МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ «КОМПЬЮТЕРНЫЙ ИНЖИНИРИНГ»

КАФЕДРА ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

У Т В Е Р Ж Д А Ю
Зав. кафедрой
_____
«____» _____ 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему **«Создание интерактивной информационной системы для учета движения и поставок сырья (сельхоз продукции)»**

Выполнил: Садыков Б.Т.

Научный руководитель: Айтмуратов Б.Ш.

НУКУС - 2017 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последнее время нашей жизни сложно представить жизнь современного человека без новейших технологий, таких как мобильные телефоны, планшеты, персональные компьютеры, разные гаджеты и т.д. Эти технологии проникли уже во все сферы торговли, образовании и производства. До развития компьютеров, когда не было электронное вычислительных машин, приходилось работать с огромными бумажными документами, и это было очень трудоемкое дело, особенно в производственных сферах и торговле, так как поиск и обработка этих документов уходили довольно большое количество времени. Но с развитием технологий прогресс не стоит на одном месте. Появились новые программные средства, системы управления базами данных, такие как Microsoft Access, Microsoft SQL Server, 1С:Предприятие, Oracle Database которые обеспечивают ввод, обработки, хранения, изменения данных и позволяет сэкономить достаточно большое количество времени.

В современном мире сейчас практически во всех организациях используются компьютеры и для хранения и обработки различных информации выше перечисленные системы управлениями базами данных. Эти информации хранятся в базах данных. Базы данных играют особую роль в современном мире. Умение работать с ними является одним из важнейших навыков в работе с компьютером, а специалисты в этой области всегда востребованы.

База данных (БД) – это хранилище для большого количества систематизированных данных, с которыми можно производить определённые действия: добавления, удаления, изменения, копирования, упорядочивание. Эти базы данных образовываются и функционируют под управлением специальных программных комплексов (совокупностей языков

программирования и программных средств), называемых системами управления базами данных (СУБД). По существу, система управления базами данных служит посредником между пользователями и базой данных.

За последние время наблюдаются тенденция к усложнению структур данных. Простые виды информации, представимой в форме чисел и текстовых строк, дополняются многочисленными мультимедийными документами, графическими образами, хронологическими рядами и множеством прочих сложных информационных форм. В связи с этим появилось множество весьма изощренные системы управления базамы данных, которые поддерживают новые коллекции данных и способных реализовать преимущества современных аппаратных средств.

В данной работе представляется некоторые теоретические аспекты теории базы данных, основные понятия, функциональные возможности систем управления базами данных.

Целью выпускной квалификационной работы является изучение проектирование базы данных для учета движения и поставок сырья в Microsoft SQL Server.

Для достижения цели используются следующие входные данные:

- информация о товарах,
- информация о заказах,
- информация о поставщиках,
- информация о сотрудниках,
- информация об складах.

Выходными данными являются запросы. Информация выводится на экран в специальных формах, упрощающих работу с записями таблиц БД.

Выходные данные:

1) заполнение и редактирование таблиц базы данных;

2) вывод сведений о товарах, поставщиках, покупателях;

3) вывод сведений о поставленных товарах и назначенному пункту для каждого покупателя;

Поставленная задача создания базы данных для учета сельскохозяйственной продукты является весьма актуальной на данное время. С помощью этой программы можно упростить работу фермеров – это легкий способ поиска и редактирования данных о товарах, покупателях.

ГЛАВА 1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Основы баз данных

Система управления базами данных (СУБД) – это не что иное, как компьютеризованная система хранения записей. Саму же базу данных можно рассматривать как подобие электронной картотеки, т.е. хранилище или контейнер для некоторого набора занесенных в компьютер файлов данных.

К основным функциям СУБД принято относить следующие:

1) управление данными во внешней памяти;

2) управление буферами оперативной памяти;

3) управление транзакциями;

4) журнализация и восстановление БД после сбоев;

5) поддержка языков БД.

Управление данными во внешней памяти включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в базу данных, так и для служебных целей, например, для ускорения доступа к данным.

Управление буферами оперативной памяти. СУБД, как правило, работают с БД большого объема. По крайней мере, объем базы данных существенно превышает объем оперативной памяти. Так что, если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости

Управление транзакциями. Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция либо успешно выполняется, и СУБД фиксирует произведенные изменения данных во внешней памяти, либо ни одно из этих изменений никак не

отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД, поэтому поддержание механизма транзакций является обязательным условием как однопользовательских, так и многопользовательских СУБД.

Журнализация и восстановление БД после сбоя. Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее целостное состояние БД после любого аппаратного или программного сбоя.

Поддержка языков БД. Для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка - язык определения схемы БД (SDL - Schema Definition Language) и язык манипулирования данными (DML - Data Manipulation Language). SDL служил, главным образом, для определения логической структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, позволяющих вводить, удалять, модифицировать и выбирать данные. В современных СУБД, обычно, поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД и обеспечивающий базовый пользовательский интерфейс. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language) [2].

Язык SQL содержит специальные средства определения ограничений целостности БД. Ограничения целостности хранятся в специальных таблицах-каталогах. Обеспечение контроля целостности производится на языковом уровне. При компиляции операторов модификации БД, компилятор SQL, на основании имеющихся ограничений целостности, генерирует соответствующий программный код.

Специальные операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми запросами. Для пользователя представление является такой же таблицей, как любая базовая таблица, хранимая в БД, но с его помощью можно ограничить или расширить видимость БД для конкретного пользователя. Поддержание представлений производится также на языковом уровне.

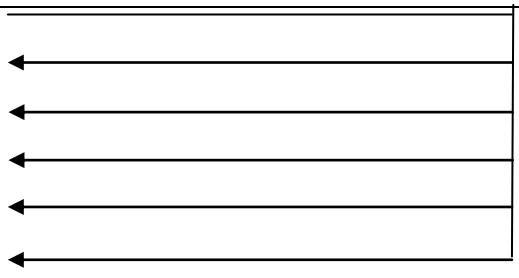
Наконец, авторизация доступа к объектам БД производится на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL разного вида пользователь должен обладать различными полномочиями. Пользователь, создавший таблицу БД, обладает полным набором полномочий для работы с этой таблицей. В число таких полномочий входит право на передачу всех или части полномочий другим пользователям, включая полномочие на передачу полномочий. Полномочия пользователей описываются в специальных таблицах-каталогах, а контроль полномочий поддерживается на языковом уровне .

Реляционная модель - модель представления данных, которая описывает структуру данных, допустимые операции над данными и специальные правила, обеспечивающие целостность данных.

Понятие функциональной зависимости является базовым, так как на его основе формулируется определение всех остальных видов зависимостей.

В разработанной базе данных «Склад» существуют следующие функциональные зависимости между атрибутами:

Таблица 2.1 – Поставщики

Наименование атрибутов	Функциональные зависимости
Код_поставщика	
Название_поставщика	
Адрес_поставщика	
Телефон_поставщика	
Код_товара	
Код_заказа	

Таблицы 2.2 – Заказы

Наименование атрибутов	Функциональные зависимости
Код_заказа	
Название_заказа	
Дата	
Цена	
Количество_товаров	
Код_поставщика	
Код_товара	

Комплекс задач этого этапа состоит из выявления общих информационных объектов и связей между ними, анализа общих информационных требований к системе и выявление информационных потоков, отображающих процессы производства, обработки и взаимодействия данных.

Таблица 2.3 – Товары

Наименование атрибутов	Функциональные зависимости
Код_товара	
Название_товара	
Код_заказа	
Количество_товаров	
Имеется_товаров	
Ожидается_товаров	
Код_поставщика	

Таблица 2.4 – Сотрудники

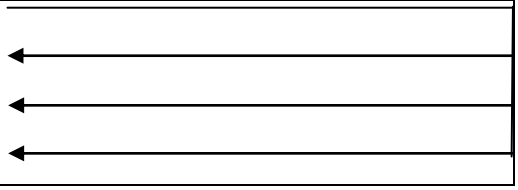
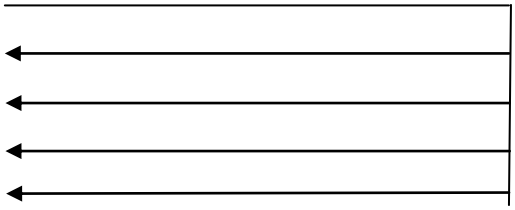
Наименование атрибутов	Функциональные зависимости
Код_сотрудника	
ФИО_сотрудника	
Телефон_сотрудника	
Код_ склада	

Таблица 2.5 – склады

Наименование атрибутов	Функциональные зависимости
Код_склада Название_ склад Адрес_ склада Телефон_ склада Код_сотрудника	

2.1 Инфологическая модель базы данных

Информационные потоки отображают алгоритмический аспект обработки данных и в большей степени относятся к области проектирования приложений. Информация, предоставляемая в БД, в первую очередь должна отображать реальные объекты прикладной области и связи между ними.

Результатом инфологического проектирования является концептуальная модель, которая представляет структуру данных не зависимую от любой физической реализации. Инфологическая модель данных представлена на рисунке 2.1

2.1.1 Описание сущностей

Сущность (объектное множество, таблица) – это собирательное понятие, абстракция реально существующего процесса, объекта или явления, о котором необходимо хранить информацию.

Ниже приведен перечень сущностей, спроектированных в ходе выполнения данной курсовой работы:

- Сущность «Поставщики» хранит информацию о поставщиках.

- Сущность «Заказы» содержит информацию о заказах.

- Сущность «Товары» хранит информацию о товарах.

- Сущность «Склады» содержит информацию об складах.

- Сущность «Сотрудники» хранит информацию о сотрудниках, которые работают в складах.

2.1.2 Описание связей

Связь – ассоциирование двух и более сущностей. Если бы назначением БД было только хранение отдельных, не связанных между собой данных, то ее структура могла быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по назначениям других, для чего необходимо установить между ними определенные связи.

Взаимосвязи между таблицами БД могут быть типизированы по следующим основным видам:

1. отношение “один к одному” (1:1) означает, что каждая запись одной таблицы соответствует только одной записи в другой таблице;

2. отношение “один ко многим” (1:M) возникает, когда одна запись взаимосвязана со многими другими;

3. отношение “многие к одному” означает, что многие записи связаны с одной (M:1);

4. отношение “многие ко многим” (M:N) возникает между двумя таблицами в тех случаях, когда:

– одна запись из первой таблицы может быть связана более чем с одной записью из второй таблицы;

– одна запись из второй таблицы может быть связана более чем с одной записью из первой таблицы.

Отношения между сущностями приведены в таблице 2.6

Таблица 2.6 – Отношения между таблицами

Номер связи	Родительская таблица	Дочерняя таблица	Тип связи
1	Поставщики	Заказы	1:M
2	Заказы	Товары	1:M
3	Товары	Склады	1:M
4	Склады	Сотрудники	1:M

Таблица 2.6 показывает классификацию связей между таблицами.

В данной работе, в базе данных «Склад» используется связь «один ко многим».

Это объясняется характеристикой самой БД, в которой присутствует четыре связи «один ко многим». К первой из них можно отнести связь «Поставщики-Заказы». Здесь, за каждым заказом может быть закреплен лишь один поставщик. В свою очередь, каждый поставщик имеет несколько заказов.

Связь «Заказы-Товары» является связью «один ко многим», т.к. один заказ может содержать несколько товаров.

Связь «Товары-Склады», «Склады-Сотрудники» также являются связью «один ко многим».

2.1.3 ER-диаграмма

Результатом инфологического проектирования является концептуальная модель, которая представляет структуру данных не зависимую от любой физической реализации. Инфологическая модель данных представлена на рисунке 2.1

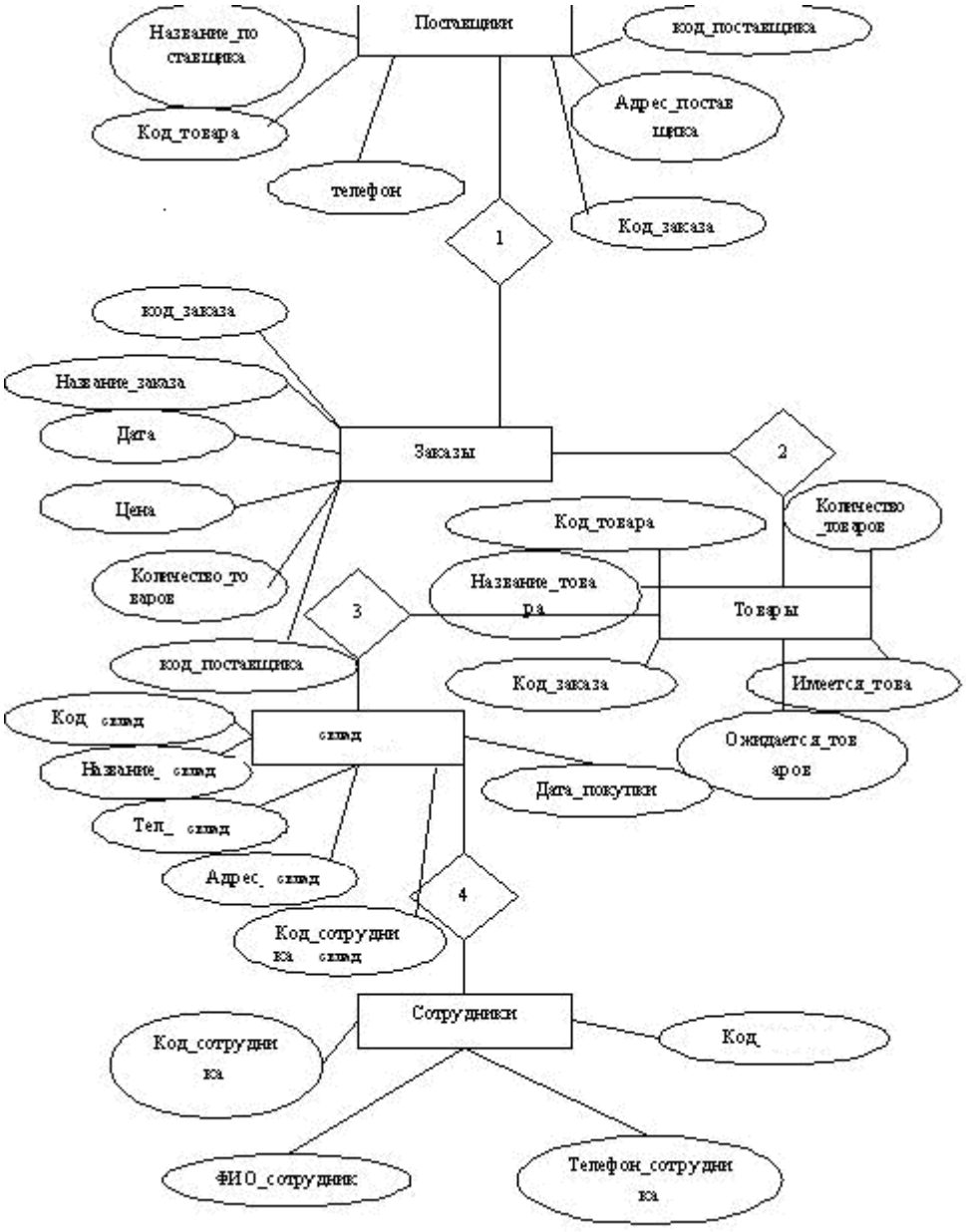


Рисунок 2.1 – Инфологическая модель

2.2 Даталогическая модель

В этом разделе приводится состав таблиц БД. Для каждого поля таблицы указывается размер поля (количество символов), тип. Для первичных ключей необходимо ввести запрет неопределенных значений. Для остальных полей возможность запрета неопределенных значений определяется семантикой предметной области.

