ярославль | 1 |

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

Таблица 1.19 – Контрольные данные примера для сущности «Регион»

|  |  |
| --- | --- |
| Код региона | Регион |
| 1 | Ярославская область |
| 2 | Московская область |

# 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 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 «Контакты» в Microsoft SQL Server



Рисунок 2.1- Таблица «Контакты»

Аналогичные действия проводятся для создания всех таблиц БД «Записная книжка»

На рисунке 2.2 изображена диаграмма базы данных БД «Записная книжка».

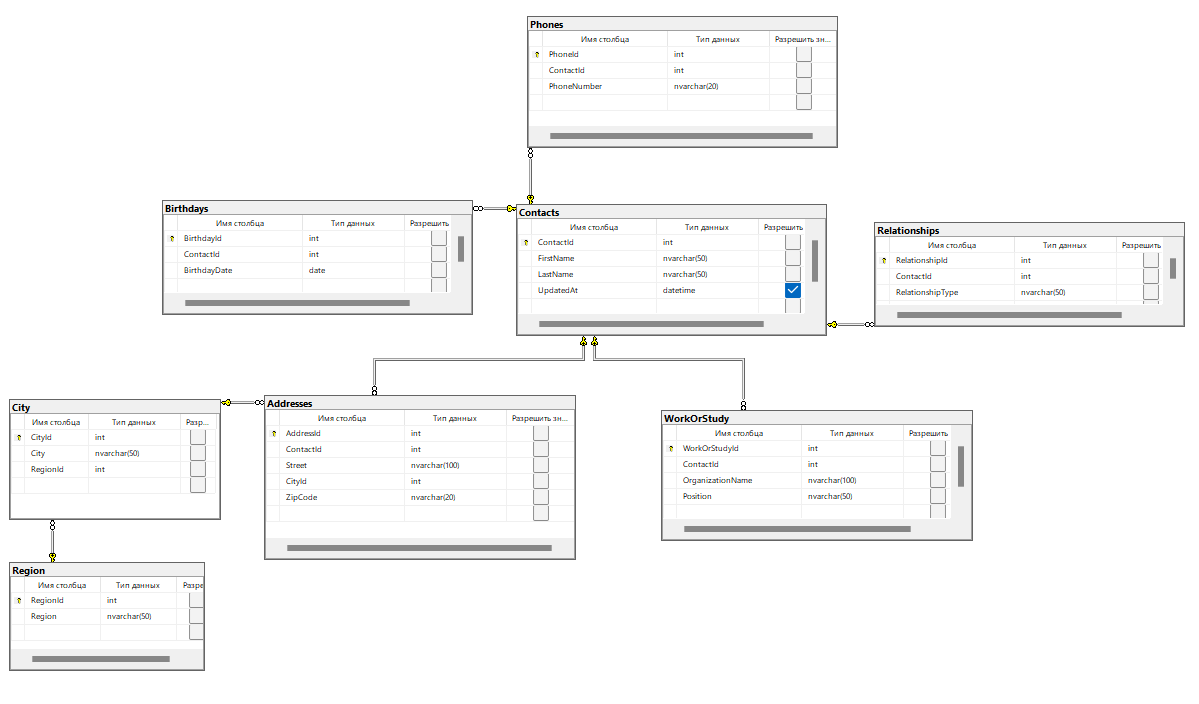


Рисунок 2.2 – Диаграмма базы данных «Записная книжка»

## 2.2 Заполнение данными

Созданные таблицы необходимо заполнить данными, которые мы записали в таблицах выше.

Для заполнения мы используем INSERT.

На рисунке 2.3 изображена таблица «Контакты» через INSERT.

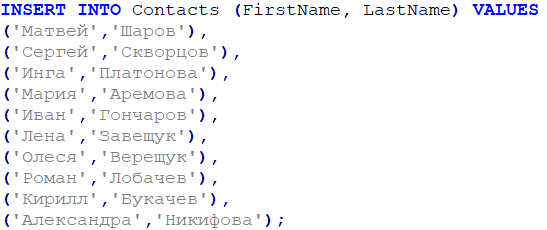


Рисунок 2.3 – Таблица «Контакты» через INSERT

Заполним данными основную таблицу при помощи команды INSERT INTO.

