**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙФЕДЕРАЦИИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПБПУ»)

**Институт среднего профессионального образования**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отделение информационных технологий

Специальность: 09.02.07 Информационные системы и программирование

Дисциплина: МДК 11.01 Технологии разработки и защиты баз данных

Группа: 32919/4

**КУРСОВОЙ ПРОЕКТ**

**на тему:**

**«Проектирование и реализация базы данных магазина игрушек»**

Студент Авдина Марина Александровна

(ФИО) (подпись)

Руководитель Тузова Диана Анатольевна

(ФИО) (подпись)

Оценка

(оценка) (подпись) (ФИО)

Санкт - Петербург

2025

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙФЕДЕРАЦИИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПБПУ»)

**Институт среднего профессионального образования**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отделение информационных технологий

**Задание на курсовое проектирование**

**по МДК 11.01 «Технологии разработки и защиты баз данных»**

Специальность: 09.02.07 Информационные системы и программирование

**студенту** Авдиной М.А. группа\_\_32919/4\_\_\_\_

(фамилия, инициалы)

**Тема:** **Проектирование и реализация** **базы данных магазина игрушек**

**Исходные данные к проекту:**

1. Литература по описанию объекта информатизации
2. Литература по инструментальным средствам разработки прикладных программ
3. Лекции, стандарты.
4. Интернет – сайты по предметной области

**Перечень подлежащих разработке вопросов:**

1. Аналитический обзор программных средств, технологий, аналогов ПП для создания базы данных и интерфейсов пользователей.
2. Проектирование реляционной базы данных (инфологическая и даталогическая модели с указанием связей (с пояснениями) между таблицами).
3. Разработка механизмов управления данными в БД (добавление, удаление и обновление данных).
4. Организация выборки информации из разработанной базы данных (сформулировать не менее 5 запросов всех типов, реализуемых средствами выбранной СУБД).
5. Проработка технологии доступа к базе данных (определение круга пользователей базы данных и оценка возможности разграничения полномочий пользователей).
6. Разработка ПО для решения поставленной задачи.
7. Разработка алгоритмов программных модулей.
8. Тестирование работы приложения.
9. Оформление документации (руководство системного программиста).
10. Оформление пояснительной записки.

**Руководитель Д.А. Тузова**

(подпись, дата) (инициалы, фамилия)

**Задание принял к выполнению:**

**Студент М.А. Авдина**

(подпись, дата) (инициалы, фамилия)

**содержание**

[Введение 4](#_Toc201186200)

[1. Сравнительная характеристика СУБД 5](#_Toc201186201)

[1.1. СУБД Oracle Database 5](#_Toc201186202)

[1.2. СУБД Microsoft Access 6](#_Toc201186203)

[1.3. СУБД Microsoft SQL Server 8](#_Toc201186204)

[1.4. Выбор СУБД 10](#_Toc201186205)

[2. Реализация базы данных 12](#_Toc201186206)

[2.1. Описание предметной области 12](#_Toc201186207)

[2.2. Разработка таблиц в Microsoft SQL Server 14](#_Toc201186208)

[2.3. Реализация запросов в Microsoft SQL Server 18](#_Toc201186209)

[3. Реализация интерфейса базы данных 25](#_Toc201186210)

[3.1. Разграничение прав доступа 25](#_Toc201186211)

[3.2. Функционал формы для БД 27](#_Toc201186212)

[3.3. Функционал формы для работы с БД 30](#_Toc201186213)

[Заключение 34](#_Toc201186214)

[Список использованных источников 35](#_Toc201186215)

[Приложение А 36](#_Toc201186216)

# **ВВЕДЕНИЕ**

В современном мире информации наличие эффективных и надежных систем управления базами данных (СУБД) становится все более важным аспектом для успешного функционирования бизнеса.

Основной целью данной курсовой работы является создание базы данных для магазина игрушек, которая бы обеспечивала удобное хранение и обработку данных, а также способствовала бы оптимизации внутренних бизнес-процессов. В процессе работы над проектом была произведена сравнительная характеристика нескольких популярных СУБД с целью выбора, наиболее подходящей для данной задачи.

Актуальность предложенной базы данных заключается в необходимости быстрого и удобного доступа к информации о товарах, клиентах и продажах магазина.

Для реализации интерфейса и функционала был использован Windows Forms, что позволило создать интуитивно понятное и доступное решение для пользователей.

В результате проведенной работы был разработан прототип базы данных, который способен эффективно поддерживать операции магазина и улучшать качество обслуживания клиентов.

1. **СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД**
   1. **СУБД Oracle Database**

Что такое Oracle база данных?

Oracle Database — это одна из самых мощных и надёжных систем управления базами данных (СУБД), признанная во всём мире стандартом для хранения, обработки и защиты больших объёмов информации.

Немного истории:

Компания Oracle Corporation выпустила первую коммерческую версию своей СУБД в 1979 году. С тех пор Oracle Database прошла долгий путь от простой реляционной базы данных до полноценной экосистемы, способной обслуживать высоконагруженные, распределённые и облачные решения.

Сегодня Oracle Database используется в крупнейших мировых компаниях — от банков до авиакомпаний, от государственных органов до стартапов, строящих свои системы с расчётом на будущее масштабирование.

Ключевые особенности Oracle БД:

* Надёжность хранения данных

Oracle обеспечивает целостность данных даже в самых сложных сценариях: сбои питания, ошибки приложений, аварии серверов. Механизмы восстановления и резервного копирования входят в стандартный функционал.

* Высокая производительность

Системы на базе Oracle способны обрабатывать миллионы транзакций в секунду. Благодаря оптимизированной архитектуре, поддержке параллельной обработки запросов и мощной системе индексации, производительность остаётся высокой даже при работе с огромными объёмами данных.

* Поддержка сложной бизнес-логики

Благодаря встроенному языку PL/SQL, разработчики могут создавать сложные процедуры, функции и триггеры прямо внутри базы данных, минимизируя необходимость внешней бизнес-логики.

* Масштабируемость

Oracle Database одинаково эффективно работает как на одном сервере, так и в распределённых облачных инфраструктурах. Решения Oracle легко масштабируются горизонтально и вертикально в зависимости от роста компании.

* Безопасность мирового уровня

Oracle Database предлагает широкий набор инструментов безопасности:

* шифрование данных;
* аудит действий пользователей;
* разграничение прав доступа;
* защита от sql-инъекций и других видов атак.

Где применяется Oracle Database?

* Банковский сектор: обработка транзакций, хранение данных клиентов, системы кредитного скоринга.
* Электронная коммерция: учёт заказов, управление каталогами товаров, аналитика.
* Здравоохранение: хранение медицинских карт, интеграция систем диагностики.
* Государственные проекты: налоговые базы данных, системы электронных документов.
* Телекоммуникации: биллинг, CRM-системы.

Почему стоит изучать Oracle БД?

* Высокий спрос на специалистов. Компании активно ищут разработчиков и администраторов, которые умеют работать с Oracle Database.
* Карьерные перспективы. Знание Oracle открывает путь к позициям от Junior Developer до Senior Architect и Database Administrator (DBA).
* Отличная база для дальнейшего роста. Изучение архитектуры Oracle помогает лучше понимать принципы построения масштабируемых, безопасных и надёжных приложений.

Заключение:

Oracle Database — это не просто СУБД, а основа для построения надёжных, высокопроизводительных и безопасных информационных систем. Если вы хотите строить карьеру в сфере работы с базами данных или развиваться как разработчик корпоративных приложений, знание Oracle Database станет вашим мощным конкурентным преимуществом.

* 1. **СУБД Microsoft Access**

MS Access – наиболее распространенная система управления базами данных, входящая в состав пакета Microsoft Office. Программа предназначена для создания и дальнейшего хранения больших массивов данных. Информация хранится в виде таблиц, связанных между собой с помощью ключевых полей.