Таблица 2.7 – Атрибуты таблицы «Поставщики»

Наименование атрибутов	Тип полей	NULL
Код_поставщика	int	
Название_поставщика	text	
Адрес_поставщика	text	
Телефон_поставщика	numeric(10)	
Код_товара	int	
Код_заказа	int	

Ключи таблицы: Код_поставщика (первичный ключ).

Таблица 2.8 – Атрибуты таблицы «Заказы»

Наименование атрибутов	Тип полей	NULL
Код_заказа	int	
Название_заказа	text	
Дата	datetime	
Цена	money	
Количество_товаров	numeric(10)	
Код_поставщика	int	
Код_товара	int	

Ключи таблицы: Код_заказа (первичный ключ).

Таблица 2.9 – Атрибуты таблицы «Товары»

Наименование атрибутов	Тип полей	NULL
Код_товара	int	
Название_товара	text	
Код_заказа	int	

Количество_товаров	numeric(10)	
Имеется_товаров	int	
Ожидается_товаров	numeric(10)	
Код_поставщика	int	

Ключи таблицы: Код_товара (первичный ключ).

Таблица 2.10 – Атрибуты таблицы «Сотрудники»

Наименование атрибутов	Тип полей	NULL
Код_сотрудника	int	
ФИО_сотрудника	text	
Телефон_сотрудника	numeric(10)	
Код_склада	int	

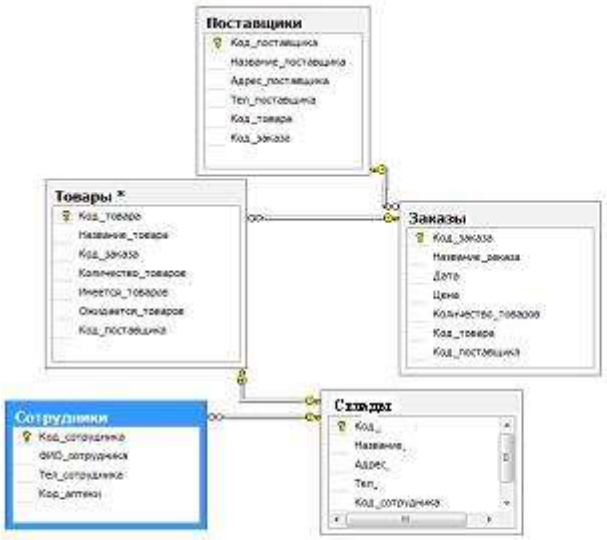
Ключи таблицы: Код_сотрудника (первичный ключ).

Таблица 2.11 – Атрибуты таблицы «Склады»

Наименование атрибутов	Тип полей	NULL
Код_ склада	int	
Название_ склада	text	
Адрес_ склада	text	
Телефон_ склада	numeric(10)	
Код_сотрудника	int	

Ключи таблицы: Код_склада (первичный ключ).

2.2.1 Диаграмма связи по полям



2.3 Нормализация

В ходе создания таблиц мы уже определились со структурой полей, их типами и размерностью, а также смыслом хранимой в них информации. И следующим шагом на пути проектирования структуры базы данных является нормализация.

В качестве примера используем табл. 2.7, которая содержит данные занятий одного курса. Для каждой строки занятия имеются код, данные преподавателя и студента.

Таблица 2.7 – Исходная таблица

Код предмета	Предмет	Преподаватель	Код студента	Фамилия студента
1	СУБД	Айтмуратов	30	Абсеметов
1	СУБД	Айтмуратов	31	Нукусбаев
1	СУБД	Айтмуратов	32	Жолдасбаев
2	Математика	Арзиев	31	Абсеметов

2	Математика	Арзиев	32	Жолдасбаев

При использовании универсального отношения возникают две проблемы:

1) избыточность данных;

2) потенциальная противоречивость (аномалии).

Под избыточностью понимают повторение данных в разных строках одной таблицы или в разных таблицах БД. Так, для каждого студента повторяются данные «1, СУБД и «2, Математика»

Аномалии – это проблемы, возникающие в данных из-за дефектов проектирования БД. Существуют три вида аномалий: вставки, удаления и модификации.

Аномалии вставки проявляются при вводе данных в дефектную таблицу. Добавляя информацию о студенте, мы должны добавить номер, название предмета и фамилию преподавателя. Если ввести данные, не соответствующие имеющимся в таблице, будет не ясно, какая из строк БД содержит правильную информацию.

Аномалии удаления возникают при удалении данных из дефектной схемы.

Аномалии модификации возникают при изменении данных дефектной схемы.

Решение перечисленных проблем состоит в разделении данных и связей, что обеспечивается процедурой нормализации. Концепции и методы нормализации были разработаны Э. Ф. Коддом [4].

Нормализация отношений – это формальный аппарат ограничений на формирование отношений, который позволяет устранить дублирование и потенциальную противоречивость хранимых данных, уменьшает трудозатраты на ведение БД. Процесс нормализации заключается в декомпозиции исходных отношений на более простые отношения. Цель

нормализации – получение такого проекта БД, в котором «каждый факт появляется лишь в одном месте».

Каждая ступень процесса нормализации приводит схему отношений в последовательные нормальные формы. Для каждой ступени имеются наборы ограничений. Выделяют следующую последовательность нормальных форм:

первая нормальная форма (1НФ);

вторая нормальная форма (2НФ);

третья нормальная форма (3НФ);

нормальная форма Бойса-Кодда (БКНФ);

четвертая нормальная форма (4НФ);

пятая нормальная форма (5НФ).

Отношение находится в первой нормальной форме (1НФ), когда каждая строка содержит только одно значение для каждого атрибута (столбца), то есть все атрибуты отношения имеют единственное значение (являются атомарными).

Отношение находится во второй нормальной форме (2НФ), если оно находится в 1НФ, и каждый неключевой атрибут полностью функционально зависит от всех составляющих первичного ключа. Если атрибут не зависит полностью от первичного ключа, то он внесен ошибочно и должен быть удален. Нормализация производится путем нахождения существующего отношения, к которому относится данный атрибут, или созданием нового отношения, в который атрибут должен быть помещен.

Для приведения нашей таблицы в 2НФ необходимо декомпозировать исходное отношение на два (табл. 2.8, 2.9), в которых все неключевые атрибуты будут полностью функционально зависеть от ключа [5].

Таблица 2.8 – Таблица «Занятия»

Предмет	Преподаватель
СУБД	Айтмуратов
Математика	Арзиев

Декомпозиция исходной таблицы (табл 2.9).

Таблица 2.9 – Таблица «Студенты»

Код предмета	Код студента	Фамилия студента
1	30	Абсеметов
1	31	Нукусбаев
1	32	Жолдасбаев
2	31	Абсеметов
2	32	Жолдасбаев

Отношение находится в третьей нормальной форме (3НФ), если оно находится во 2НФ и ни один из его неключевых атрибутов не связан функциональной зависимостью с любым другим неключевым атрибутом. Атрибуты, зависящие от других неключевых атрибутов, нормализуются путем перемещения зависимого атрибута и атрибута, от которого он зависит, в новое отношение. Формально, для приведения схемы в 3НФ необходимо исключить все транзитивные зависимости.

Нормальная форма Бойса-Кодда (БКНФ) является развитием 3НФ и требует, чтобы в отношении были только такие функциональные зависимости, левая часть которых является потенциальным ключом отношения. Потенциальный ключ представляет собой атрибут (или множество атрибутов), который может быть использован для данного отношения в качестве первичного ключа. Фактически первичный ключ – это один из потенциальных ключей, назначенный в качестве первичного. Детерминантом называется левая часть функциональной зависимости. Отношение находится в БКНФ тогда и только тогда, когда каждый детерминант отношения является потенциальным ключом [4,5].

Теперь проведем нормализацию относительно наших таблиц. Рассмотрим таблицу «Покупатель» (табл. 2.10).

Таблица 2.10 – Таблица «Покупатель»

Атрибут	Тип данных	PKEY	FKEY	NOT_NULL
ID_покупатель	INT	*		
ФИО	VARCHAR(50)			*
Адрес	VARCHAR(50)			*
Телефон	VARCHAR(50)			*

Наше отношение находится в БКНФ, так как все атрибуты имеют единственное значение (атомарные), любой неключевой атрибут функционально зависит от первичного ключа, отсутствует транзитивная зависимость, а так же ключ является потенциальным ключом.

ГЛАВА 2. СРЕДА ПРОГРАММИРОВАНИЯ C#

2.1 Язык программирования C#

C#(Си-Шарп) - это новый язык программирования от компании Microsoft. Он входит в новую версию Visual Studio - Visual Studio.NET. Кроме C# в Visual Studio.NET входят Visual Basic.NET и Visual C++. Кроме того фирма Borland объявила, что последующие версии C++ Builder и Delphi будут поддерживать платформу .NET (последнее лежит в русле политики Borland - так, например, нынешние версии C++ Builder и Delphi поддерживают, например, такую технологию от Microsoft, как ActiveX).

Одна из причин разработки нового языка компанией Microsoft - это создание компонентно-ориентированного языка для новой платформы .NET. Другие языки были созданы до появления платформы .NET, язык же C# создавался специально под эту платформу и не несет с собой груза совместимости с предыдущими версиями языков. Хотя это не означает, что для новой платформы это единственный язык.

Платформа .NET—многоязыковая среда, открытая для свободного включения новых языков, создаваемых не только Microsoft, но и другими фирмами. Все языки, включаемые в платформу .NET, должны опираться на единый каркас, роль которого играет .NET Framework. Это серьезное ограничение, одновременно являющееся и важнейшим достоинством.

.NET Framework и библиотека классов

В основе большинства приложений, создаваемых в среде VC++ 6.0, лежал каркас приложений (Application Framework), ключевую роль в котором играла библиотека классов MFC. Когда создавался новый проект MFC — EXE, ActiveX или DLL, из каркаса приложений выбирались классы, необходимые для построения проекта с заданными свойствами. Выбранные классы определяли каркас конкретного приложения.

В каркасе Framework.NET можно выделить два основных компонента:

- статический - FCL (Framework Class Libraru) – библиотека классов каркаса
- динамический – CLR (Common Language Runtime) – общеязыковую испольнительную среду

Каркас .NET также содержит библиотеку классов (Class Library). Она служит тем же целям, что и любая библиотека классов, входящая в каркас. Библиотека включает множество интерфейсов и классов, объединенных в группы по тематике. Каждая группа задается пространством имен (namespace), корневое пространство имен называется System. Классы библиотеки связаны отношением наследования. Все классы являются наследниками класса System.Object. Для классов библиотеки, равно как и для классов в языке C#, не определено множественное наследование.

Среда выполнения Common Language Runtime

Библиотека классов — это статическая составляющая каркаса. В .NET Framework есть и динамическая составляющая — система, определяющая среду выполнения,— CLR (Common Language Runtime). Роль этой среды весьма велика — в ее функции входит управление памятью, потоками, безопасностью, компиляция из промежуточного байт-кода в машинный код и многое другое. Важный элемент CLR — это мощный механизм «сборки мусора» (garbage collector), управляющий работой с памятью.

Язык C# в полной мере позволяет использовать все возможности CLR.

Код, создаваемый на C#, в принципе безопасен. Иными словами, если вы будете придерживаться установленной методики программирования на C#, то вряд ли сможете написать код, способный привести к неправильному обращению с памятью.

В целом, следует отметить следующие моменты:

• среда .NET Framework, задающая единый каркас многоязыковой среды разработки приложений, во многом ориентирована на компонентное

программирование. Она оказывает несомненное влияние на все языки, поддерживающие эту технологию;

• разработчики .NET Framework просто не могли не создать новый язык программирования, который в полной мере отвечал бы всем возможностям .NET Framework, не неся на себе бремени прошлого. Таким языком и стал язык C#;

• главным достоинством языка C# можно назвать его согласованность с возможностями .NET Framework и вытекающую отсюда компонентную ориентированность

Структура программы

Прежде всего, рассмотрим обобщенную структуру программы на языке программирования C#. Проиллюстрируем структуру программы показательным примером (рис. 2.1.).