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

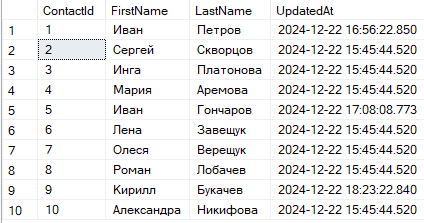


Рисунок 2.4 – Заполненная таблица «Контакты»

Аналогичные действия проводятся для всех таблиц БД «Записная книжка»

Листинг импорта данных в базу данных «Записная книжка» 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 AFTER 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 «Записная книжка» представлен в приложении В.

## 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 БД записной книжки необходимо создать несколько представлений.

С помощью инструкции CREATE VIEW нужно создать представление vw\_ContactDetails. Далее представлен код для создания данного представления:

CREATE VIEW vw\_ContactDetails AS

SELECT

c.ContactId,

c.FirstName,

c.LastName,

a.Street,

ci.City,

r.Region,

a.ZipCode,

p.PhoneNumber

FROM

Contacts c

LEFT JOIN

Addresses a ON c.ContactId = a.ContactId

LEFT JOIN

City ci ON a.CityId = ci.CityId

LEFT JOIN

Region r ON ci.RegionId = r.RegionId

LEFT JOIN

Phones p ON c.ContactId = p.ContactId;

После этого в окне результатов появится таблица представления, это изображено на рисунке 2.5.

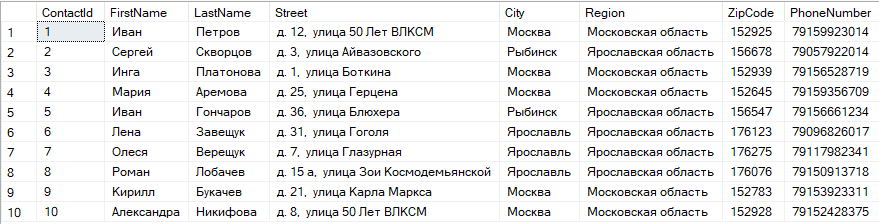


Рисунок 2.5 – представление для вывода подробностей как зовут нашего контакта с его адресом и телефоном

Так же показывать в каких отношениях с контактом мы состоим для правильного обращения. Для этого создадим представление vw\_ContactRelationships. Далее представлен код для создания данного представления:

CREATE VIEW vw\_ContactRelationships AS

SELECT

c.ContactId,

c.FirstName,

c.LastName,

r.RelationshipType

FROM

Contacts c

LEFT JOIN

Relationships r ON c.ContactId = r.ContactId;

После создания представления нужно обратиться к нему инструкцией SELECT:

USE AddressBook

SELECT \* FROM vw\_ContactRelationships;

Далее в окне результатов появится таблица представления, это показано на рисунке 2.6.



Рисунок 2.6 – представление для вывода данных об отношениях контактов

## 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 Transact-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 нужно создать хранимую процедуру GetBirthdayGreetings, которая поздравляет контакты в выбранном диапазоне. Далее представлен код создания данной процедуры:

CREATE PROCEDURE GetBirthdayGreetings

@StartDate DATE = NULL, -- Начальная дата диапазона (опционально)

@EndDate DATE = NULL -- Конечная дата диапазона (опционально)

AS

BEGIN

SET NOCOUNT ON;

-- Проверяем, если задан только один параметр, то считаем диапазон от этой даты до самой даты

IF @StartDate IS NOT NULL AND @EndDate IS NULL

BEGIN

SET @EndDate = @StartDate;

END

-- Если ни одна дата не указана, то будем использовать текущую дату

IF @StartDate IS NULL AND @EndDate IS NULL

BEGIN

SET @StartDate = GETDATE();

SET @EndDate = GETDATE();

END

-- Формируем поздравления, сравнивая только день и месяц

SELECT

c.FirstName,

c.LastName,

b.BirthdayDate,

CONCAT('Поздравляем ', c.FirstName, ' ', c.LastName,

' с днем рождения!') AS BirthdayGreeting

FROM

Birthdays b

JOIN

Contacts c ON b.ContactId = c.ContactId

WHERE

DATEPART(DAY, b.BirthdayDate) BETWEEN DATEPART(DAY, @StartDate) AND DATEPART(DAY, @EndDate)

AND DATEPART(MONTH, b.BirthdayDate) BETWEEN DATEPART(MONTH, @StartDate) AND DATEPART(MONTH, @EndDate)

ORDER BY

DATEPART(MONTH, b.BirthdayDate), DATEPART(DAY, b.BirthdayDate);

END;

С помощью инструкции CREATE PROCEDURE нужно создать хранимую процедуру UpdateContactName, которая с изменением имени и фамилии вносит новую дату с изменения. Листинг создания данной процедуры представлен в приложении Г.