MS Access позволяет хранить в базе девять основных типов данных, в том числе и медиафайлы, гиперссылки. На основе созданных в MS Access баз данных, можно создавать отчеты, формировать выборки и быстро получать доступ к необходимой информации.

Среди существенных недостатков программы стоит выделить тот момент, что она совместима только с устройствами, работающими на базе операционной системы Windows.

Базы данных позволяют создавать таблицы, в которых хранятся большие массивы данных. С помощью MS Access, можно установить связи между отдельными таблицами, определив ключевые строки в каждой из них. При этом каждое ключевое поле должно содержать уникальную информацию, которая позволяет четко идентифицировать строку.

MS Access – удобный инструмент для дальнейшей обработки информации. С помощью программы, можно выполнять поиск данных в таблицах, сортировать их в зависимости от целей, выполнять фильтрацию и даже производить вычисления.

Говоря о возможностях MS Acess, нельзя не отметить такие как:

* использование форм, с помощью которых можно значительно облегчить внесение или редактирование информации при вводе;
* хранение данных в удобном виде;
* поиск необходимой информации с помощью фильтров или запросов;
* представление данных в удобном для пользователя виде с помощью сформированных отчетов;
* возможность автоматизации работы с информацией;
* использование макросов для быстрой и удобной обработки данных.

Готовые базы данных в дальнейшем можно подключать к сайту, предоставляя доступ к той или иной информации, а также использовать при создании сложных программ или приложений.

При этом важно помнить, что структура и содержание таблиц, связей между ними должно быть тщательно продумано. В противном случае, можно получить продукт, который будет работать не совсем корректно.

MS Access применяется в различных сферах деятельности – в бизнесе, экономике, программировании.

В бизнесе.

С помощью созданных в MS Access баз данных, можно вести бухгалтерский учет, не используя при этом сторонние приложения, хранить информацию о клиентах и их заказах. Достаточно часто MS Access используют для учета товаров в магазинах.

На предприятиях.

MS Access позволяет создавать справочники, содержащие всю необходимую информацию о сотрудниках организации – их паспортных данных, месте проживания, обязанностях и даже их фото.

Применяют MS Access и в университетах и колледжах, школах, формируя на основе программы базы данных поступающих и уже обучающихся студентов.

В ряде случаев MS Access используют для организации документооборота.

В программировании.

При разработке сложных программ зачастую используются базы данных, хранящие необходимые наборы сведений и данных. Учитывая простоту работы с MS Access, именно ее зачастую используют при создании новых продуктов.

К примеру, при разработке программного обеспечения в сфере лингвистики, удобно организовать хранение словарей и корпусов данных с помощью MS Access.

Персональное использование.

Продвинутые пользователи могут применять базы данных для хранения личной информации. К примеру, хранить контактные данные, создавать собственную базу личной библиотеки или аудиотеки.

Не стоит забывать и том, что база данных позволяет хранить не только числовую и текстовую информацию, но и медиафайлы.

* 1. **СУБД Microsoft SQL Server**

MS SQL Server появилась ещё в конце 1980-х и с тех пор держится в топе самых популярных СУБД в мире. Она быстрая, удобная и подходит почти для всего: от небольших сайтов до высоконагруженных веб-приложений.

MS SQL Server — это система управления реляционными базами данных, работающая по клиент-серверной модели. Прозвучало много новых слов — сейчас разберём по порядку, что это всё значит.

«Клиент-серверная» означает, что вся логика работы с данными вынесена на сервер, а пользователи подключаются к нему через приложения-клиенты. Идея в том, что нам не надо хранить полную версию базы данных у себя на компьютере — достаточно просто иметь хороший интернет.

Более того, экосистему SQL Server можно развернуть и в облаке: в этом случае обслуживание базы данных берут на себя дата-центры Microsoft, а вы можете сосредоточиться на разработке клиентских приложений. Версия SQL Server, которая позволяет так делать, называется Azure SQL.

Так как SQL Server — это СУБД от Microsoft, она заточена в основном на работу с Windows, хотя частично поддерживает и Linux. Частично — потому что в версии для этой системы нет некоторых важных функций. Поэтому, если ищете СУБД для Linux-машины, лучше дополнительно поизучать конкурентов.

Под капотом у SQL Server лежит движок Database Engine — это ядро СУБД, которое и отвечает за всю работу с данными. Именно движок принимает SQL-запросы от пользователей и распределяет их между ресурсами сервера. Технически он выглядит как служба в операционной системе — sqlservr.exe.

А ещё на один сервер можно установить несколько экземпляров SQL Server, и тогда движков тоже будет несколько — и каждый будет работать с каким-то своим набором баз данных. Например, на сервере Skillbox один движок может заниматься базой статей в Skillbox Media, а другой — онлайн-курсами.

Теперь заберёмся чуть поглубже и посмотрим, как именно SQL Server работает с клиентскими запросами. Допустим, мы зашли на маркетплейс, чтобы купить футболку — но не простую, а поло 48 размера бренда «Школьный вальс». Выставляем фильтры, нажимаем «Применить», а дальше — следим за руками:

* браузер превращает наши фильтры в запрос и отправляет на сервер;
* там наш запрос встречает движок sql server, создаёт на его основе новую задачу (task) и переводит её в режим ожидания;
* далее задачу подхватывает рабочий поток (worker), анализирует содержимое запроса и составляет план: как его выполнить максимально быстро и не затратно по ресурсам;
* потом в игру вступает планировщик задач (task scheduler) — он следит за тем, чтобы рабочие потоки эффективно расходовали ресурсы процессора и не мешали друг другу;
* только на этом этапе запрос начинает по-настоящему выполняться — операторы sql работают с базой данных и извлекают из неё нужные данные;
* сервер пуляет данные обратно в браузер, и вот мы видим список нужных футболок — прошла всего секунда, а сколько всего случилось :).

Конечно, это очень упрощённая схема — она не учитывает, например, обработку HTTP-запросов на веб-сервере и некоторые аспекты архитектуры SQL Server.

Преимущества:

Отказоустойчивость**.** Например, в SQL Server можно создавать кластеры серверов, которые будут работать с одним экземпляром СУБД. Если какая-то машина вдруг выйдет из строя, её работу сразу перехватит другая, и всё это случится абсолютно бесшовно и незаметно для пользователя. Эта технология называется кластеризацией.

А ещё SQL Server постоянно создаёт контрольные точки, из которых можно восстановить базу данных в случае сбоя или повреждения данных.

Совместимость с Microsoft**.**Естественно, SQL Server тесно интегрирован в экосистему «оконных» приложений — например, с помощью расширения mssql можно управлять базами данных из IDE Visual Studio.

Машинное обучение.С помощью SQL Server можно обучать ML-модели прямо внутри базы данных. Для этого там есть специальная служба, которая умеет выполнять скрипты для работы с big data на Python и R. А если к этому делу подключить какой-нибудь TensorFlow, то там и до своего ChatGPT недалеко :)

Бизнес-аналитика. Можно делать интерактивные дашборды на основе данных с помощью встроенной службы отчётов, а можно выгрузить начинку базы данных в сторонний сервис — тот же майкрософтовский PowerBI.

Удобная студия управления.Считается, что SQL Server Management Studio — это одна из лучших сред разработки баз данных из существующих.

Большое сообщество. SQL Server — вторая по популярности СУБД в мире, и в интернете по ней море учебных материалов, блогов и сообществ. Энтузиасты даже собрали все обучающие ресурсы в один большой гайд. Похожий, кстати, есть на «Хабре» — довольно старый, но с основами разобраться поможет.

* 1. **Выбор СУБД**

Почти не используем и плохо знакомы с СУБД Oracle Database в работе и поэтому не будем использовать его в курсовом проекте.

С Microsoft Access знакомы достаточно, однако часто возникают сложности в составлении запросов, что не происходит в Microsoft SQL Server. Microsoft SQL Server предлагает высокую производительность обработки запросов и может эффективно справляться с большими объемами данных. Это делает его идеальным выбором для проектов, которые могут требовать масштабирования в будущем. В отличие от Microsoft Access, который лучше подходит для небольших приложений, SQL Server способен обрабатывать большие объемы транзакций и поддерживать множество пользователей одновременно.