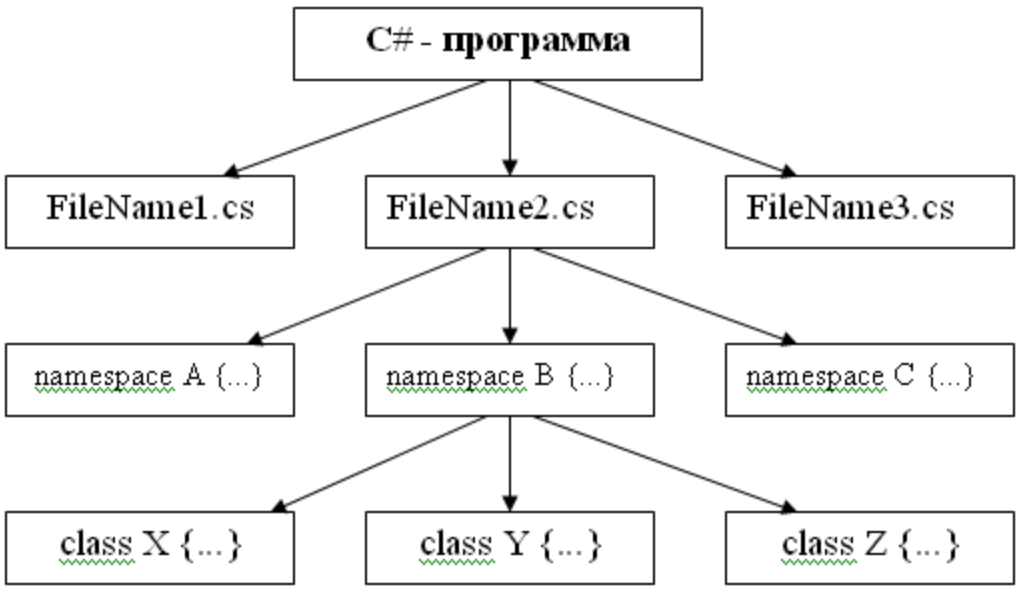


Рис. 2.1. Структура программы на C#

Заметим, что программа на C# может состоять как из одного, так и из нескольких файлов, содержащих исходный текст на языке программирования C#. Каждый такой файл имеет расширение .CS.

Любой файл с исходным текстом на языке программирования C# может как содержать пространства имен, так и не содержать их (в нашем примере файл названы FileName2.cs содержит три пространства имен (A, B и C), а FileName1.cs и FileName3.cs не содержат пространств имен).

Наконец, каждое пространство имен может как содержать описание (одного или нескольких) классов, так и не содержать их (в нашем примере пространство имен B содержит три описания трех классов (X, Y и Z), а пространство имен A и C не содержат ни одного описания классов).

Следовательно, программа на C# состоит из одного или нескольких файлов. Каждый файл может содержать одно или несколько пространств имен. Каждое пространство имен может содержать вложенные пространства имен и типы, такие как классы, структуры, интерфейсы, перечисления и делегаты— функциональные типы. При создании нового проекта C# в среде Visual Studio выбирается один из 10 возможных типов проектов, в том числе Windows Application, Class Library, Web Control Library, ASP.NET Application и ASP.NET Web Service. На основании сделанного выбора автоматически создается каркас проекта.

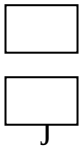
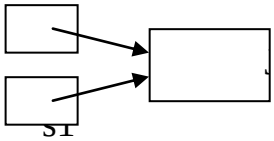
Сопоставление ссылочных типов и типов-значений

Рассмотрим более подробно реализацию двух основных семейств типов данных, а именно, ссылочных типов и типов-значений, применительно к языку программирования C#. Для определенности рассмотрим случай одного из простейших объектов языка программирования C#, а именно, переменной.

таблица 2.1. Сопоставление ссылочных типов и типов-значений

	Типы-значения	Ссылочные типы
Переменная содержит	значение	ссылку на значение
Переменная хранится	в стеке	в куче
Значение по умолчанию	0, false, '\0'	null

Оператор присваивания	копирует значение	копирует ссылку

	int i = 25;
	int j = i;
	 
		string s = “John";
		string s1 = s;
		 

В соответствии с названиями, переменная в случае использования типов-значений содержит собственно значение, а при использовании ссылочных типов – не само значение, а лишь ссылку (указатель) на него.

Местом хранения переменной, определенной как тип-значение, является стек, а определенной как ссылочный тип – «куча» (последнее необходимо для динамического выделения и освобождения памяти для хранения переменной произвольным образом).

Значением, которым переменная инициализируется по умолчанию (необходимость выполнения этого требования диктуется идеологией безопасности Microsoft .NET) в случае определения посредством типа-значения является 0 (для целого или вещественного типа данных), false (для логического типа данных), '\0' (для строкового типа данных), а в случае определения посредством ссылочного типа – значение пустой ссылки null.

При выполнении оператора присваивания в случае переменной-значения копируется значение, а в случае переменной-ссылки – ссылка.

Классы в C#. Объектно-ориентированное программирование и проектирование построено на классах. Любую программную систему, выстроенную в объектном стиле, можно рассматривать как совокупность классов, возможно объединённые в проекты, пространство имен, решения.

У класса две различные роли: модуля и типа данных.

Класс – это модуль, архитектурная единица построения программной системы. Модульность построения – основное свойство программных

систем. В ООП программная система строящаяся по модульному принципу, состоит из классов, являющихся основным видом модуля.

Класс – это тип данных, задающий реализацию некоторой абстракции данных, характерной для задачи, в интересах которой создаётся программная система.

Состав класса, его размер поределяется не архитектурными соображениями, а той абстракцией данных, которую должен реализовать класс. Объектно-ориентированная разработка программной системы основана на стиле, называемом проектированием от данных. Проектирование системы сводится к поиску абстрактных данных, подходящих для конкретной задачи. Каждая из таких абстракций реализуется в виде класса, которые и становятся модулями – архитктурными единицами построения нашей системы. В основе класса лежит абстрактный тип данных.

Синтаксис описания класса:

[атрибуты] [модификаторы] class имя_класса[: список_родителей]

{тело_класса}

В теле класса могут быть объявлены:

- константы;
- поля;
- конструкторы и деструкторы;
- методы;
- события;
- делегаты;
- классы (структуры, интерфейсы, перечисления)

Строки (класс System.String)

Строки в C# - это экземпляры класса System.String. Вообще говоря, в C# есть тип string, но класс System.String яляется более продвинутым, так что его использование часто оказывается более оправданным и простым. Этот

класс имеет множество методов и свойств, некоторые из которых перечислены ниже:

Свойство **Length**. Возвращает длину строки. Пример использования:

```
String s="qqq";
int k=s.Length; //В k запишется 3
```

Compare. Статический метод, сравнивающий две строки. Возвращает 0, есть строки равны, отрицательное значение, если первая строка меньше второй и положительное значение, если первая строки больше второй (больше и меньше в алфавитном смысле, разумеется). Пример использования:

```
namespace test
{ class Test
  {  public static void Main()
    {  String s1="arbour", s2="ace", s3="azote";
       System.Console.WriteLine(String.Compare(s1, s1));
//Выдаст 0, т. к. "arbour" равно "arbour".
       System.Console.WriteLine(String.Compare(s1, s2));
//Выдаст -1, т. к. "arbour" меньше "ace".
       System.Console.WriteLine(String.Compare(s1, s3));
//Выдаст 1, т. к. "arbour" больше "azote".
    } } }
```

Equals. Метод, возвращает true, если строки равны, false - если не равны. Может быть как статическим, так и не статическим. Пример использования:

```
String s1="qqq", s2="www";
System.Console.WriteLine(String.Equals(s1, s2).ToString());
//Статический метод
System.Console.WriteLine(System.Console.WriteLine(String.Equals(s1,s2).ToString()); //Не статический метод
System.Console.WriteLine(s1.Equals(s2).ToString());.ToString());
```

Метод **Substring**. Поволяет извлечь из строки подстроку. Пример использования:

```
String s1="abcdefg", s2;
s2=s1.Substring(3, 2);
System.Console.WriteLine(s2); //Напечатается "de"
```

Параметры тут такие: первый - с какого места извлекаем (нумерация с нуля) и второй - сколько символов извлекаем.

Метод **Insert**. Вставляет в строку другую строку. Пример использования:

```
    String s1="abcdefg", s2;
s2=s1.Insert(1, "xyz");
System.Console.WriteLine(s2); //Напечатается "axyzbcdefg"
```

Первый параметр тут - это куда вставляем (нумерация, как всегда, с нуля), второй - что за строчку вставляем.

Метод **IndexOf**. Позволяет найти в строке подстроку. Пример использования:

```
    String s1="abcabcab", s2="bc", s3="zzz";
System.Console.WriteLine(s1.IndexOf(s2)); //Напечатается 1
System.Console.WriteLine(s1.IndexOf(s3)); //Напечатается -1
```

Этот метод возвращает номер позиции, на котором в строке находится передаваемая в качестве параметра подстрока. Если такой подстроки нет, то возвращается -1.

Метод **Replace**. Производит замену в строке. Пример использования:

```
    String s1="abcabcab", s2="bc", s3;
s3=s1.Replace(s2, "q");
System.Console.WriteLine(s3); //Напечатается aqaqab
```

Методы **EndWith** и **StartsWith**. Проверяют, не завершается ли или не начинается ли строка с или на заданную строку соответственно. Пример использования:

```
    String s1="arbour";
if(s1.StartsWith("ar"))
System.Console.WriteLine("Строка начинается на \"ar\"");
else
 System.Console.WriteLine("Строка не начинается на \"ar\"");
```

Методы **ToUpper** и **ToLower** переводят строку в верхний или нижний регистр соответственно. Пример использования:

```
String s1="aRbRur";
s1=s1.ToLower();
```

Методы **Trim**, **TrimEnds** и **TrimStart**. Удаляют пробельные символы из начала и конца строки (Trim), только с конца строки (TrimEnds) и только с начала строки (TrimStart). Пример использваония:

```
String s1="  ar  brur ";
System.Console.Write(s1.Trim());
```

При изменении строки фактически старый экземпляр класса System.String уничтожается, и создается новый с там же именем и измененным содержанием. Это означает, что при интенсивной работе со строками программа может работать не так быстро, как хотелось бы. Если мы не хотим, что бы каждый раз создавался новый экземпляр класса, то вместо класса System.String надо использовать класс StringBuilder.

Преимущества и недостатки языка Си #

Проанализировав основные особенности языка программирования C#, а также исследовав структуру и принципы построения программ на этом языке, обозначим наиболее очевидные преимущества изучаемого языка программирования.

– Прежде всего, необходимо отметить, что язык программирования C# претендует на подлинную объектную ориентированность (а всякая языковая сущность претендует на то, чтобы быть объектом).

– Кроме того, язык программирования C# призван практически реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования (фрагментов) программ.

– Принципиально важным отличием от предшественников является изначальная ориентация на безопасность кода (что особенно заметно в сравнении с языками C и C++).

– Унифицированная, максимально близкая по масштабу и гибкости к Common Type System, принятой в Microsoft .NET, система типизации является важным преимуществом языка C#.

– Расширенная поддержка событийно-ориентированного программирования выгодно отличает язык программирования C# от целого ряда предшественников.

– Язык программирования C# является "родным" для создания приложений в среде Microsoft .NET, поскольку наиболее тесно и эффективно интегрирован с ней.

– Объединение лучших идей современных языков программирования (Java, C++, Visual Basic и др.) делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

Несмотря на значительное количество принципиальных преимуществ по сравнению с существующими аналогами, язык программирования C# не лишен и отдельных недостатков, которые, весьма вероятно, носят субъективный, локальный, временный характер.

– Прежде всего, необходимо отметить то обстоятельство, что язык программирования C# имеет довольно сложный синтаксис (можно утверждать, что примерно 75% его синтаксических возможностей аналогичны языку программирования Java, 10% подобны языку программирования C++, а 5% – заимствованы из языка программирования Visual Basic). Объем действительно свежих концептуальных идей в языке C# относительно невысок (по мнению некоторых исследователей, он, составляет около 10% от общего объема конструкций языка).

– Утверждение, что язык программирования C# является чисто объектным, допускает неоднозначную интерпретацию.

– На сегодня компилятор и среда разработки программного обеспечения, поддерживающие язык C#, обладают относительно невысокой производительностью (т.е. код программы на языке C# компилируется и выполняется примерно в 100 раз медленнее, чем тот же код на языке C). Справедливости ради нужно отметить, что производительность программ на C# вполне сравнима с тем же показателем для языка Java.

Пока программы, написанные на языке C#, не могут функционировать под управлением альтернативных операционных систем (ведутся работы по обеспечению совместимости с операционными системами Linux и FreeBSD семейства UNIX)

2.2. Язык SQL как язык баз данных

Стремительный рост популярности SQL является одной из самых важных тенденций в современной компьютерной промышленности. За несколько последних лет SQL стал *единственным* языком баз данных [1]. На сегодняшний день SQL поддерживают свыше ста СУБД, работающих как на персональных компьютерах, так и на больших ЭВМ. Был принят, а затем дополнен официальный международный стандарт на SQL. Язык SQL является важным звеном в архитектуре систем управления базами данных, выпускаемых всеми ведущими поставщиками программных продуктов, и служит стратегическим направлением разработок компании Microsoft в области баз данных [5-7]. Зародившись в результате выполнения второстепенного исследовательского проекта компании IBM, SQL сегодня широко известен и в качестве мощного рыночного фактора.

SQL является инструментом, предназначенным для обработки и чтения данных, содержащихся в компьютерной базе данных. SQL - это сокращенное название структурированного языка запросов (Structured Query Language).

Как следует из названия, SQL является *языком программирования*, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемых *реляционными*. На рисунке изображена схема работы SQL. Согласно этой схеме, в вычислительной системе имеется *база данных,* в которой хранится важная информация. Если вычислительная система относится к сфере бизнеса, то в базе данных может храниться информация о материальных ценностях, выпускаемой продукции, объемах продаж и зарплате[8-10]. В базе данных на персональном компьютере может храниться информация о выписанных чеках, телефонах и адресах или информация, извлеченная из более крупной вычислительной системы. Компьютерная программа, которая управляет базой данных, называется *системой управления базой данных,* или СУБД [1].

Если пользователю необходимо прочитать данные из базы данных, он запрашивает их у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется *запросом* к базе данных: отсюда и название — структурированный язык *запросов.*

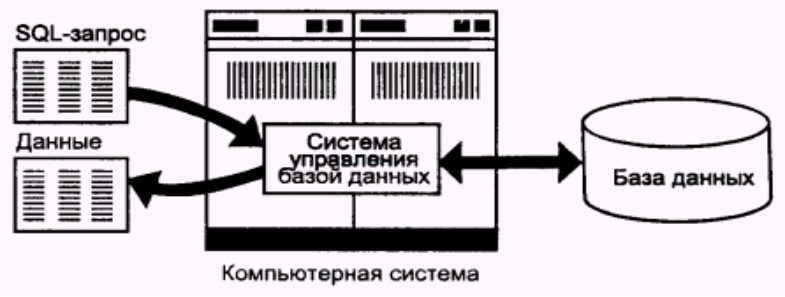


Рис. 2.2. Применение SQL для доступа к базе данных

Однако это название не совсем соответствует действительности. Во-первых, сегодня SQL представляет собой нечто гораздо большее, чем простой инструмент создания запросов, хотя именно для этого он и был первоначально предназначен. Несмотря на то, что чтение данных по-

прежнему остается одной из наиболее важных функций SQL, сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю, а именно:

- *Организация данных.* SQL дает пользователю возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных.