После удачного создания хранимых процедур они появятся в одноименном пункте MS SQL (см. рисунок 2.8)

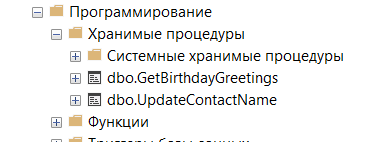


Рисунок 2.8 – созданные хранимые процедуры в MS SQL

В результате программных разработок была создана БД записная книжка, ее таблицы. Таблицы заполнены корректными данными. Также разработаны триггера, представления и хранимые процедуры.

# Заключение

В ходе выполнения курсовой работы была разработана база данных «Снабжение магазинов» на языке Transact-SQL.

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 оптовыми базами для учета, хранения и вывода различной информации.

В настоящее время хранение данных о контактах просто необходимы для понимания информации о контакте и поэтому всегда существует необходимость создания новых элементов и операций, поэтому в базу данных могут быть внесены изменения.

# 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. Смирнов И. В. Реляционные базы данных: теория и практика.
  8. Козлов А. Д. Основы работы с базами данных: практическое руководство.

# Приложение А

Листинг создания таблиц для базы данных:

USE AddressBook;

-- Создаем таблицу для хранения контактов

CREATE TABLE Contacts (

ContactId INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

UpdatedAt DATETIME DEFAULT GETDATE()

);

-- Создаем таблицу для хранения адресов

CREATE TABLE Addresses (

AddressId INT PRIMARY KEY IDENTITY(1,1),

ContactId INT NOT NULL,

Street NVARCHAR(100) NOT NULL,

CityId INT NOT NULL,

ZipCode NVARCHAR(20) NOT NULL,

FOREIGN KEY (ContactId) REFERENCES Contacts(ContactId)

);

-- Создаем таблицу для хранения регионов

CREATE TABLE Region (

RegionId INT PRIMARY KEY IDENTITY(1,1),

Region NVARCHAR(50) NOT NULL,

FOREIGN KEY (RegionId) REFERENCES City(RegionId)

);

-- Создаем таблицу для хранения городов

CREATE TABLE City (

CityId INT PRIMARY KEY IDENTITY(1,1),

City NVARCHAR(50) NOT NULL,

RegionId INT NOT NULL,

FOREIGN KEY (CityId) REFERENCES Addresses(CityId)

);

-- Создаем таблицу для хранения телефонных номеров

CREATE TABLE Phones (

PhoneId INT PRIMARY KEY IDENTITY(1,1),

ContactId INT NOT NULL,

PhoneNumber NVARCHAR(20) NOT NULL,

FOREIGN KEY (ContactId) REFERENCES Contacts(ContactId)

);

-- Создаем таблицу для хранения информации о характере знакомства или родства

CREATE TABLE Relationships (

RelationshipId INT PRIMARY KEY IDENTITY(1,1),

ContactId INT NOT NULL,

RelationshipType NVARCHAR(50) NOT NULL, -- Например, "Друг", "Коллега", "Родственник"

FOREIGN KEY (ContactId) REFERENCES Contacts(ContactId)

);

-- Создаем таблицу для хранения информации о местах работы или учебы

CREATE TABLE WorkOrStudy (

WorkOrStudyId INT PRIMARY KEY IDENTITY(1,1),

ContactId INT NOT NULL,

OrganizationName NVARCHAR(100) NOT NULL,

Position NVARCHAR(50) NOT NULL, -- Например, "Менеджер", "Студент"

FOREIGN KEY (ContactId) REFERENCES Contacts(ContactId)

);

-- Таблица для хранения дней рождения

CREATE TABLE Birthdays (

BirthdayId INT PRIMARY KEY IDENTITY(1,1),

ContactId INT NOT NULL,

BirthdayDate DATE NOT NULL,

FOREIGN KEY (ContactId) REFERENCES Contacts(ContactId)

);

# Приложение Б

Листинг для заполнения через INSERT в базу данных:

USE AddressBook;

GO

-- Вставляем данные в таблицу Region

INSERT INTO Region (Region) VALUES

('Ярославская область'),

('Московская область');