Планируем использовать среду разработки Visual Studio 2022 для создания графического интерфейса работы с БД. SQL Server будет наиболее удобной в работе с средой разработки. Это позволяет легко разрабатывать приложения и управлять базами данных в единой экосистеме. Сравнительно с Oracle Database, которая может требовать больше усилий для интеграции с другими инструментами, SQL Server предлагает более удобные решения.

SQL Server поддерживает полные транзакции с возможностью отката (rollback) и подтверждения (commit), что обеспечивает надежность операций с данными. Это особенно важно для приложений, которые требуют высокой степени надежности и консистентности данных. В то время как Microsoft Access имеет ограничения по работе с транзакциями в многопользовательской среде, SQL Server справляется с этой задачей гораздо лучше.

SQL Server постоянно обновляется и включает в себя современные функции, такие как поддержка JSON, улучшенные механизмы работы с временными рядами и интеграция с облачными сервисами. Это делает его актуальным выбором для современных приложений.

1. **РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ**
   1. **Описание предметной области**

Магазин игрушек — это торговая точка, специализирующаяся на продаже различных видов игрушек для детей и коллекционеров. В ассортимент магазина могут входить как традиционные игрушки (куклы, машинки, конструкторы), так и современные интерактивные и образовательные игрушки. Магазин может работать как в физическом формате, так и в онлайн-режиме.

Предусмотрены функциональные возможности сортировки данных, что существенно упрощает управление информацией и повышает эффективность работы с базой. Система позволяет сортировать данные по различным критериям, таким как название товара, цена, страна производителя и так далее. Это дает возможность быстро находить нужные данные и анализировать ситуацию в магазине.

Кроме того, база данных поддерживает поиск по ключевому полю, что позволяет пользователям мгновенно находить конкретные записи, используя уникальные идентификаторы или другие значимые атрибуты. Например, администратор может быстро просмотреть сколько заказов было оформлено за указанную дату

Сущности базы данных:

* + Товары: информация о каждом товаре, включая название, описание, цену, страну производителя, количество остатков товара на складе и изображения.
  + Страны: информация о странах, которые производят игрушки для магазина, а также их валюту
  + Заказы: информация о всех произведённых заказах. Включает в себя данные о покупателе, магазине, в котором произошел заказ, дате заказа, сумме заказа, приобретённых товарах и работнике, который обслуживал покупателя
  + Покупатели: содержит данные о людях, которые когда-либо пользовались услугами магазина, их имя, отчество и номер мобильного телефона
  + Магазины: содержит информацию о каждой точке из сети магазина игрушек, включая адрес, контактный телефон и рейтинг на яндекс картах
  + Работники: содержит данные о работниках со всех точек сети. Включает в себя данные об имени работника, его фамилии, номера телефона, id магазина, в котором он работает, а также должность

Связи сущностей в базе данных выглядят следующим образом:

* Таблица страны соединена с таблицей товары связью 1 ко многим по ключевому полю id из стран с полем страны из таблицы товары. Одна страна может выпускать множество товаров.
* Таблица товары соединена с таблицей заказы связью 1 ко многим по ключевому полю id из товаров с полем товары из таблицы заказы. Один вид товара могут приобретать множество покупателей.
* Таблица покупатели соединена с таблицей заказы связью 1 ко многим по ключевому полю id из покупателей с полем покупатель из таблицы заказы. На одного покупателя может быть оформлено множество заказов
* Таблица работники соединена с таблицей заказы связью 1 ко многим по ключевому полю id из таблицы работники с полем работник из таблицы заказы. Один работник может принять множество заказов
* Таблица магазины соединена с таблицей работники связью 1 ко многим по ключевому полю id из магазинов с полем магазин из таблицы работники. В одном магазине может работать несколько работников

В базе данных есть 2 уровня доступа для 2 ролей:

* Администратор: может просматривать данные из всех таблиц, добавлять новые записи и удалять старые, осуществлять поиск и сортировку данных по ключевым полям
* Работник: имеет доступ только к просмотру всех таблиц

Для того чтобы лучше продемонстрировать структуру БД магазина игрушек, была составлена полная ER-Модель (Рисунок 1).

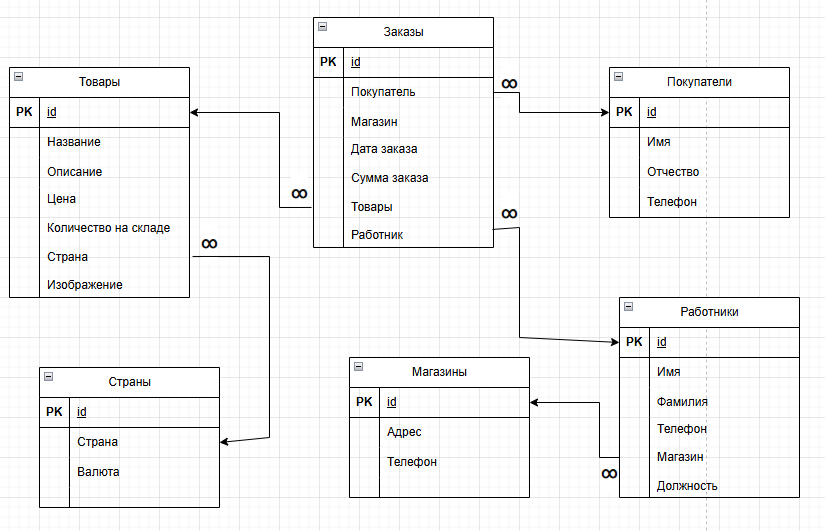


Рисунок 1 - ER-модель БД магазина игрушек

* 1. **Разработка таблиц в Microsoft SQL Server**

Создание таблиц в SQL для базы данных магазина игрушек — это важный шаг в проектировании структуры данных, который позволит эффективно управлять информацией о товарах, клиентах и заказах.

Используя интерфейс СУБД Microsoft SQL, были воссозданы все сущности и связи, которые были описаны в предметной области и представлены на ER-модели (Рисунок 2 – 9).

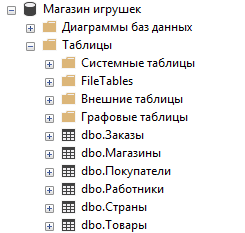


Рисунок 2 - База данных и её сущности

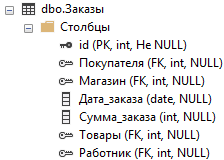


Рисунок 3 - Атрибуты сущности «Заказы»

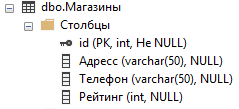


Рисунок 4 - Атрибуты сущности «Магазины»

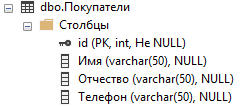


Рисунок 5 - Атрибуты сущности «Покупатели»

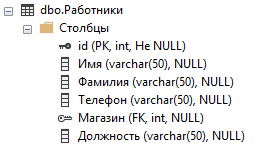


Рисунок 6 - Атрибуты сущности «Работники»

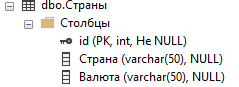


Рисунок 7 - Атрибуты сущности «Страны»

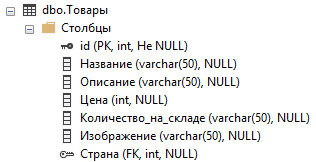


Рисунок 8 - Атрибуты сущности «Товары»