- *Чтение данных.* SQL дает пользователю или приложению возможность читать из базы данных содержащиеся в ней данные и пользоваться ими.

- *Обработка ванных.* SQL дает пользователю или приложению возможность изменять базу данных, т.е. добавлять в нее новые данные, а также удалять или обновлять уже имеющиеся в ней данные.

- *Управление доступом.* С помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа.

- *Совместное использование данных.* SQL координирует совместное использование данных пользователями, работающими параллельно, чтобы они не мешали друг другу.

- *Целостность данных.* SQL позволяет обеспечить целостность базы данных, защищая ее от разрушения из-за несогласованных изменений или отказа системы.

Таким образом, SQL является достаточно мощным языком для взаимодействия с СУБД.

Во-вторых, SQL — это не полноценный компьютерный язык типа PASCAL, BASIC или C. В SQL нет оператора IF для проверки условий, нет оператора GOTO для организации переходов и нет операторов DO или FOR для создания циклов. SQL является *подъязыком* баз данных, в который входит около тридцати операторов, предназначенных для управления базами данных. Операторы SQL *встраиваются* в базовый язык, например PASCAL,

BASIC или C, и дают возможность получать доступ к базам данных. Кроме того, из такого языка, как C, операторы SQL можно посылать СУБД в явном виде, используя *интерфейс вызовов функций.*

2.3. Технология ADO.NET

Первое, что необходимо сказать про ADO.NET, — то, что это не просто улучшенная и расширенная версия классического ADO. У традиционного ADO и ADO.NET уществуют как схожие черты (концепция объектов «соединения» и «командных * > объектов), так и существенные различия (к примеру, в ADO.NET вы уже не найдете такого важного типа, как Recordset). Самые важные типы ADO.NET, в свою очередь, не имеют эквивалентов в мире классического ADO (это справедливо, например, для DataSet)[2].

ADO.NET: общая картина

Все типы ADO.NET предназначены для выполнения единого набора задач: установить соединение с хранилищем данных, создать и заполнить данными объект DataSet, отключиться от хранилища данных и вернуть изменения, внесенные в объект DataSet, обратно в хранилище данных. Объект DataSet — это очень важный и интересный тип данных, представляющий локальный набор таблиц и информацию об отношениях между ними. В некоторых отношениях DataSet очень напоминает рассоединенный Recordset из «старого» ADO. Главное различие между рассоединенным Recordset и DataSet заключается в том, что Recordset представляет собой единственную таблицу, а DataSet — набор связанных таблиц. На практике нам ничто не мешает создать на клиенте объект DataSet, который будет представлять *полную копию* удаленной базы данных. После создания объекта DataSet и его заполнения данными можно программными средствами производить запросы к нему и перемещаться по таблицам.

Возможно выполнение всех операций, как при работе с обычными базами данных: добавлять в таблицы новые записи, удалять и изменять существующие, применять к ним фильтры и т. п. После того как клиент завершит внесение изменений, информация о них будет отправлена в хранилище данных для обработки. Скорее всего, одним из первых вопросов у каждого, кто начал разбираться с ADO.NET, будет: «А как создать DataSet?» Ответ будет звучать так; при помощи управляемого провайдера (managed provider). Управляемый провайдер — это набор классов, реализующих интерфейсы, определенные в пространстве имен System.Data. Речь идет об интерфейсах IDbCommand, IDbDataAdapter, IDbConnection и IDataReader. В состав ADO.NET включены два управляемых провайдера: провайдер SQL и провайдер OleDb. Провайдер SQL специально оптимизирован под взаимодействие с Microsoft SQL Server версии 7.0 и последующих. Для других источников данных предлагается использовать провайдер OleDb, который можно использовать для обращения к любым хранилищам данных, поддерживающим протокол OLE DB. Однако учтите, что провайдер OleDb работает при помощи «родного» OLE DB и требует возможности взаимодействия при помощи COM. Конечно же, в самом скором будущем появятся управляемые провайдеры от производителей СУБД, которые будут эффективно взаимодействовать со «своими» источниками данных. Однако до этого времени в большинстве случаев придется использовать OleDb провайдер.

ГЛАВА 3. СОЗДАНИЯ БАЗЫ ДАННЫХ В СРЕДЕ ПРОГРАММИРОВАНИЯ C#

3.1 Описание программного продукта

Данный программный продукт написан в Microsoft Visual Studio 2015 на языке программирования C#. Для создания базы данных использовался Microsoft SQL Server Express Edition.

Для корректной работы программы необходим один ПК следующей минимальной конфигурации:

1) операционная система Windows;
2) разрешение экрана 1000*700;
3) процессор 1600 Hz;
4) RAM 512 MB;
5) Free HDD space – 100 MB;
6) дисплей, манипулятор «мышь», клавиатура.

Для создания и хранения базы данных необходимо установить Microsoft SQL Server Express Edition.

Назначение программы: организация простой и понятной работы с созданной базой данных.

В первую очередь, чтобы приступить к работе, необходимо создать подключение к базе данных. Следующим шагом будет создание таблиц и заполнение их данными.

3.2 Физическая модель базы данных «Sklad»

Создадим таблицу «Sklad» (рис. 3.1) со следующими полями и соответствующими ограничениями:

```
CREATE TABLE [dbo].[sklad] (
    [Id]                  INT           IDENTITY (1, 1) NOT
NULL,
    [id_post]             INT           NULL,
    [Название_Товара]     VARCHAR (50)  NULL,
    [Единица_измаерения] VARCHAR (50)  NULL,
    [Количество]          INT           NULL,
    [цена_покупки]        INT           NULL,
    [цена_продажи]        INT           NULL,
    [дата поступления]    DATETIME      NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

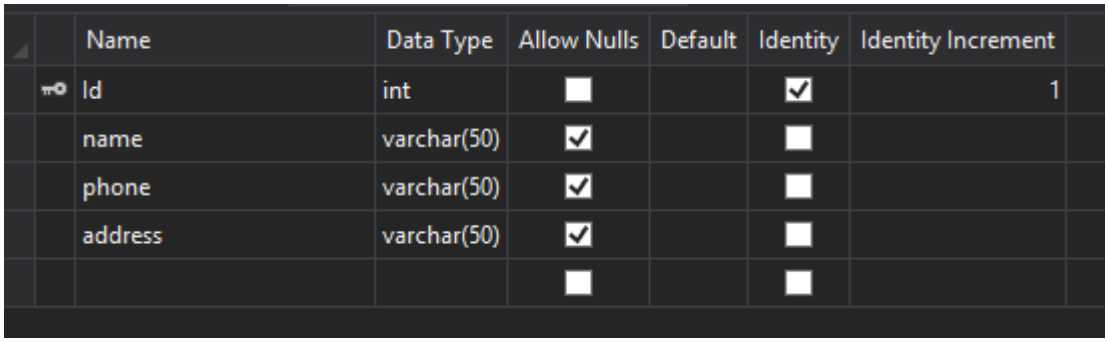
	Name	Data Type	Allow Nulls	Default	Identity	Identity Increment
🔑	Id	int	☐		☑	1
	id_post	int	☑		☐	
	Название_Товара	varchar(50)	☑		☐	
	Единица_измаерения	varchar(50)	☑		☐	
	Количество	int	☑		☐	
	цена_покупки	int	☑		☐	
	цена_продажи	int	☑		☐	
	дата поступления	datetime	☑		☐	
			☐		☐	

Рисунок 3.1 – Определение таблицы «Sklad»

Таблица «Postavshik» (рис. 3.2) будет состоять из таких полей:

```
CREATE TABLE [dbo].[Postavshik] (
    [Id]      INT           IDENTITY (1, 1) NOT NULL,
    [name]    VARCHAR (50)  NULL,
    [phone]   VARCHAR (50)  NULL,
    [address] VARCHAR (50)  NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
```

```
);
```

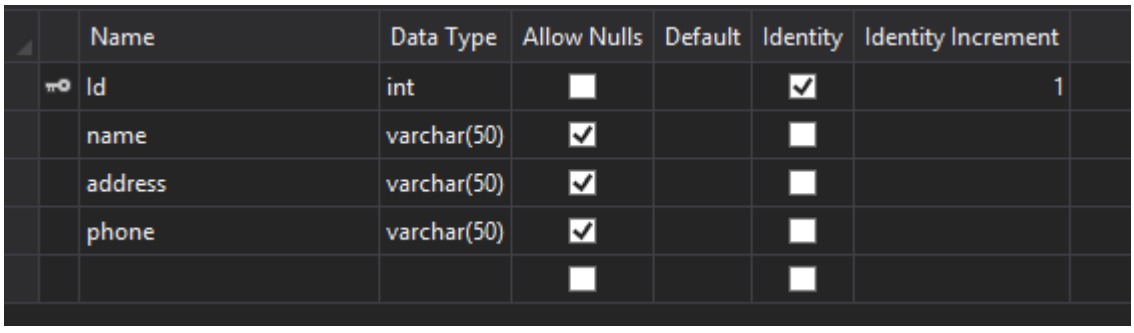


	Name	Data Type	Allow Nulls	Default	Identity	Identity Increment
🔑	Id	int	☐		☑	1
	name	varchar(50)	☑		☐	
	phone	varchar(50)	☑		☐	
	address	varchar(50)	☑		☐	
			☐		☐	

Рисунок 3.2 – Определение таблицы «Postavshik»

Создадим таблицу «Pokupatel» (рис. 3.3) с атрибутами:

```
CREATE TABLE [dbo].[pokupatel] (
    [Id]      INT          IDENTITY (1, 1) NOT NULL,
    [name]    VARCHAR (50) NULL,
    [address] VARCHAR (50) NULL,
    [phone]   VARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```



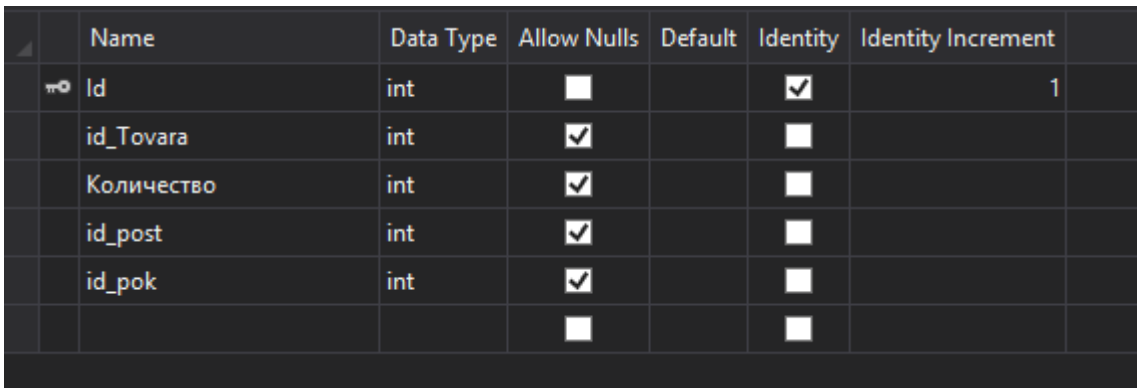
	Name	Data Type	Allow Nulls	Default	Identity	Identity Increment
🔑	Id	int	☐		☑	1
	name	varchar(50)	☑		☐	
	address	varchar(50)	☑		☐	
	phone	varchar(50)	☑		☐	
			☐		☐	

Рисунок 3.3 – Определение таблицы «Pokupatel»

В таблицу «cont» (рис.3.4) необходимо добавить следующие поля:

```
CREATE TABLE [dbo].[cont] (
    [Id]          INT IDENTITY (1, 1) NOT NULL,
    [id_Tovara]   INT NULL,
    [Количество] INT NULL,
    [id_post]     INT NULL,
    [id_pok]      INT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
```

```
    CONSTRAINT [FK_cont_ToPokup] FOREIGN KEY ([id_pok])
REFERENCES [dbo].[pokupatel] ([Id]),
    CONSTRAINT [FK_cont_ToPost] FOREIGN KEY ([id_post])
REFERENCES [dbo].[Postavshik] ([Id]),
    CONSTRAINT [FK_cont_ToSklad] FOREIGN KEY ([id_Tovara])
REFERENCES [dbo].[sklad] ([Id])
);
```

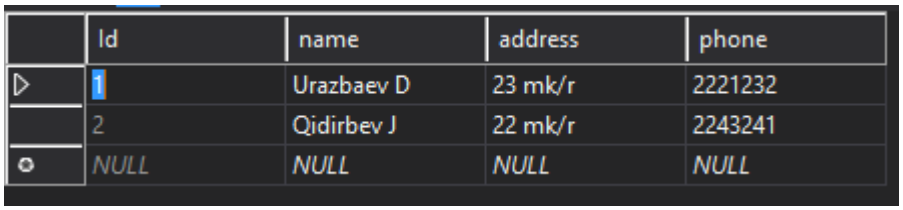


Name	Data Type	Allow Nulls	Default	Identity	Identity Increment
Id	int	☐		☑	1
id_Tovara	int	☑		☐	
Количество	int	☑		☐	
id_post	int	☑		☐	
id_pok	int	☑		☐	
		☐		☐	

Рисунок 3.4 – Определение таблицы «Sklad»

Необходимо заполнить наши таблицы соответствующими данными. Для примера рассмотрим заполнение таблицы «user». Нужно в столбце «username» прописать логин , который работает в складе, чтобы он мог зайти в систему и редактировать данные без помощи системного администратора.