-- Вставляем данные в таблицу City

INSERT INTO City (City, RegionId) VALUES

('Москва', 2),

('Рыбинск', 1),

('Ярославль', 1);

-- Вставляем данные в таблицу Contacts

INSERT INTO Contacts (FirstName, LastName) VALUES

('Матвей','Шаров'),

('Сергей','Скворцов'),

('Инга','Платонова'),

('Мария','Аремова'),

('Иван','Гончаров'),

('Лена','Завещук'),

('Олеся','Верещук'),

('Роман','Лобачев'),

('Кирилл','Букачев'),

('Александра','Никифова');

-- Вставляем данные в таблицу Addresses

INSERT INTO Addresses (ContactId, Street, CityId, ZipCode) VALUES

(1, 'д. 12, улица 50 Лет ВЛКСМ', 1, '152925'),

(2, 'д. 3, улица Айвазовского', 2, '156678'),

(3, 'д. 1, улица Боткина', 1, '152939'),

(4, 'д. 25, улица Герцена', 1, '152645'),

(5, 'д. 5, улица Блюхера', 2, '156547'),

(6, 'д. 31, улица Гоголя', 3, '176123'),

(7, 'д. 7, улица Глазурная', 3, '176275'),

(8, 'д. 15 а, улица Зои Космодемьянской', 3, '176076'),

(9, 'д. 11, улица Карла Маркса', 1, '152783'),

(10, 'д. 8, улица 50 Лет ВЛКСМ', 1, '152928');

-- Вставляем данные в таблицу Phones

INSERT INTO Phones (ContactId, PhoneNumber) VALUES

(1, '79159923014'),

(2, '79057922014'),

(3, '79156528719'),

(4, '79159356709'),

(5, '79156661234'),

(6, '79096826017'),

(7, '79117982341'),

(8, '79150913718'),

(9, '79053923311'),

(10, '79152428375');

-- Вставляем данные в таблицу Relationships

INSERT INTO Relationships (ContactId, RelationshipType) VALUES

(1, 'Брат'),

(2, 'Друг'),

(3, 'Подруга'),

(4, 'Подруга'),

(5, 'Друг'),

(6, 'Однокурсница'),

(7, 'Однокурсница'),

(8, 'Однокурсник'),

(9, 'Однокурсник'),

(10, 'Подруга');

-- Вставляем данные в таблицу WorkOrStudy

INSERT INTO WorkOrStudy (ContactId, OrganizationName, Position) VALUES

(1, 'ООО НПО Криста', 'Программист'),

(2, 'ООО НПО Криста', 'Программист'),

(3, 'МУ МВД России', 'Оперативник'),

(4, 'МУ МВД России', 'Оперативник'),

(5, 'ПАО «ОДК-САТУРН»', 'Инженер'),

(6, 'Рыбинский авиационный колледж', 'Студентка'),

(7, 'Рыбинский авиационный колледж', 'Студентка'),

(8, 'Рыбинский авиационный колледж', 'Студент'),

(9, 'Рыбинский авиационный колледж', 'Студент'),

(10, 'ПАО «ОДК-САТУРН»', 'Бухгалтер');

-- Вставляем данные в таблицу Birthdays

INSERT INTO Birthdays (ContactId, BirthdayDate) VALUES

(1, '2000-01-09'),

(2, '2003-03-04'),

(3, '1999-04-03'),

(4, '2001-08-15'),

(5, '2004-02-23'),

(6, '2004-01-11'),

(7, '2003-09-19'),

(8, '2003-10-17'),

(9, '2002-11-24'),

(10, '2002-12-29');

# Приложение В

Листинг создания триггеров:

-- Триггер для обновления поля UpdatedAt в таблице Contacts при изменении данных в таблице Phones

CREATE TRIGGER trg\_UpdateContactWhenPhoneChanged

ON Phones

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Обновляем поле UpdatedAt в таблице Contacts для каждого обновленного телефонного номера

UPDATE c

SET c.UpdatedAt = GETDATE()

FROM Contacts c

INNER JOIN inserted i ON c.ContactId = i.ContactId;

PRINT 'Дата изменения контакта обновлена после изменения телефонного номера.';

END;

-- Триггер для обновления поля UpdatedAt в таблице Contacts при изменении данных в таблице Addresses