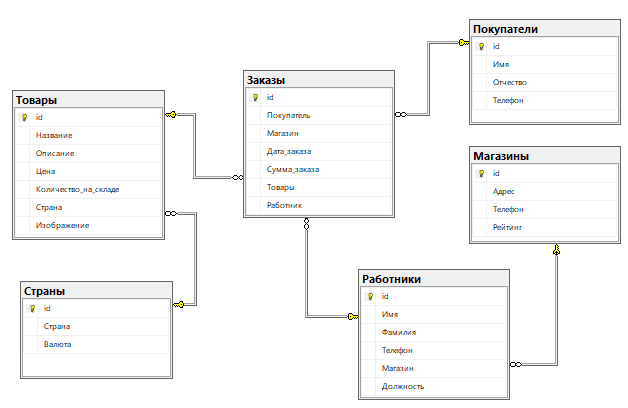


Рисунок 9 - Диаграмма базы данных

Для поля id в каждой таблице укажем свойство идентификатора, который будет автоматически пронумеровывать данные начиная с 1 и с шагом 1. Заполним все поля таблиц тестовыми данными (Рисунок 10 - 16)

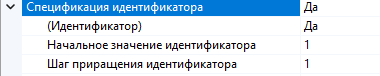


Рисунок 10 - Идентификатор включен

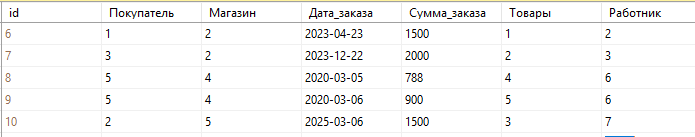


Рисунок 11 - Данные в таблице «Заказы»

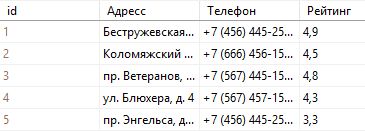


Рисунок 12 - Данные в таблице «Магазины»

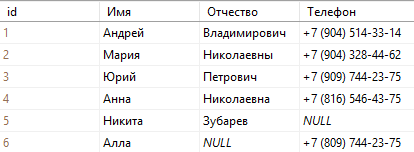


Рисунок 13 - Данные в таблице «Покупатели»

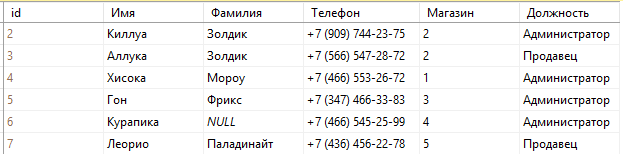


Рисунок 14 - Данные в таблице «Работники»



Рисунок 15 - Данные в таблице «Страны»

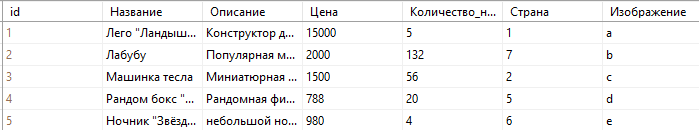


Рисунок 16 - Данные в таблице «Товары»

Созданные таблицы обеспечивают основную функциональность для управления данными о товарах, клиентах, заказах и т.д., в базе данных магазина игрушек. Создание связей между таблицами через внешние ключи позволяет эффективно организовать данные и выполнять сложные запросы для анализа и отчетности.

* 1. **Реализация запросов в Microsoft SQL Server**

Для обеспечения полного функционала, в базе данных будут использоваться запросы для поиска записи по данным ключевого поля, а также функция сортировки по выбранному полю любой таблицы.

Составим запрос для поиска записей в таблице по ключевому полю. Для примера попробуем найти записи о покупателе с именем Мария (Рисунок 9)

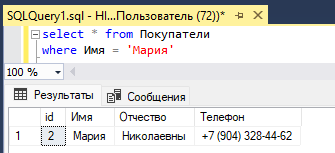


Рисунок 17 - Данные найдены успешно

Далее составим запрос для вывода всех данных из таблицы заказы, которые отсортированы по дате. Для начала просмотрим все данные, а затем произведём сортировку и проследим, как изменился вывод (Рисунок 10 – 11)

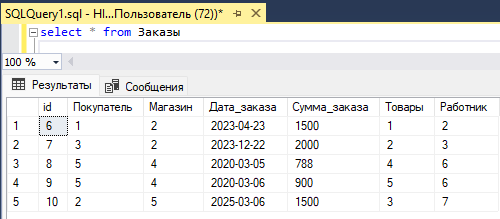


Рисунок 18 - Таблица заказы

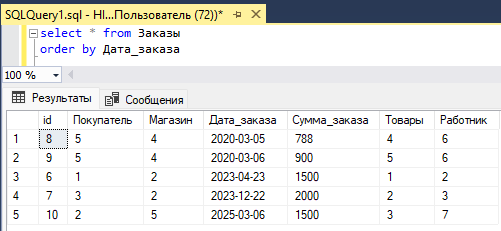


Рисунок 19 - Отсортированная таблица заказы

Как мы видим на рисунке 11, после выполнения запроса на сортировку, данные вывелись в порядке возрастания даты заказа

Так же для полноценной работы с базой данных необходимо продемонстрировать функции удаления и добавления новых записей. Для начала попробуем добавить новую запись в таблицу магазины, для этого в скобках мы прописываем те поля, в которые хотим записать данные: Магазины (Адрес, Телефон, Рейтинг). Здесь не указан столбец id, так как он заполняется автоматически благодаря идентификатору (Рисунок 20 – 21)

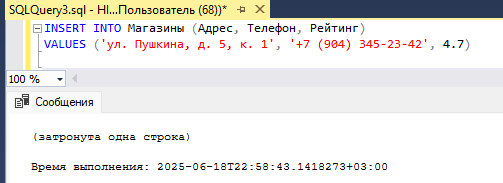


Рисунок 20 - Запрос выполнен успешно

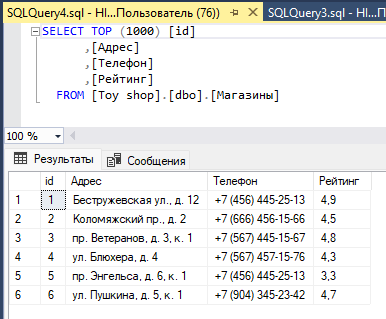


Рисунок 21 - Данные в таблице появились

Теперь попробуем удалить эту же запись по ключевому полю – id (Рисунок 22 – 23)

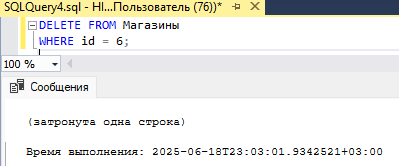


Рисунок 22 - Запрос выполнен

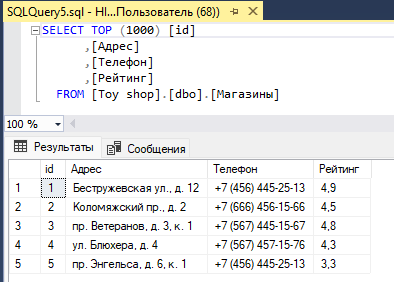


Рисунок 23 - Данные в таблице изменены

Для того чтобы при добавлении нового заказа в таблицу "Заказы" автоматически уменьшалось количество товара в таблице "Товары", можно использовать транзакцию. Это гарантирует, что оба действия (вставка заказа и обновление количества товара) будут выполнены атомарно. Если одно из действий не удастся, то изменения не будут применены. Для примера добавим заказ на игрушку «Лабубу» с id=2 (Рисунок 24 – 26).