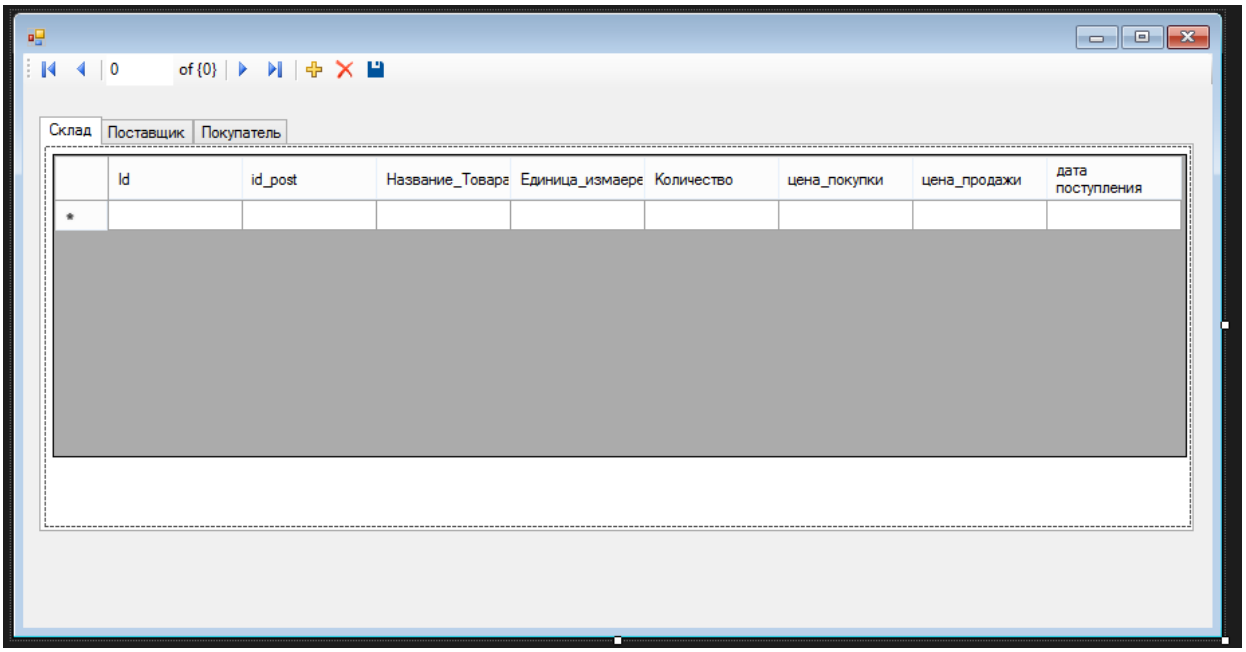
Заполнение таблицы «Pokupatel» (рис. 3.5).



Id	name	address	phone
1	Urazbaev D	23 mk/r	2221232
2	Qidirbev J	22 mk/r	2243241
NULL	NULL	NULL	NULL

Рисунок 3.5 – Заполнение таблицы «Pokupatel»

Вид программы



.

3.3 Описание реализации системы

Защита базы данных от несанкционированного доступа осуществляется с помощью создания роли руководителя, который имеют определенные права доступа к таблицам и отдельным столбцам таблиц базы. Каждый пользователь имеет свою учетную запись на SQL Server, защищенную паролем, для которой определена роль.

При запуске программы открывается окно входа в систему (рис. 3.7), в котором нужно ввести свои данные (логин или пароль) и нажать «Вход».

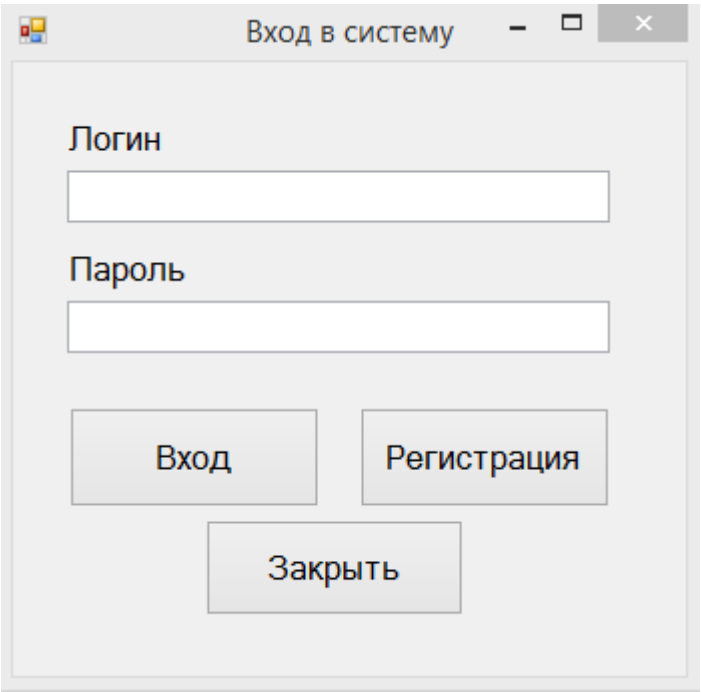


Рисунок 3.7 – Вход в систему

Соответствующий код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace sklad
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        // Строка соединения
        string MyConString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\skladBase.mdf;
Integrated Security=True";
        //private SqlConnection connection1;
        // таблица с данными в памяти

        // Вход пользователя (врача) в систему
        private void btn_Enter_Click(object sender, EventArgs e)
```

```
        {
            int count = 0;
            using (SqlConnection connection = new SqlConnection(MyConString))
            {
                using (SqlCommand command = new SqlCommand("SELECT * FROM
users WHERE username='" + tb_Login.Text + "' AND password='" +
GetHashString(mtb_Password.Text) + "'", connection))
                {
                    connection.Open();
                    using(SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                            count += 1;
                        reader.Close();
                    }
                    connection.Close();

                }
            }
            if (count == 0){
                MessageBox.Show("Пароль неверен!");
                return;
            }
            // открытие следующей формы
            this.Hide();
            Form3 f3 = new Form3();
            f3.Show();

        }
        // шифрование md5
        string GetHashString(string s)
        {
            //переводим строку в байт-массим
            byte[] bytes = Encoding.Unicode.GetBytes(s);

            //создаем объект для получения средст шифрования
            MD5CryptoServiceProvider CSP =
            new MD5CryptoServiceProvider();
            //вычисляем хеш-представление в байтах
            byte[] byteHash = CSP.ComputeHash(bytes);
            string hash = string.Empty;
            //формируем одну цельную строку из массива
            foreach (byte b in byteHash)
                hash += string.Format("{0:x2}", b);
            return hash;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form2 f2 = new Form2();
            f2.Show();

        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            Application.Exit();

        }
    }
}
```

Если данные не соответствуют существующей учетной записи, то следует нажать кнопку «Регистрация». Откроется окно для регистрации пользователя (рис. 3.8).

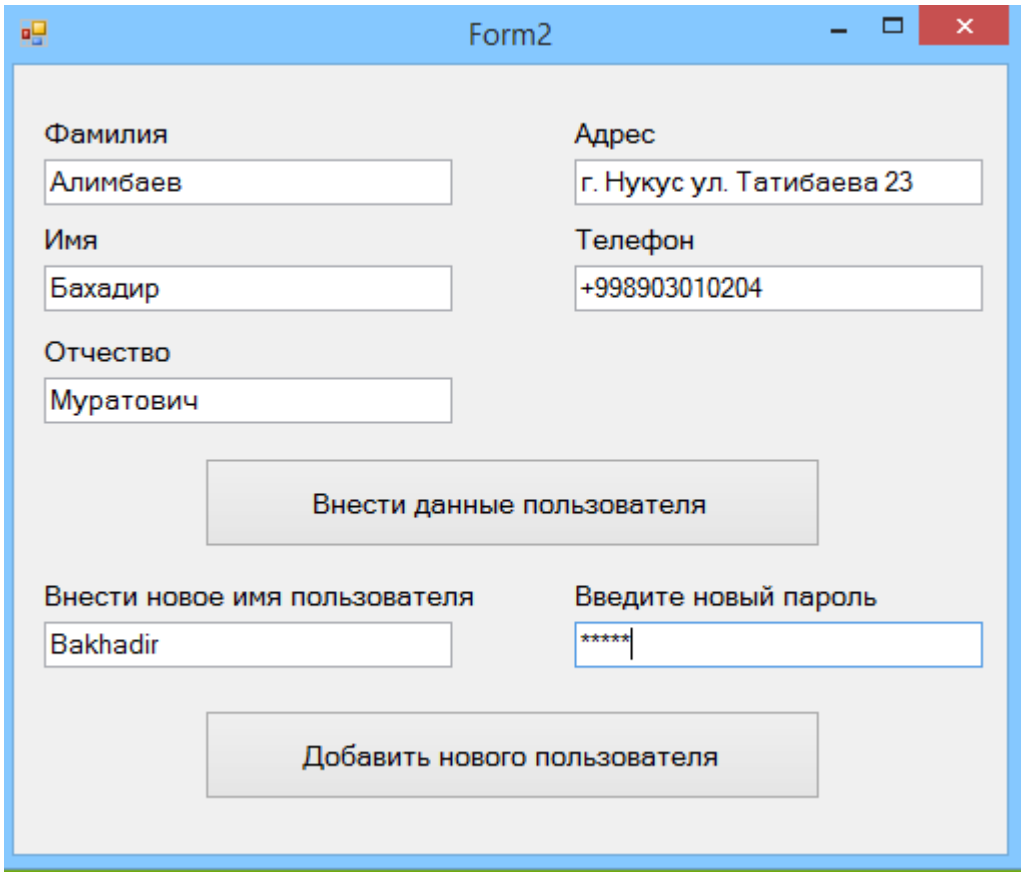


Рисунок 3.8 – Окно регистрации пользователя

Код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedClinic
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        // Строка соединения
        string MyConString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\MedBase.mdf;In
tegrated Security=True";
        private SqlConnection connection;
        // таблица с данными в памяти
        //private SqlDataReader sqlReader;
        // Вход пользователя  в систему
        private void btn_Add_NewInfo_Click(object sender, EventArgs e)
        {
            connection = new SqlConnection(MyConString);
            connection.Open();// открыли строку
            SqlCommand com = connection.CreateCommand();
            com.CommandText = @"SELECT MAX (ID_user) FROM [users];";
            int id = (int)com.ExecuteScalar() + 1;
            SqlCommand command = new SqlCommand(@"Insert into [users] Values
(" + id.ToString() + "," + "'" + tb_Login.Text + "'," + "'" +
GetHashString(tb_Password.Text) + "', 10);", connection);
            command.ExecuteNonQuery();
            connection.Close();
            // закрытие соединения
            this.Close();
            // открытие предыдущей формы
            Form1 fl = new Form1();
            fl.Show();

        }

        private void btn_Add_NewUser_Click(object sender, EventArgs e)
        {
            connection = null;
            connection = new SqlConnection(MyConString);
            connection.Open();// открыли строку
            SqlCommand com1 = connection.CreateCommand();
            com1.CommandText = @"SELECT MAX (ID_Врач) FROM [Врач];";
            int id = (int)com1.ExecuteScalar() + 1;

            com1.CommandText = @"Insert into [Врач] Values (" + id.ToString()
+ "," + "'" + tb_FirstName.Text + "'," + "'" + tb_Name.Text + "'," + "'" +
tb_LastName.Text + "'," + "'" + tb_Address.Text + "'," + " '" +
tb_Telefon.Text + "');";
            com1.ExecuteNonQuery();
            connection.Close();
```

```
        }
        // шифрование md5
        string GetHashString(string s)
        {
            //переводим строку в байт-массим
            byte[] bytes = Encoding.Unicode.GetBytes(s);

            //создаем объект для получения средст шифрования
            MD5CryptoServiceProvider CSP =
            new MD5CryptoServiceProvider();
            //вычисляем хеш-представление в байтах
            byte[] byteHash = CSP.ComputeHash(bytes);
            string hash = string.Empty;
            //формируем одну цельную строку из массива
            foreach (byte b in byteHash)
                hash += string.Format("{0:x2}", b);
            return hash;
        }

    }
}
```

Если данные соответствуют существующей учетной записи, то открывается окно выбора таблицы.

Код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Sklad
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        String myConString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Sklad.m
df;Integrated Security=True";
```

```
        private void skladBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
        {
            this.Validate();
            this.skladBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.skladDataSet);

        }

        private void skladBindingNavigatorSaveItem_Click_1(object
sender, EventArgs e)
        {
            this.Validate();
            this.skladBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.skladDataSet);

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'skladDataSet.pokupatel' table. You can move, or remove it, as needed.

this.pokupatelTableAdapter.Fill(this.skladDataSet.pokupatel);
            // TODO: This line of code loads data into the
'skladDataSet.Postavshik' table. You can move, or remove it, as
needed.

this.postavshikTableAdapter.Fill(this.skladDataSet.Postavshik);
            // TODO: This line of code loads data into the
'skladDataSet.sklad' table. You can move, or remove it, as needed.
            this.skladTableAdapter.Fill(this.skladDataSet.sklad);

        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Sklad
{
    public partial class Form1 : Form
    {
        public Form1()
```

```
        {
            InitializeComponent();
        }

        String myConString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Sklad.m
df;Integrated Security=True";

        private void skladBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
        {
            this.Validate();
            this.skladBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.skladDataSet);

        }

        private void skladBindingNavigatorSaveItem_Click_1(object
sender, EventArgs e)
        {
            this.Validate();
            this.skladBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.skladDataSet);

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'skladDataSet.pokupatel' table. You can move, or remove it, as needed.

this.pokupatelTableAdapter.Fill(this.skladDataSet.pokupatel);
            // TODO: This line of code loads data into the
'skladDataSet.Postavshik' table. You can move, or remove it, as
needed.

this.postavshikTableAdapter.Fill(this.skladDataSet.Postavshik);
            // TODO: This line of code loads data into the
'skladDataSet.sklad' table. You can move, or remove it, as needed.
            this.skladTableAdapter.Fill(this.skladDataSet.sklad);