CREATE TRIGGER trg\_UpdateContactWhenAddressChanged

ON Addresses

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Обновляем поле UpdatedAt в таблице Contacts для каждого обновленного адреса

UPDATE c

SET c.UpdatedAt = GETDATE()

FROM Contacts c

INNER JOIN inserted i ON c.ContactId = i.ContactId;

PRINT 'Дата изменения контакта обновлена после изменения адреса.';

END;

# Приложение Г

Листинг создания хранимых процедур:

CREATE PROCEDURE GetBirthdayGreetings

@StartDate DATE = NULL, -- Начальная дата диапазона (опционально)

@EndDate DATE = NULL -- Конечная дата диапазона (опционально)

AS

BEGIN

SET NOCOUNT ON;

-- Проверяем, если задан только один параметр, то считаем диапазон от этой даты до самой даты

IF @StartDate IS NOT NULL AND @EndDate IS NULL

BEGIN

SET @EndDate = @StartDate;

END

-- Если ни одна дата не указана, то будем использовать текущую дату

IF @StartDate IS NULL AND @EndDate IS NULL

BEGIN

SET @StartDate = GETDATE();

SET @EndDate = GETDATE();

END

-- Формируем поздравления, сравнивая только день и месяц

SELECT

c.FirstName,

c.LastName,

b.BirthdayDate,

CONCAT('Поздравляем ', c.FirstName, ' ', c.LastName,

' с днем рождения!') AS BirthdayGreeting

FROM

Birthdays b

JOIN

Contacts c ON b.ContactId = c.ContactId

WHERE

DATEPART(DAY, b.BirthdayDate) BETWEEN DATEPART(DAY, @StartDate) AND DATEPART(DAY, @EndDate)

AND DATEPART(MONTH, b.BirthdayDate) BETWEEN DATEPART(MONTH, @StartDate) AND DATEPART(MONTH, @EndDate)

ORDER BY

DATEPART(MONTH, b.BirthdayDate), DATEPART(DAY, b.BirthdayDate);

END;

-- Создание процедуры для обновления имени и фамилии контакта с обновлением даты изменения данных

CREATE PROCEDURE UpdateContactName

@ContactId INT, -- Идентификатор контакта, чьи данные нужно обновить

@NewFirstName NVARCHAR(50), -- Новое имя контакта

@NewLastName NVARCHAR(50) -- Новая фамилия контакта

AS

BEGIN

SET NOCOUNT ON;

-- Проверяем, изменяются ли имя или фамилия

IF EXISTS (SELECT 1 FROM Contacts WHERE ContactId = @ContactId AND

(FirstName <> @NewFirstName OR LastName <> @NewLastName))

BEGIN

-- Обновляем имя, фамилию и дату изменения

UPDATE Contacts

SET FirstName = @NewFirstName,

LastName = @NewLastName,

UpdatedAt = GETDATE() -- Обновляем дату изменения данных

WHERE ContactId = @ContactId;

-- Возвращаем сообщение об успешном обновлении

PRINT 'Контакт успешно обновлен.';

END

ELSE

BEGIN

-- Если имя и фамилия не изменились, выводим сообщение

PRINT 'Имя и фамилия не изменились, обновление не требуется.';

END

END;

# Приложение Д

Листинг создания представлений:

CREATE VIEW vw\_ContactDetails AS

SELECT

c.ContactId,

c.FirstName,

c.LastName,

a.Street,

ci.City,

r.Region,

a.ZipCode,

p.PhoneNumber

FROM

Contacts c

LEFT JOIN

Addresses a ON c.ContactId = a.ContactId

LEFT JOIN

City ci ON a.CityId = ci.CityId

LEFT JOIN

Region r ON ci.RegionId = r.RegionId

LEFT JOIN

Phones p ON c.ContactId = p.ContactId;

CREATE VIEW vw\_ContactRelationships AS

SELECT

c.ContactId,

c.FirstName,

c.LastName,

r.RelationshipType

FROM

Contacts c

LEFT JOIN

Relationships r ON c.ContactId = r.ContactId;

CREATE VIEW vw\_ContactBirthdaysAndWork AS

SELECT

c.ContactId,

c.FirstName,

c.LastName,

b.BirthdayDate,

w.OrganizationName,

w.Position

FROM

Contacts c

LEFT JOIN

Birthdays b ON c.ContactId = b.ContactId

LEFT JOIN

WorkOrStudy w ON c.ContactId = w.ContactId;