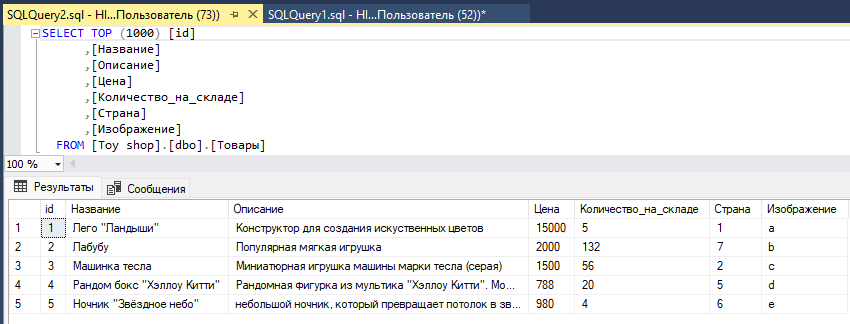


Рисунок 24 - Изначальный вид таблицы товары

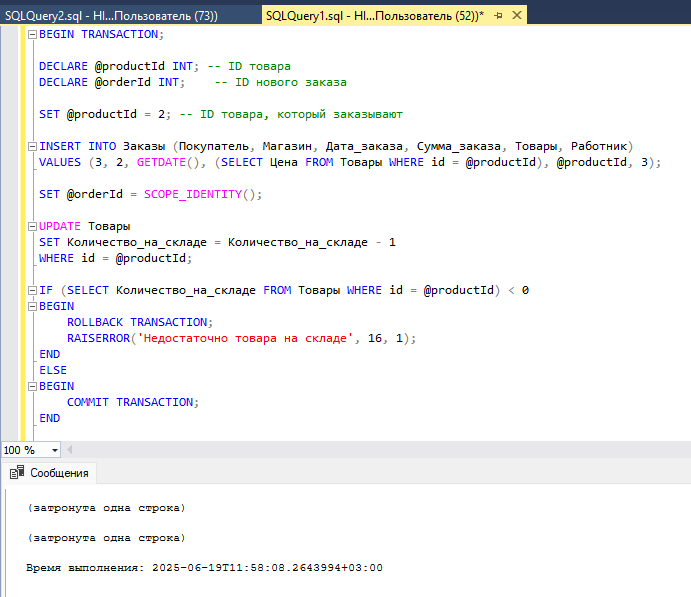


Рисунок 25 - Транзакция успешно проведена

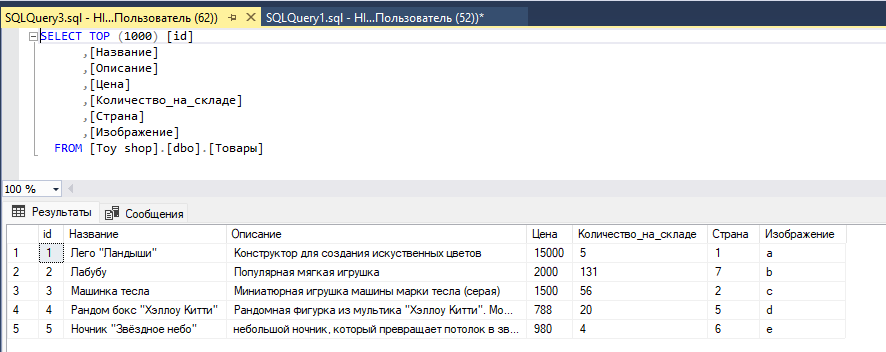


Рисунок 26 - Данные в таблице изменились

В данной транзакции изначально происходит добавления записи в таблицу заказы, затем SET @orderId = SCOPE\_IDENTITY(): получает идентификатор только что вставленного заказа. Потом происходит обновление таблицы с товарами и проверка на отрицательное число запасов на складе. Если после обновления количество товара становится отрицательным, транзакция откатывается, и выводится ошибка.

Создадим хранимую процедуру, которая будет выводить сумму продаж конкретного работника за определённый период времени (Рисунок 27)

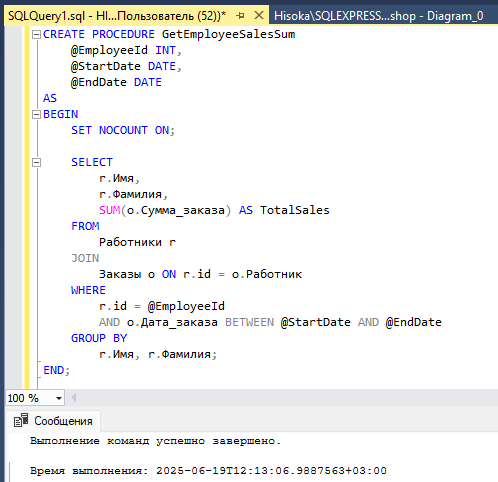


Рисунок 27 - Триггер создан

Протестируем созданный триггер (Рисунок 28 – 29)

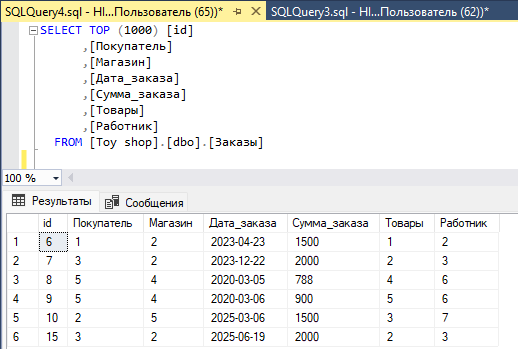


Рисунок 28 - Данные в таблице заказы

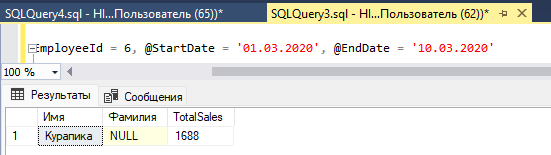


Рисунок 29 - Триггер работает успешно

1. **РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА БАЗЫ ДАННЫХ**
   1. **Разграничение прав доступа**

Создание функции авторизации с двумя уровнями доступа — это распространенная задача, которая может быть реализована различными способами в зависимости от используемой технологии и архитектуры приложения.

В приложении есть два уровня доступа:

* администратор: может просматривать данные из всех таблиц, добавлять новые записи и удалять старые, осуществлять поиск и сортировку данных по ключевым полям;
* работник: имеет доступ только к просмотру всех таблиц.

Был создан проект приложения Windows Forms (.NET Framework) на языке программирования C#. В приложении были созданы 3 формы:

* форма авторизации;
* форма с функционалом администратора;
* форма с функционалом работника.

При вводе логина и пароля администратора, открывается интерфейс с полным функционалом (Рисунок 24 – 25)

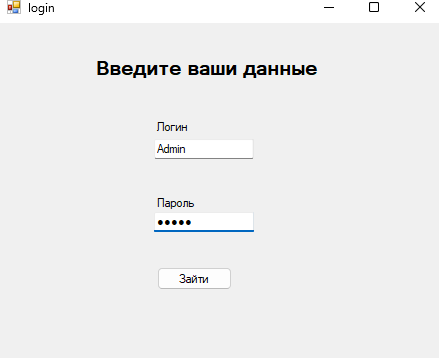


Рисунок 30 - Введены данные администратора

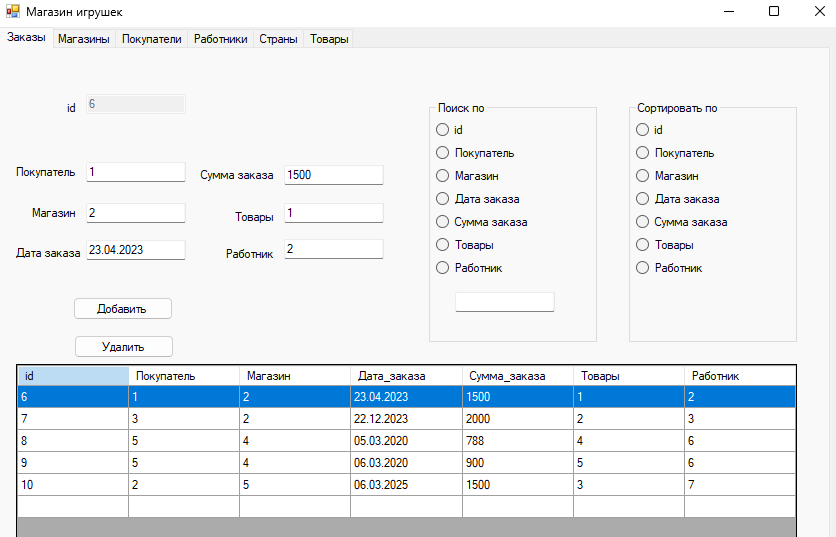


Рисунок 31 - Открыт интерфейс администратора

При перезапуске программы и вводе логина и пароля обычного пароля, появится интерфейс только для просмотра данных из всех таблиц (Рисунок 16 – 27)