        }
    }
}
```

После нажатия первой кнопки открывается окно таблицы «соотрудники» (рис. 3.10).

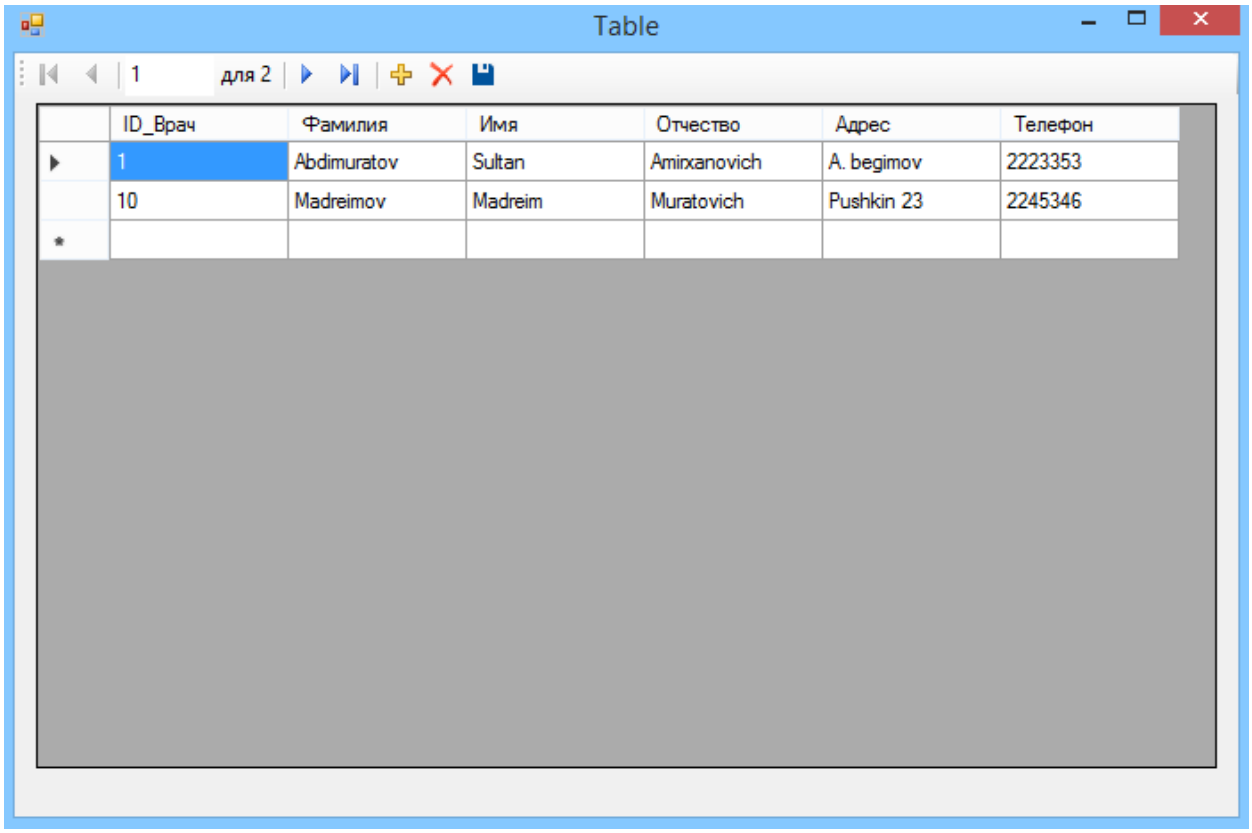


Рисунок 3.10 – Таблица соотрудник

Код программы:

Исходныйкодфайла Form1.Designer.cs:

```
namespace Sklad
{
partialclassForm1
    {
///<summary>
/// Required designer variable.
///</summary>
private System.ComponentModel.IContainer components = null;

///<summary>
/// Clean up any resources being used.
///</summary>
///<param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
protectedoverridevoid Dispose(bool disposing)
        {
if (disposing && (components != null))
            {
                components.Dispose();
            }
base.Dispose(disposing);
```