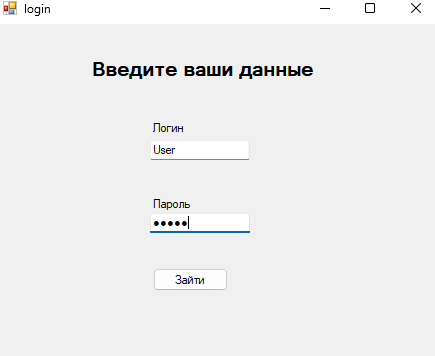


Рисунок 32 – Введены данные обычного работника

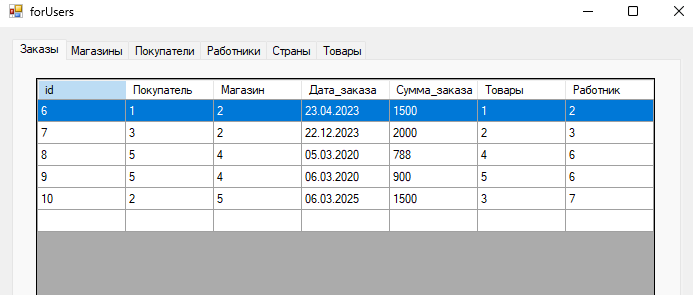


Рисунок 33 - Интерфейс для обычного работника

* 1. **Функционал формы для БД**

В уровне доступа обычного продавца, находится только просмотр записей из всех таблиц, что было представлено на рисунке 27. Поэтому рассмотрим функционал администратора.

При нажатии на запись из таблицы, данные из строки будут отображаться в соответствующих полях на вкладке формы для этой таблицы (Рисунок 28 – 29)

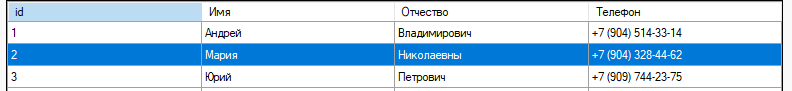


Рисунок 34 - Выбрали запись

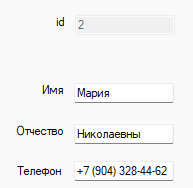


Рисунок 35 - Данные отобразились в полях

Для того чтобы посмотреть на работу функции добавления новой записи, заполним поля новыми данными, а затем нажмём кнопку «Добавить» (Рисунок 30 – 32)

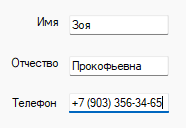


Рисунок 36 - Новые данные введены



Рисунок 37 - Кнопка нажата

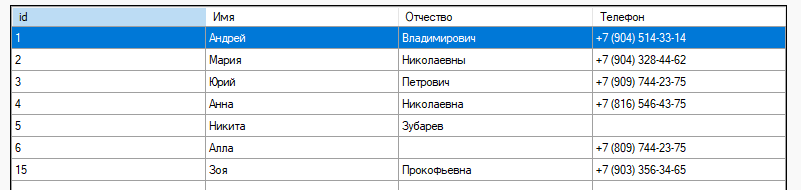


Рисунок 38 - Новые данные появились в таблице

Теперь посмотрим на работу функции удаления. Для этого нажмём на нужную запись в таблице и нажмём кнопку «Удалить» (Рисунок 33 – 35).

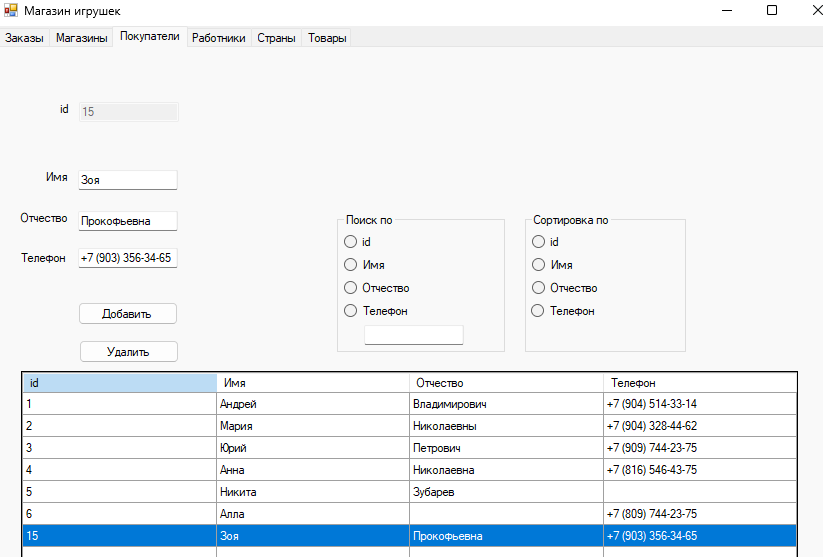


Рисунок 39 - Запись выбрана



Рисунок 40 - Кнопка нажата

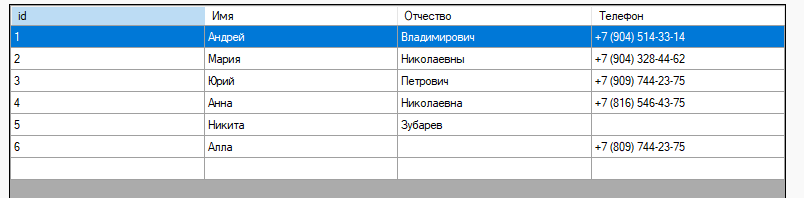


Рисунок 41 - данные успешно удалились

Протестируем функцию поиска всех записей по заданному параметру. Для этого заполним поле из группы «Поиск по» и выберем по какому полю из таблицы будет производиться поиск (Рисунок 36 - 37)

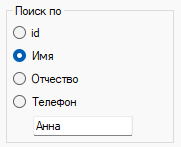


Рисунок 42 - Ввели значение и выбрали поле для поиска

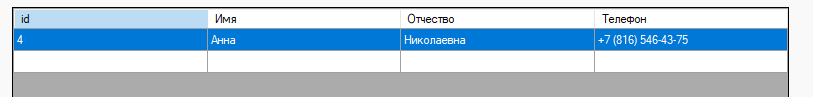


Рисунок 43 - Запись успешно найдена

Теперь рассмотрим функцию сортировки записей по выбранному полю. Для этого выберем любое поле из группы «Сортировать по» (рисунок 38 – 39)

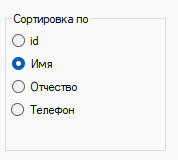


Рисунок 44 - Поле выбрано

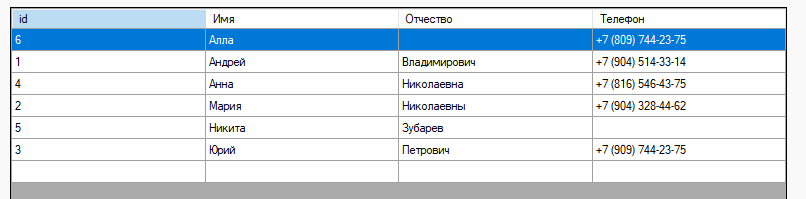


Рисунок 45 - данные успешно отсортированы

* 1. **Функционал формы для работы с БД**

На форме представлены следующие элементы управления:

* TabControl (Рисунок 40)

Описание**:** TabControl — это элемент управления, который позволяет организовать содержимое в виде вкладок. Каждая вкладка может содержать различные элементы управления и данные.