```
}
```

3.3 Пример использование программы

При первом запуске программы перед пользователем возникнет окно авторизации с кнопками вход и регистрация (рис. 3.11).

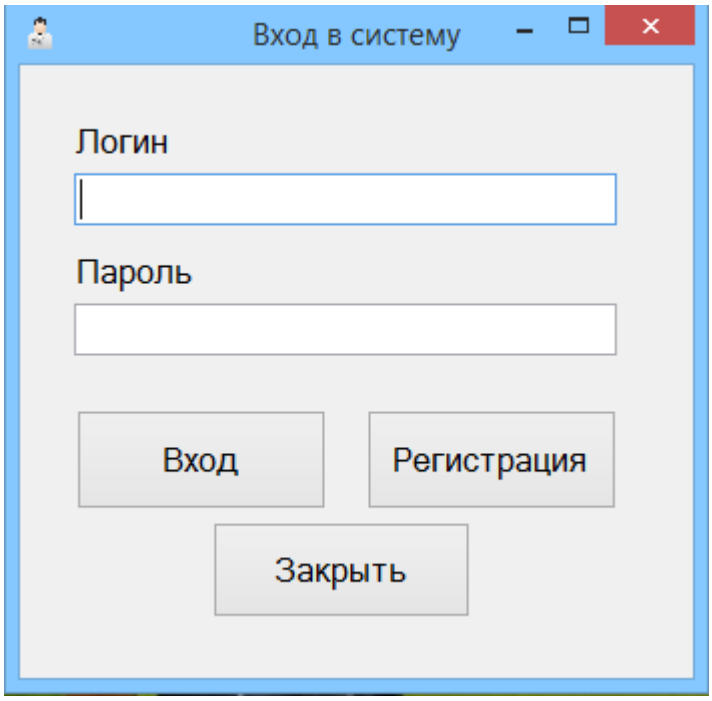


Рисунок 3.11 – Окно авторизации

Если пользователь не зарегистрирован в системе, то ему следует нажать кнопку «Регистрация». Откроется новое окно как на рисунке 3.12. Зарегистрируем нового пользователя .

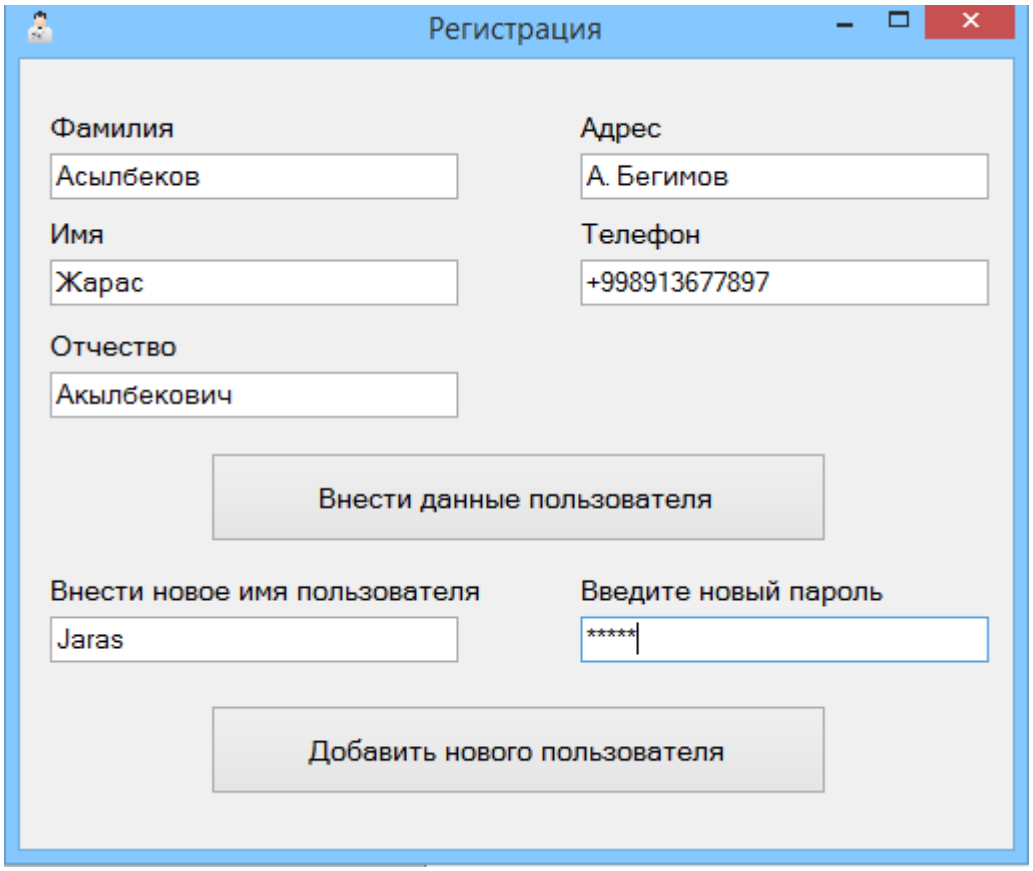


Рисунок 3.12 – Окно регистрации

После нажатия кнопки «Добавить нового пользователя» и «Внести данные пользователя» происходит переход в окно «Вход в систему». Пользователь должен ввести свой новый логин и пароль (рис. 3.13).

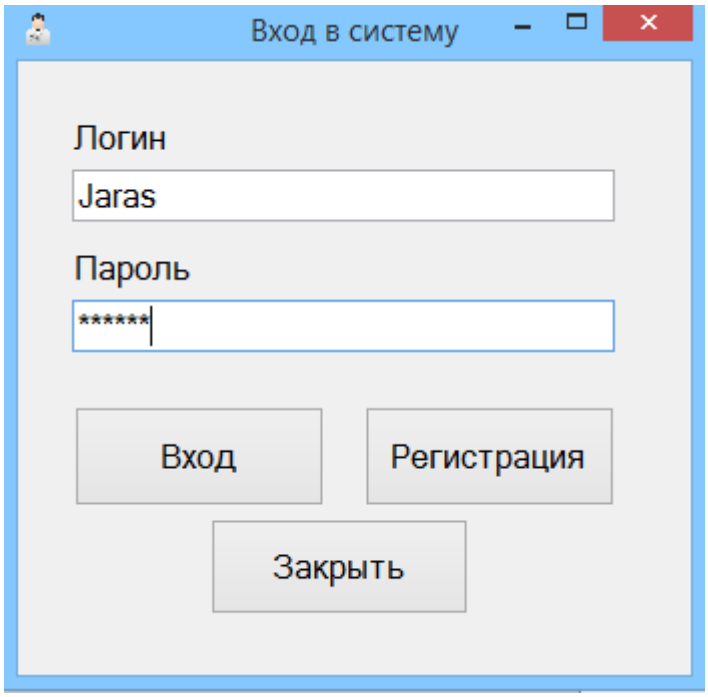


Рисунок 3.13 – Окно авторизации

После нажатия на кнопку «Вход» производится обращение к БД. В случае правильного ввода логина и пароля, и открывается окно выбора таблиц, что означает о разрешении дальнейшей работы.

После корректного подключения пользователь может переходить по всем таблицам. Нажав (рис. 3.14).

После запуска появляется следующее окно:

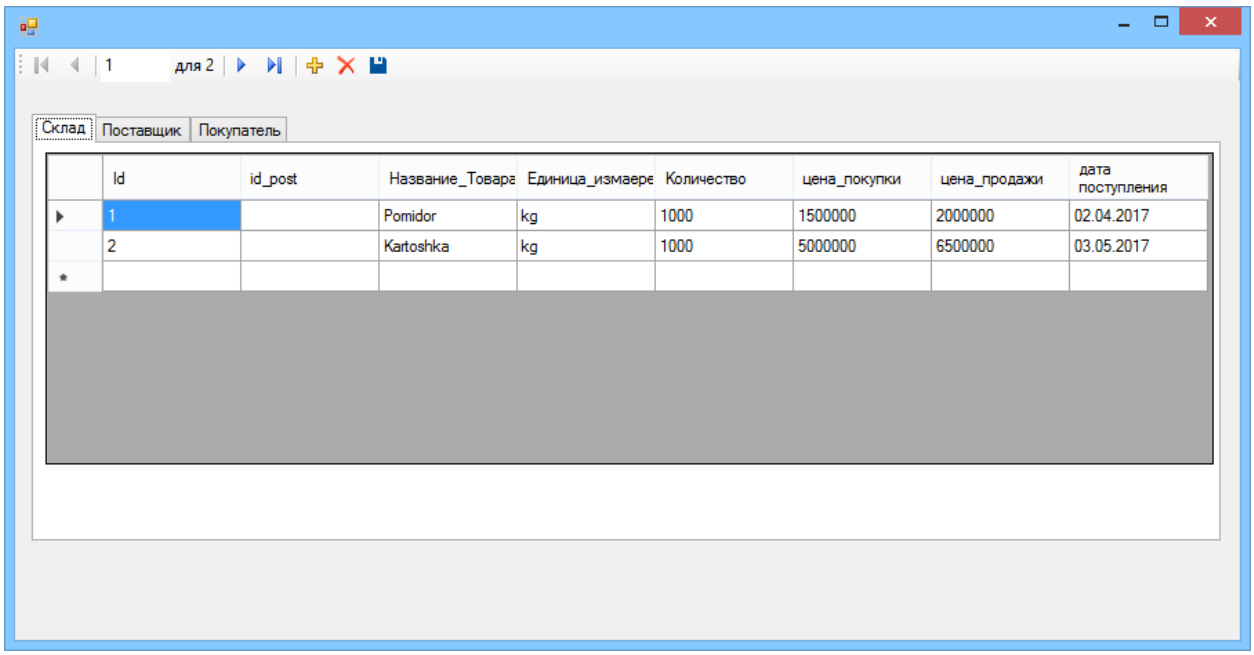
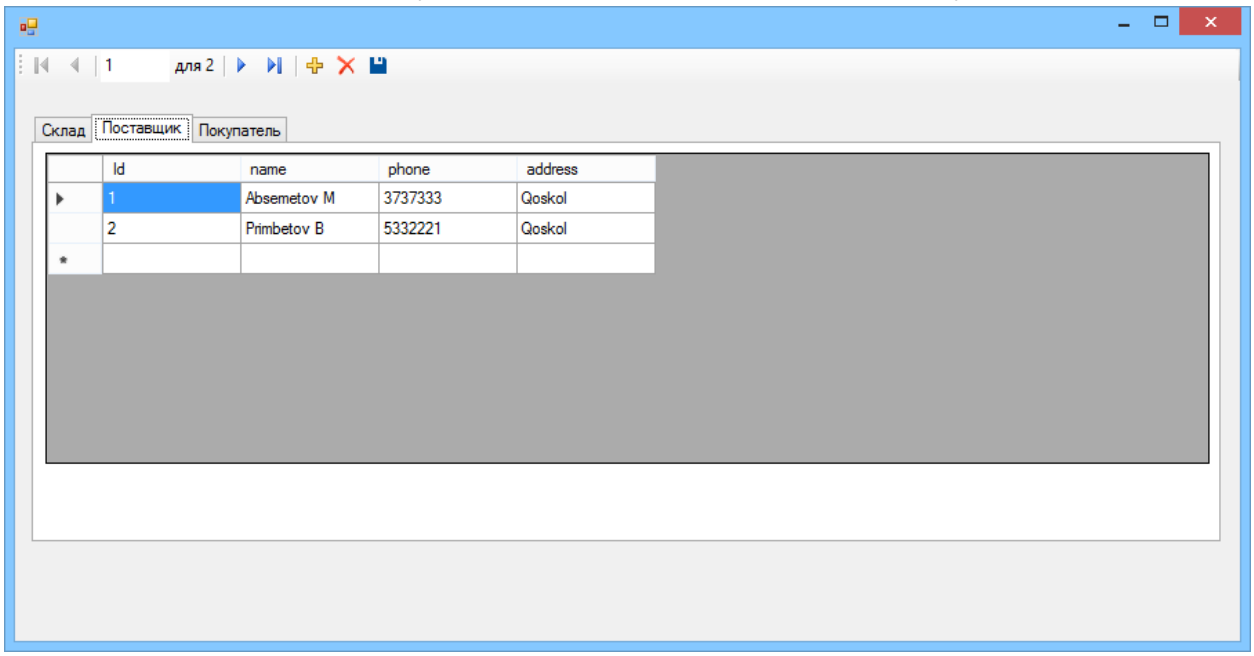
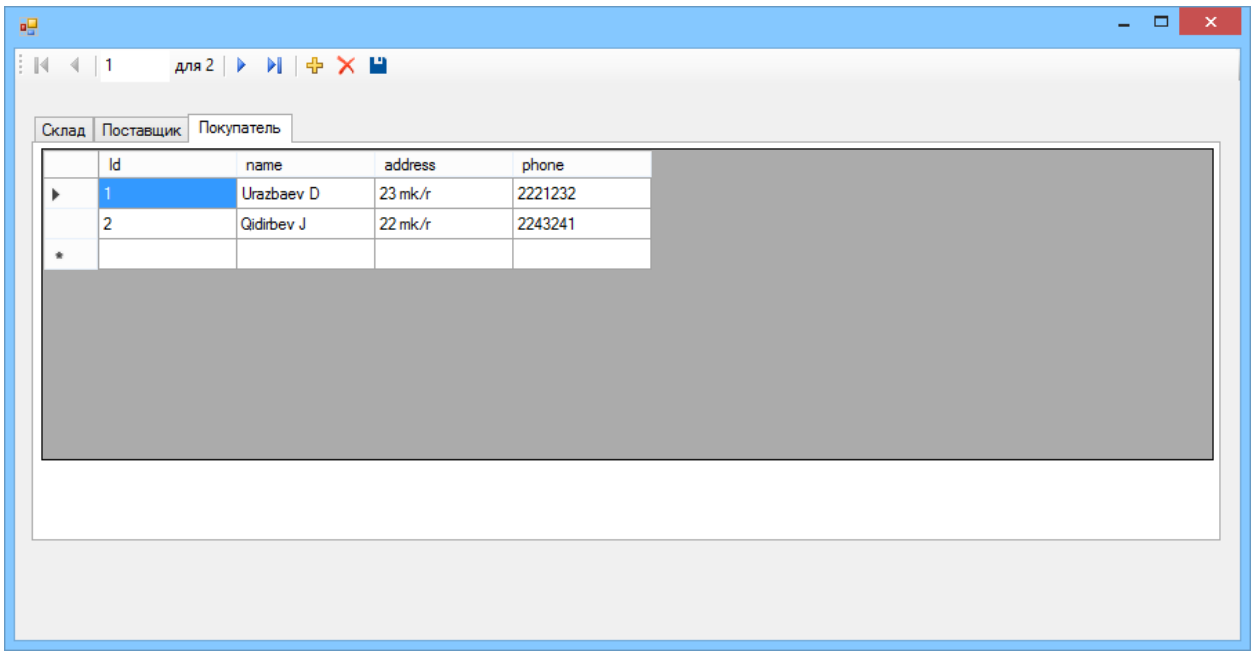


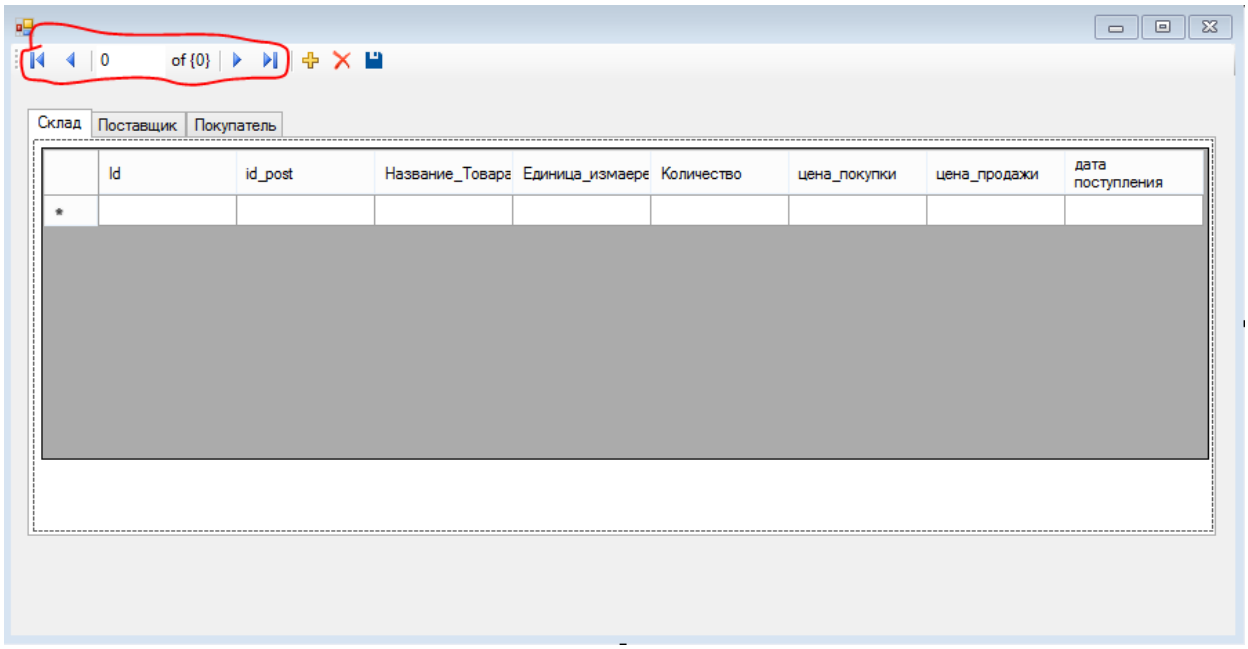
Рисунок 3.14 – Окно запуска

Нажав на вкладки можно увидеть записи наших соответствующих таблиц:

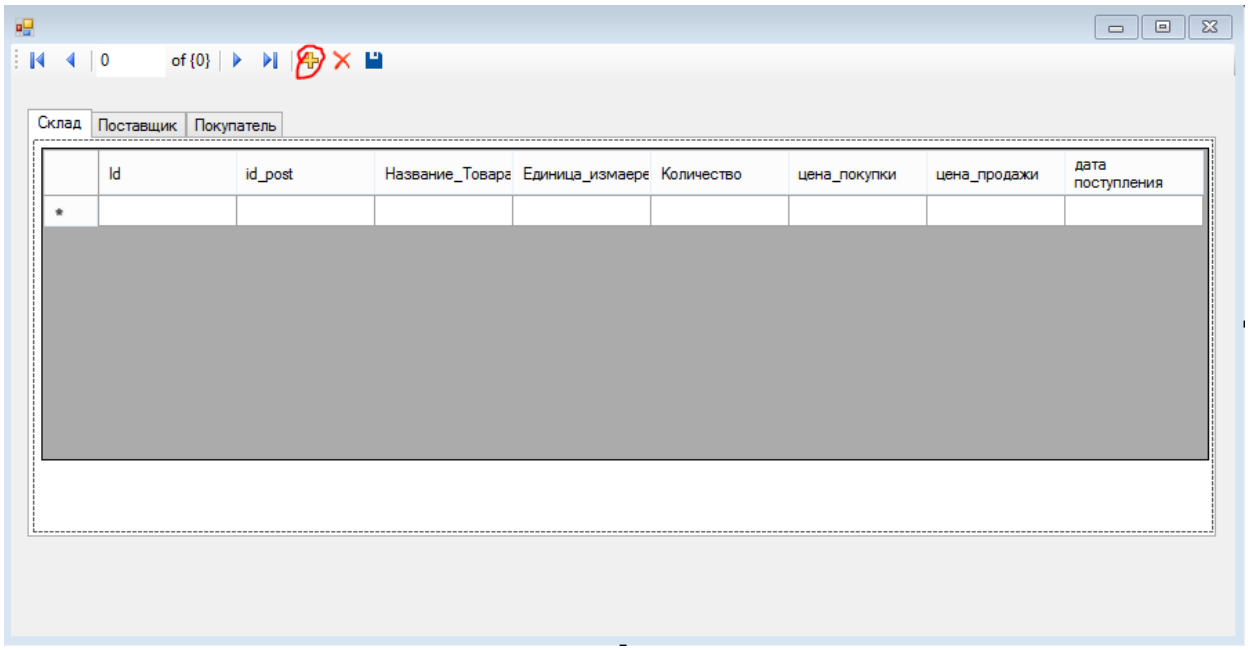




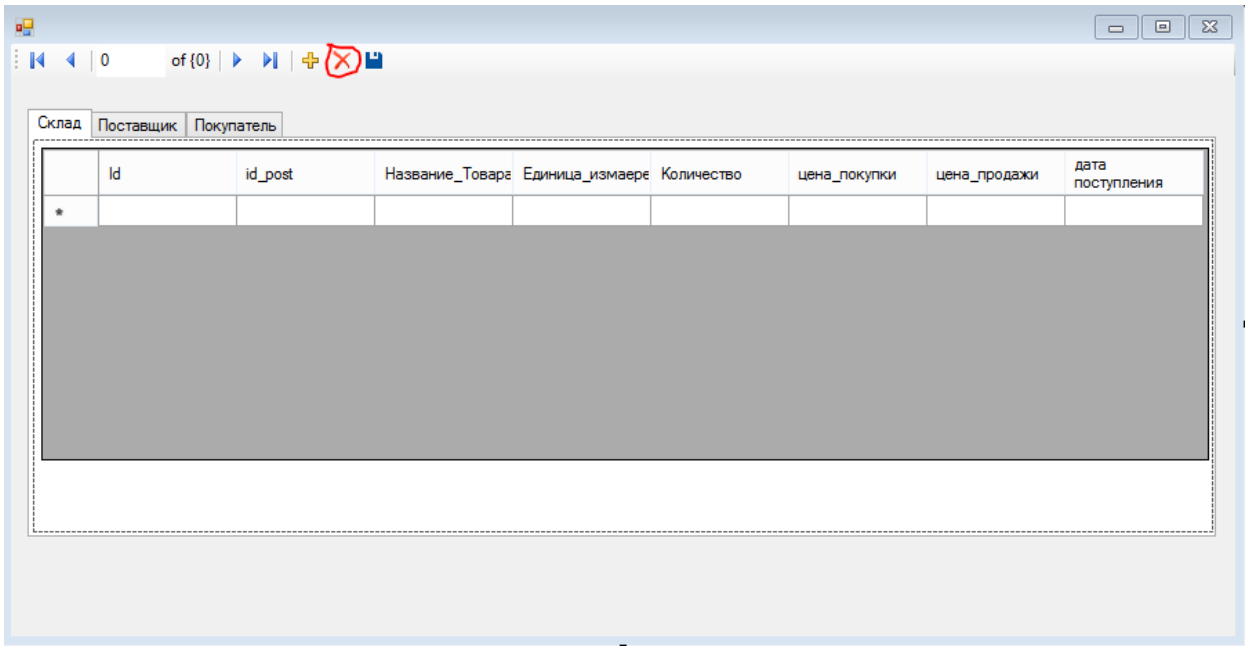
Нажав на кнопки стрелок на верху окна программы можно пройтись по записям
таблицы.



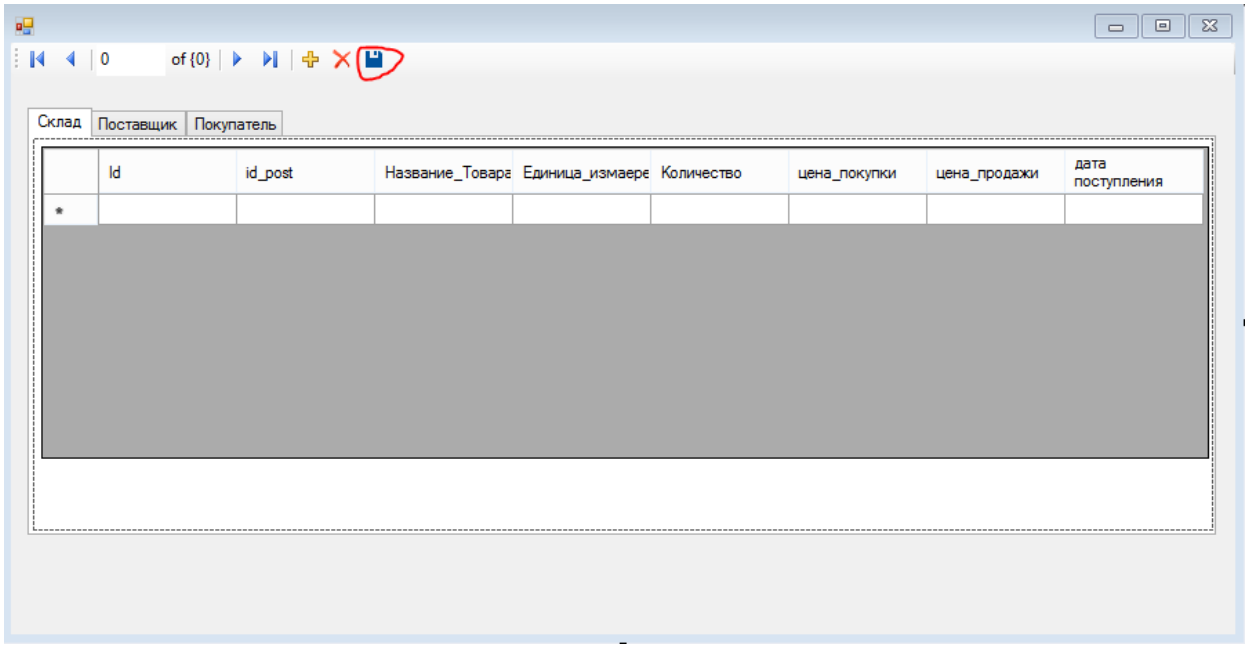
Нажав на кнопко с иконкой «+» можно добавить новые записи в таблицы базы данных:



Нажав на кнопко с иконкой «Х» можно удалить записы из таблицы базы данных:



Нажав на кнопко с иконкой «Дискеты» можно сохранить изменения в таблицах:



ГЛАВА 4. ОРГАНИЗАЦИЯ БЕЗОПАСНОСТИ ЖИЗНИДЕЯТЕЛЬНОСТИ

4.1 Значение и задачи безопасности труда

Безопасность жизнедеятельности – это система научных знаний о сохранении жизни, здоровья и обеспечении безопасности человека при любых видах его деятельности и в любой сфере обитания.

Вопросы обеспечения безопасности труда на производстве, сохранения здоровья и жизни людей всегда занимали значительное место в трудовом законодательстве и внимании к ним, по мере развития общества, постоянно возрастало.

. В документе “обеспечения безопасности труда сотрудников” установлены гарантии осуществления прав трудящихся на охрану труда и обеспечивает единый порядок регулирования отношений в области охраны труда между работодателем и работниками на предприятиях, в учреждениях и организациях всех форм собственности независимо от сферы хозяйственной деятельности и ведомственной подчиненности и направленные на создание условий труда, отвечающих требованиям сохранения жизни и здоровья работников в процессе трудовой деятельности и в связи с ней. Охрана труда - это система обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающая трудовые, социально-экономические, санитарно-гигиенические, организационно-технические и иные мероприятия.

Таким образом, в качестве основных задач безопасности жизнедеятельности можно выделить:

1. ***идентификация опасных и вредных факторов среды;***
2. ***регламентация их уровней в допустимых пределах;***
3. ***разработка комплекса мероприятий по их устранению или защите человеческого организма от негативного из воздействия.***

Согласно «Основы законодательства об охране труда» сказано: «Для организации работы по охране труда на предприятии создается в случае необходимости службы охраны труда или привлекаются специалисты по охране труда на договорной основе».

4.2 Характеристика условий и безопасности труда

Состояние условий труда в Открытом акционерном обществе рассмотрено на примере бухгалтерии (таблица 1).

Из полученной таблицы можно сделать вывод о том, что показатели микроклимата (температура воздуха, относительная влажность, скорость движения воздуха) на рабочих местах бухгалтерии находятся в пределах нормы, сочетание таких микроклиматических условий при длительном и систематическом воздействии на человека сохраняют его нормальное тепловое состояние без напряжения механизма терморегуляции.

Таблица 1

Характеристика параметров условий труда и безопасности на рабочих места

Параметр	Ед. измерен.	Норма	Ф акт
1.Температура а) теплый период года б) холодный и переходной периоды	 о С о С	 20-23 15-23	 2 5 2 0
2.Относительная влажность	%	не более 75	6 0
3.Скорость движения воздуха	м\с	0,2-0,3	0 ,2
4.Освещенность рабочих мест: искусственная естественная	 АК КЭО,%	 300 1,5	 3 00 1 ,5
5.Производственный шум	дБА	60	Ф тс.
6.Вибрация на рабочих местах	дБ	92	Ф тс.

7.Численность рабочих	Чел.		4
8.Технические средства безопасности: зануление предохранители сигнализация			Е сть Е сть Е сть
9.Площадь помещения на 1 человека	М2	6	5 ,3
10.Объем помещения на 1 человека	М3	20	1 8,7

Одним из важнейших элементов условий труда является освещение. Правильно выполненная система освещения повышает общую работоспособность, создает нормальные условия труда. При освещении бухгалтерии используется естественное освещение, создаваемое светом солнца, его дополняют рабочим комбинированным освещением, т.е. сочетанием общего освещения с местным. При общем освещении светильники размещаются в верхней зоне помещения равномерно, местное освещение позволяет получить концентрирующий световой поток непосредственно на рабочей поверхности.

В бухгалтерии имеется защитное заземление, устраняющее опасность поражения людей электрическим током при появлении напряжения на конструктивных частях электрооборудования, т.е. при замыкании на корпусе, а также устройства защитного отключения, обеспечивающее автоматическое отключение опасного участка цепи.

Применение автоматических средств обнаружения пожаров является одним из основных условий обеспечения пожарной безопасности, т.к. позволяет оповестить дежурный персонал о пожаре и месте его возникновения. В бухгалтерии установлены тепловые извещатели, срабатывающие при определенной максимальной температуре.

Среди гигиенических проблем современности проблемы гигиены труда пользователей ПЭВМ относятся к числу наиболее актуальных, поскольку непрерывно расширяется круг задач, решаемых ПЭВМ, и все большие

контингенты людей вовлекаются в процесс использования вычислительной техники.

Совокупность изменений, наблюдаемых в состоянии здоровья профессиональных пользователей ПЭВМ, включает заболевания опорно-двигательного аппарата, органов зрения, центральной нервной и сердечно-сосудистой системы, желудочно-кишечного тракта, аллергические расстройства, отмечают осложнения беременности и родов, неблагоприятное влияние на плод. Получены данные о повышенном уровне онкологических заболеваний.

По мере накопления новых данных по рассматриваемой проблеме становятся все более очевидными причинно-следственные связи между условиями труда и состоянием здоровья пользователей ПЭВМ. Так заболевания опорно-двигательного аппарата (рук, шеи, плечевого пояса, спины) связанны с вынужденной рабочей позой, гиподинамией в сочетании с монотонностью труда. Часто на рабочих местах отсутствует специализированная мебель и с эргономических позиций организация рабочих мест неудовлетворительна. Характерной особенностью труда за компьютером является необходимость выполнения точных зрительных работ на светящемся экране в условиях перепада яркостей в поле зрения, наличии мельканий, неустойчивости и нечеткости изображения. Объекты зрительной работы находятся на разном расстоянии от глаз пользователя (от 30 до 70 см) и приходится часто переводить взгляд в направлениях экран-клавиатура-документация (согласно хронометражным данным от 15 до 50 раз в минуту). Частая пере адаптация глаза к различным яркостям и расстояниям является одним из главных негативных факторов при работе с дисплеями. Неблагоприятным фактором световой среды является несоответствие нормативным значениям уровней освещенности рабочих поверхностей стола, экрана, клавиатуры. Нередко на экранах наблюдается зеркальное отражение источников света и окружающих предметов. Все выше изложенное