Назначение**:** Удобен для разделения информации на логические группы, что помогает пользователю легче ориентироваться в приложении. Например, можно использовать вкладки для различных категорий настроек или информации. В нашем случае переключение между таблицами базы данных

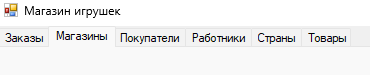


Рисунок 46 - Вид элемента. Выбрана вкладка с магазинами

* TextBox (Рисунок 41)

Описание**:** TextBox — это элемент управления, который позволяет пользователям вводить текстовую информацию. Он может быть однострочным или многострочным.

Назначение**:** используется для ввода данных пользователем, таких как имена, адреса, пароли и т.д. Многострочный текстовый блок может использоваться для ввода комментариев или описаний.

Свойства: для того, чтобы в полях автоматически появлялись данные из таблиц, можно установить свойство (DataBindings). В значении Text нужно выбрать столбец из таблицы, например: магазиныBindingSource - Адресс



Рисунок 47 - Вид элемента

* Button (Рисунок 42)

Описание**:** Button — это элемент управления, который представляет собой кнопку, на которую пользователь может нажать для выполнения определенного действия.

Назначение**:** используется для инициирования событий, таких как отправка формы, выполнение команд или переход к другой части приложения. В нашем случае у нас две кнопки: добавить и удалить



Рисунок 48 - Кнопка добавить

* GroupBox (Рисунок 43)

Описание**:** GroupBox — это контейнер, который позволяет организовать другие элементы управления в логическую группу. Он имеет заголовок и может содержать различные элементы управления внутри.

Назначение**:** упрощает восприятие интерфейса, группируя связанные элементы управления вместе. Например, можно использовать GroupBox для группировки радиокнопок, связанных с одним выбором.

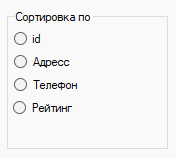


Рисунок 49 - Группа радио-кнопок для сортировки

* RadioButton (Рисунок 44)

Описание**:** RadioButton — это элемент управления, который позволяет пользователю выбрать один вариант из нескольких предложенных. Обычно они используются в группах.

Назначение**:** позволяет пользователю сделать выбор из ограниченного числа вариантов. В группе радиокнопок только одна кнопка может быть выбрана одновременно.



Рисунок 50 - Выбранная и не выбранная радио-кнопки

* DataGridView (Рисунок 45)

Описание**:** DataGridView — это элемент управления, который отображает данные в табличной форме. Он поддерживает функции редактирования, сортировки и фильтрации данных.

Назначение**:** используется для отображения и редактирования данных из базы данных или других источников данных в удобном табличном формате. Это особенно полезно для приложений, работающих с большими объемами информации.

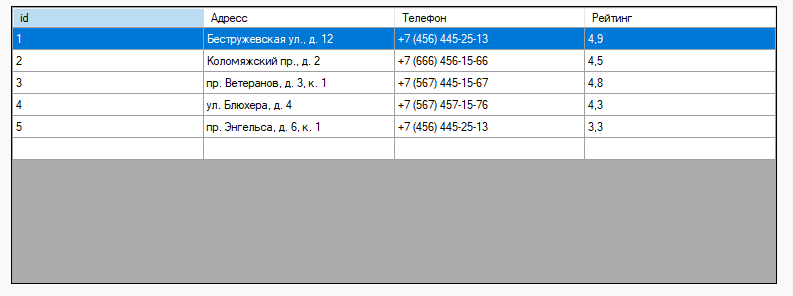


Рисунок 51 - Вид элемента

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была успешно разработана база данных для магазина игрушек, отвечающая современным требованиям по организации и хранению данных. Сравнительный анализ нескольких СУБД позволил выбрать Microsoft SQL Server как наиболее подходящий инструмент для реализации проекта, обеспечивший высокую производительность и надежность. Разработка интерфейса с использованием Windows Forms сделала взаимодействие с пользователем удобным и доступным.

Созданная база данных позволяет эффективно управлять ассортиментом товаров, отслеживать продажи и обрабатывать заказы, что в свою очередь способствует повышению уровня обслуживания клиентов и оптимизации бизнес-процессов магазина.

Предлагаемая система может быть далее доработана и усовершенствована, что откроет новые перспективы для развития магазина и повышения его конкурентоспособности на рынке.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 c.

2 Мюллер, Р.Д. Проектирование баз данных и UML / Р.Д. Мюллер; Пер. с англ. Е.Н. Молодцова. - М.: Лори, 2013. - 420 c.

3 Бьюли, А. Изучаем SQL / А. Бьюли; Пер. с англ. Н. Шатохина. – М.: Символ-Плюс, 2016. – 312 с.

4 Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель, SQL. Полное руководство. / Грофф Джеймс Р., Вайнберг Пол Н., Оппель Эндрю Дж.- М.: Вильямс И.Д., 2018. – 960 c.

5 Бен-Ган, И. Microsoft SQL Server 2012. Создание запросов. Учебный курс Microsoft: Пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. — М.: Издательство «Русская редакция», 2014. — 720 с.

6 Техническая документация по SQL Server [Офиц. сайт]. URL: https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15

7 Windows Forms [Офиц. сайт]. URL: <https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-5.0>

8. Астахова И.Ф. и др. SQL в примерах и задачах. Учебное пособие. Мн. Новое знание, 2010-176 стр.

**ПРИЛОЖЕНИЕ А**

Код дополнительного модуля Class1:

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ToyShop

{

public class Class1

{

SqlConnection con = new SqlConnection("Data Source=HISOKA\\SQLEXPRESS;Initial Catalog=\"Toy shop\";Integrated Security=True");

public void Oreder\_By(object sender, DataGridView dgv, string tabel)

{

RadioButton selectedRadioButton = sender as RadioButton;

SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM {tabel} ORDER BY {selectedRadioButton.Text}", con);

DataTable table = new DataTable();

adapter.Fill(table);

dgv.DataSource = table;

}

public void Search(object sender, DataGridView dgv, string text, string tabel)

{

RadioButton selectedRadioButton = sender as RadioButton;

SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM {tabel} WHERE {selectedRadioButton.Text}='{text}'", con);

DataTable table = new DataTable();

adapter.Fill(table);

dgv.DataSource = table;

}

public void Delete(DataGridView d, string text, string tabel)

{

con.Open();

SqlCommand command = new SqlCommand($"Delete from {tabel} where id=@id", con);

command.Parameters.AddWithValue("@id", text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM {tabel}", con);

DataTable table = new DataTable();

adapter.Fill(table);

d.DataSource = table;

}}}

Код модуля для авторизации:

using System;

using System.Windows.Forms;

namespace ToyShop

{

public partial class login : Form

{

public login()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "User" && textBox2.Text == "12345")

{

forUsers forUsers = new forUsers();

forUsers.Show();

this.Hide();

}

else if (textBox1.Text == "Admin" && this.textBox2.Text == "12345")

{

Form1 forAdmins = new Form1();

forAdmins.Show();

this.Hide();

}

else

{

MessageBox.Show("Неверный логин или пароль.");

}

}

}

}

Код модуля основной формы:  
using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ToyShop

{

public partial class Form1 : Form

{

SqlConnection con = new SqlConnection("Data Source=HISOKA\\SQLEXPRESS;Initial Catalog=\"Toy shop\";Integrated Security=True");

Class1 class1 = new Class1();

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

this.товарыTableAdapter.Fill(this.toy\_shopDataSet.Товары);

this.страныTableAdapter.Fill(this.toy\_shopDataSet.Страны);

this.работникиTableAdapter.Fill(this.toy\_shopDataSet.Работники);

this.магазиныTableAdapter.Fill(this.toy\_shopDataSet.Магазины);

this.заказыTableAdapter.Fill(this.toy\_shopDataSet.Заказы);

this.покупателиTableAdapter.Fill(this.toy\_shopDataSet.Покупатели);

}

//таблица покупатели

private void dataGridView4\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView4.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView4.SelectedRows[0];

textBox4.Text = selectedRow.Cells["dataGridViewTextBoxColumn3"].Value?.ToString() ?? string.Empty;

textBox1.Text = selectedRow.Cells["dataGridViewTextBoxColumn4"].Value?.ToString() ?? string.Empty;

textBox2.Text = selectedRow.Cells["dataGridViewTextBoxColumn5"].Value?.ToString() ?? string.Empty;

textBox3.Text = selectedRow.Cells["dataGridViewTextBoxColumn6"].Value?.ToString() ?? string.Empty;

}

}

private void button1\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Покупатели (Имя, Отчество, Телефон) VALUES (@Имя, @Отчество, @Телефон)", con);

command.Parameters.AddWithValue("@Имя", textBox1.Text);

command.Parameters.AddWithValue("@Отчество", textBox2.Text);

command.Parameters.AddWithValue("@Телефон", textBox3.Text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Покупатели", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView4.DataSource = table;

}

private void button3\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView4, textBox4.Text, "Покупатели");

}

private void RadioButton\_CheckedChanged1(object sender, EventArgs e)

{

class1.Search(sender, dataGridView4, textBox7.Text, "Покупатели");

}

private void RadioButton\_CheckedChanged2(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView4, "Покупатели");

}

//таблица заказы

private void dataGridView2\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView2.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView2.SelectedRows[0];

textBox10.Text = selectedRow.Cells["idDataGridViewTextBoxColumn1"].Value?.ToString() ?? string.Empty;

textBox5.Text = selectedRow.Cells["покупательDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox6.Text = selectedRow.Cells["магазинDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox8.Text = selectedRow.Cells["датазаказаDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox31.Text = selectedRow.Cells["суммазаказаDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox32.Text = selectedRow.Cells["товарыDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox33.Text = selectedRow.Cells["работникDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

}

}

private void button2\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Заказы (Покупатель, Магазин, Дата\_заказа, Сумма\_заказа, Товары, Работник) VALUES (@2, @3, @4, @5, @6, @7)", con);

command.Parameters.AddWithValue("@2", textBox5.Text);

command.Parameters.AddWithValue("@3", textBox6.Text);

command.Parameters.AddWithValue("@4", textBox8.Text);

command.Parameters.AddWithValue("@5", textBox31.Text);

command.Parameters.AddWithValue("@6", textBox32.Text);

command.Parameters.AddWithValue("@7", textBox33.Text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Заказы", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView2.DataSource = table;

}

private void button4\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView2, textBox10.Text, "Заказы");

}

private void RadioButton\_CheckedChanged3(object sender, EventArgs e)

{

class1.Search(sender, dataGridView2, textBox9.Text, "Заказы");

}

private void RadioButton\_CheckedChanged4(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView2, "Заказы");

}

//таблица магазины

private void dataGridView3\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView3.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView3.SelectedRows[0];

textBox15.Text = selectedRow.Cells["idDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox11.Text = selectedRow.Cells["адрессDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox12.Text = selectedRow.Cells["телефонDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox13.Text = selectedRow.Cells["рейтингDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

}

}

private void button5\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Магазины (Адресс, телефон, Рейтинг) VALUES (@1, @2, @3)", con);

command.Parameters.AddWithValue("@1", textBox11.Text);

command.Parameters.AddWithValue("@2", textBox12.Text);

command.Parameters.AddWithValue("@3", Math.Round(float.Parse(textBox13.Text), 2));

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Магазины", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView3.DataSource = table;

}

private void button6\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView3, textBox15.Text, "Магазины");

}

private void RadioButton\_CheckedChanged5(object sender, EventArgs e)

{

class1.Search(sender, dataGridView3, textBox14.Text, "Магазины");

}

private void RadioButton\_CheckedChanged6(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView3, "Магазины");

}

//таблица работники

private void dataGridView5\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView5.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView5.SelectedRows[0];

textBox20.Text = selectedRow.Cells["idDataGridViewTextBoxColumn3"].Value?.ToString() ?? string.Empty;

textBox16.Text = selectedRow.Cells["имяDataGridViewTextBoxColumn1"].Value?.ToString() ?? string.Empty;

textBox17.Text = selectedRow.Cells["фамилияDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox18.Text = selectedRow.Cells["телефонDataGridViewTextBoxColumn2"].Value?.ToString() ?? string.Empty;

textBox34.Text = selectedRow.Cells["магазинDataGridViewTextBoxColumn1"].Value?.ToString() ?? string.Empty;

textBox35.Text = selectedRow.Cells["должностьDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

}

}

private void button7\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Работники (Имя, Фамилия, телефон, Магазин, Должность) VALUES (@1, @2, @3, @4, @5)", con);

command.Parameters.AddWithValue("@1", textBox16.Text);

command.Parameters.AddWithValue("@2", textBox17.Text);

command.Parameters.AddWithValue("@3", textBox18.Text);

command.Parameters.AddWithValue("@4", textBox34.Text);

command.Parameters.AddWithValue("@5", textBox35.Text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Работники", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView5.DataSource = table;

}

private void button8\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView5, textBox20.Text, "Работники");

}

private void RadioButton\_CheckedChanged7(object sender, EventArgs e)

{

class1.Search(sender, dataGridView5, textBox19.Text, "Работники");

}

private void RadioButton\_CheckedChanged8(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView5, "Работники");

}

//таблица страны

private void dataGridView6\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView6.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView6.SelectedRows[0];

textBox25.Text = selectedRow.Cells["idDataGridViewTextBoxColumn4"].Value?.ToString() ?? string.Empty;

textBox21.Text = selectedRow.Cells["странаDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox22.Text = selectedRow.Cells["валютаDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

}

}

private void button9\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Страны (Страна, Валюта) VALUES (@1, @2)", con);

command.Parameters.AddWithValue("@1", textBox21.Text);

command.Parameters.AddWithValue("@2", textBox22.Text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Страны", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView6.DataSource = table;

}

private void button10\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView6, textBox25.Text, "Страны");

}

private void RadioButton\_CheckedChanged9(object sender, EventArgs e)

{

class1.Search(sender, dataGridView6, textBox24.Text, "Страны");

}

private void RadioButton\_CheckedChanged10(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView6, "Страны");

}

//таблица товары

private void dataGridView7\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView7.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView7.SelectedRows[0];

textBox30.Text = selectedRow.Cells["idDataGridViewTextBoxColumn5"].Value?.ToString() ?? string.Empty;

textBox26.Text = selectedRow.Cells["названиеDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox27.Text = selectedRow.Cells["описаниеDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox28.Text = selectedRow.Cells["ценаDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox23.Text = selectedRow.Cells["количествонаскладеDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

textBox36.Text = selectedRow.Cells["странаDataGridViewTextBoxColumn1"].Value?.ToString() ?? string.Empty;

textBox37.Text = selectedRow.Cells["изображениеDataGridViewTextBoxColumn"].Value?.ToString() ?? string.Empty;

}

}

private void button11\_Click(object sender, EventArgs e)

{

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO Товары (Название, Описание, Цена, Количество\_на\_складе, Страна, Изображение) VALUES (@1, @2, @3, @4, @5, @6)", con);

command.Parameters.AddWithValue("@1", textBox26.Text);

command.Parameters.AddWithValue("@2", textBox27.Text);

command.Parameters.AddWithValue("@3", textBox28.Text);

command.Parameters.AddWithValue("@4", textBox23.Text);

command.Parameters.AddWithValue("@5", textBox36.Text);

command.Parameters.AddWithValue("@6", textBox37.Text);

command.ExecuteNonQuery();

con.Close();

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Товары", con);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView7.DataSource = table;

}

private void button12\_Click(object sender, EventArgs e)

{

class1.Delete(dataGridView7, textBox30.Text, "Товары");

}

private void RadioButton\_CheckedChanged11(object sender, EventArgs e)

{

class1.Search(sender, dataGridView7, textBox29.Text, "Товары");

}

private void RadioButton\_CheckedChanged12(object sender, EventArgs e)

{

class1.Oreder\_By(sender, dataGridView7, "Товары");

}

}

}