затрудняет работу и приводит к нарушениям основных функций зрительной системы.

Труд оператора ПЭВМ относится к формам труда с высоким нервно-эмоциональным напряжением. Это обусловлено необходимостью постоянного слежения за динамикой изображения, различения текста рукописных и печатных материалов, выполнением машинописных и графических работ. В процессе работы требуется постоянно поддерживать активное внимание.

Сейчас уже очевидно, что компьютерные технологии являясь великим достижением человечества, имеют отрицательные последствия для здоровья людей. На сегодня стоит задача снизить ущерб от вреда здоровью. Для этого необходимо соблюдение установленных гигиенических требований к режимам труда и организации рабочих мест. Профессиональные пользователи ВДТ и ПЭВМ должны проходить обязательные предварительные при поступлении на работу и периодические медицинские осмотры. Беременные женщины не допускаются к выполнению работ, связанных с ВДТ и ПЭВМ. Необходимо использовать уже имеющиеся разработки по профилактике нарушений в состоянии здоровья работающих.

Кроме перечисленных факторов на рабочем месте операторов могут иметь место шум, нарушенный ионный режим, неблагоприятные показатели микроклимата. В воздухе могут содержаться химические вещества (озон, фенол, стирол, формальдегиды и др.), что наблюдается при установке на малых площадках большого числа компьютеров и несоблюдении требований к организации рабочих мест.

4.3 Пожарная профилактика

Ответственных за пожарную безопасность отдельных территорий, зданий сооружений и т.п. определяет руководитель предприятия.

На предприятии установлен соответствующий их пожарной опасности противопожарный режим, в том числе:

оборудованы места для курения;

определен порядок обесточивания электрооборудования в случае пожара;

определены порядок и сроки прохождения противопожарного инструктажа.

В административном здании исследуемого предприятия на каждом этаже размещаются по два ручных огнетушителя. Емкости для песка, входящие в конструкцию пожарного стенда, имеют вместительность 0,1 куб м и более.

На предприятии определено лицо, ответственное за приобретение, ремонт, сохранность и готовность к действию первичных средств пожаротушения. Каждый огнетушитель, устанавливаемый на объекте, имеет порядковый номер, нанесенный на корпус белой краской, на него заведен паспорт по установленной форме. На предприятии в основном используются пенные огнетушители.

4.4 Режимы труда и отдыха при работе с персональными компьютерами

Режимы труда и отдыха при работе с ПЭВМ и ВДТ в Техническом университете не контролируются, хотя должны организовываться в зависимости от вида и категории трудовой деятельности.

Виды трудовой деятельности разделяются на 3 группы:

группа А - работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;

группа Б - работа по вводу информации;

группа В - творческая работа в режиме диалога с ЭВМ.

При выполнении в течение рабочей смены работ, относящихся к разным видам трудовой деятельности, за основную работу с ПЭВМ и ВДТ

следует принимать такую, которая занимает не менее 50% времени в течение рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливается 3 категории тяжести и напряженности работы с ВДТ и ПЭВМ (таблица 2), которые определяются:

для группы А - по суммарному числу считываемых знаков за рабочую смену, но не более 60 000 знаков за смену;

для группы Б - по суммарному числу считываемых или вводимых знаков за рабочую смену, но не более 40 000 знаков за смену;

для группы В - по суммарному времени непосредственной работы с ВДТ и ПЭВМ за рабочую смену, но не более 6 часов за смену.

Таблица 2
Время регламентированных перерывов

Категория работы с ВДТ или ПЭВМ	Уровень нагрузки за рабочую смену при видах работ с ВДТ			Суммарное время регламентированных перерывов, мин.	
	группа А, количество знаков	группа Б, количество знаков	группа В, час.	при 8-ми часовой смене	при 12-ти часовой смене
I	до 20000	до 15000	до 2,0	30	70
II	до 40000	до 30000	до 4,0	50	90
III	до 60000	до 40000	до 6,0	70	120

Для инженеров, обслуживающих учебный процесс в аудиториях с ВДТ и ПЭВМ, продолжительность работы не должна превышать 6 часов в день. Продолжительность обеденного перерыва определяется действующим законодательством о труде и Правилами внутреннего трудового распорядка предприятия организации.

Для обеспечения оптимальной работоспособности и сохранения здоровья профессиональных пользователей, на протяжении рабочей смены

должны устанавливаться регламентированные перерывы. Время регламентированных перерывов в течение рабочей смены следует устанавливать, в зависимости от ее продолжительности, вида и категории трудовой деятельности (таблица 2). Продолжительность непрерывной работы с ВДТ без регламентированного перерыва не должна превышать 2 часов.

При 8-ми часовой рабочей смене и работе на ВДТ и ПЭВМ регламентированные перерывы следует устанавливать:

для 1 категории работ через 2 часа от начала рабочей смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;

для 11 категории работ через 2 часа от начала рабочей смены и через 1,5-2,0 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;

для III категории работ через 1,5-2,0 часа от начала рабочей смены и через 1.5-2 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При 12-ти часовой рабочей смене регламентированные перерывы должны устанавливаться в первые 8 часов работы аналогично перерывам при 8-ми часовой рабочей смене, а в течение последних 4часов работы, независимо от категории и вида работ, каждый час продолжительностью 15 минут.

Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления целесообразно выполнять определённые комплексы упражнений.

С целью уменьшения отрицательного влияния монотонии целесообразно применять чередование операций осмысленного текста и числовых данных (изменение содержания работ), чередование редактирования текстов и ввода данных (изменение содержания работы).

В случаях возникновения у работающих с ВДТ и ПЭВМ зрительного дискомфорта и других неблагоприятных субъективных ощущений, несмотря на соблюдение санитарно-гигиенических, эргономических требований, режимов труда и отдыха следует применять индивидуальный подход в ограничении времени работ с ВДТ и ПЭВМ коррекцию длительности перерывов для отдыха или проводить смену деятельности на другую, не связанную с использованием ВДТ и ПЭВМ.

Работающим на ВДТ и ПЭВМ с высоким уровнем напряженности во время регламентированных перерывов и в конце рабочего дня показана психологическая разгрузка в специально оборудованных помещениях (комната психологической разгрузки).

ЗАКЛЮЧЕНИЕ

В процессе проектирования базы данных можно прийти к выводу, что правильная организация хранения и представления данных является неотъемлемой частью для успешного функционирования базы данных. Огромную роль в построении базы данных играют такие полезные функции, как создание запросов, которые позволяют делать выборку необходимых полей из большой совокупности данных, а также производить арифметические и логические операции над этими полями.

В результате проделанной работы получилась реляционная база данных, поддерживаемая СУБД Microsoft SQL Server Express Edition, в которой содержится 5 таблиц. В этих таблицах содержатся сведения о сущностях предметной области «Складского помещения»: товарах, поставщиках, ценах покупатель и т.д. В каждой из этих таблиц содержится некоторое количество записей, необходимых для проверки работоспособности приложения, разработанного на C# специально для администрирования созданной базы данных.

Это приложение содержит в себе множество уязвимостей в исходном коде. Однако целью данной работы не является идеализировать конечный результат, а лишь глубже изучить технологии проектирования баз данных и разработки приложений для управления их содержимым.

На текущий момент эта цель достигнута, поэтому выполнение данной работы завершено.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