ОГЛАВЛЕНИЕ

[Введение 2](#_Toc71712493)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 4](#_Toc71712494)

[1.1. Описание предметной области 4](#_Toc71712495)

[1.2. Анализ СУБД SQL Server Management Studio 5](#_Toc71712496)

[1.3. Язык программирования С# 5](#_Toc71712497)

[1.4. Анализ среды разработки Visual Studio. 7](#_Toc71712498)

[ГЛАВА II. РАЗРАБОТКА БАЗЫ ДАННЫХ «ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ДЛЯ ГИБДД» 11](#_Toc71712499)

[2.1. Проектирование и разработка 11](#_Toc71712500)

[2.2. Создание представлений 13](#_Toc71712501)

[2.3. Создание триггеров 14](#_Toc71712502)

[2.4. Создание хранимых процедур 17](#_Toc71712503)

[2.5. Создание пользовательских функций 19](#_Toc71712504)

[ЗАКЛЮЧЕНИЕ 21](#_Toc71712505)

[СПИСОК ЛИТЕРАТУРЫ 22](#_Toc71712506)

# Введение

Автоматизированная информационная система – совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации. АИС являются, с одной стороны, разновидностью информационных систем (ИС), с другой — автоматизированных систем (АС), вследствие чего их часто называют ИС или АС. АИС может быть определена как комплекс автоматизированных информационных технологий, предназначенных для информационного обслуживания – организованного непрерывного технологического процесса подготовки и выдачи потребителям научной, управленческой и др. информации, используемой для принятия решений, в соответствии с нуждами для поддержания эффективной деятельности.

Актуальность данной выпускной квалификационной работы обусловлена постоянным развитием информационных технологий и необходимостью автоматизировать все процессы. Применение интерфейса для базы данных позволит максимально упростить и оптимизировать труд работника ГИБДД, при документообороте, а также просмотра, поиска и добавлении информации об участниках дорожного движения.

Целью выполнения выпускной квалификационной работы является демонстрация приобретённых мною практических навыков проектирования баз данных и разработки программного обеспечения по управлению базами данных с использованием конкретной СУБД. В ходе выполнения данного дипломного проекта необходимо разработать базу данных ГИБДД и программное обеспечение для базы данных.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ предметной области;
2. Выбрать необходимые программные средства;
3. Спроектировать базу данных для каталога продукции;
4. Разработать интерфейс для созданной базы данных;
5. Создать представления в СУБД для оптимизации кода и корректности отображения данных на интерфейсе.

Объектом исследования является информационная база ГИБДД.

Предметом исследования является разработка базы данных и приложения для ГИБДД.

В ходе выполнения работы будет проведена оптимизация работы АИС для увеличения производительности и скорости ГИБДД.

В ходе исследования применялись следующие методы: анализ, моделирование, изучение научной литературы и ее обобщение.

Информационной базой выпускной квалификационной работы является научная и техническая литература, электронные источники.

# **ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

## Описание предметной области

Инспектору ГИБДД необходимо внести полную информацию о физическом лице, получающем водительское удостоверение, то есть Фамилию Имя Отчество, дату рождения, адрес проживания и иные паспортные данные, дата выдачи, кем выдано и на какие категории транспортных средств. Также в базу данных необходимо внести полную информацию о регистрируемом транспортном средстве (марка, модель, цвет, год выпуска). Категории транспортных средств регистрируются следующие: мопед, мотоцикл, легкий мотоцикл, легковые автомобили, квадроцикл, тяжелые автомобили, автобус, трамвай, троллейбус.

Если говорить конкретно, то в базе данных будет содержаться:

1. Данные о сотрудниках ГИБДД;
2. Данные о водителях;
3. Данные о водительских правах;
4. Данные о ТС;
5. Данные о регистрации ТС;
6. Данные о категориях ТС.

Данная база данных создается с целью корректного и безопасного хранения данных, ускорения выполнения работы, а также возможностью добавлять, редактировать и удалять информацию.

Главная цель системы –возможность быстрого поиска транспорта или его владельца по разным критериям для проведения различных операций над ними, а также удобный анализ информации.

## Анализ СУБД SQL Server Management Studio

Среда «SQL Server Management Studio» это единая универсальная среда для доступа, настройки и администрирования всех компонентов MS SQL Server, а также для разработки компонентов системы, редактирования текстов запросов, создания скриптов и пр. Благодаря наличию большого количества визуальных средств управления, среда «SQL Server Management Studio» позволяет выполнять множество типовых операций по администрированию MS SQL Server администраторам с любым уровнем знаний SQL Server. Удобная среда разработки, встроенный веб–браузер для быстрого обращения к библиотеке или получения справки в сети, подробный учебник, облегчающий освоение многих новых возможностей встроенная справка от сообществ в Интернете и многое другое позволяют максимально облегчить процесс разработки в среде SQL Server, а также дает богатые возможности для создания различных сценариев SQL Server.

## Язык программирования С#

Язык программирования C# – популярный язык программирования, который представляет собой современный объектно–ориентированный язык для разработки кроссплатформенных приложений. Он обеспечивает создание безопасного и верифицируемого кода, применение которого возможно в любом современном программном комплексе. Созданные на языке C# приложения могут успешно работать на любом компьютере под управлением операционной системы Windows, Linux и MacOS независимо от архитектуры железа. C# относится к семейству языков с C–подобным синтаксисом, из них он наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, в отличие от C++ и некоторых других языков, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

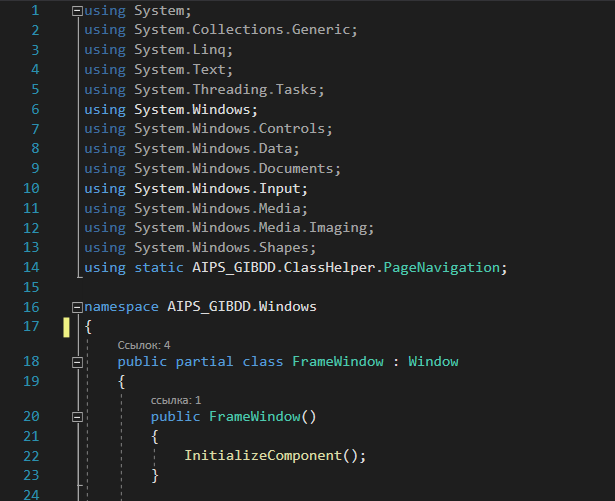


Рис. 1.1 Пример кода C# в среде разработки Visual Studio

Достоинства:

1. Высокая совместимость с языком С, позволяющая использовать весь существующий С–код.
2. Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщенное программирование.
3. Имеется возможность работы на низком уровне с памятью, адресами, портами.
4. Кроссплатформенность. Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем.
5. Эффективность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы.

Недостатки:

1. Синтаксис, провоцирующий ошибки:
2. Препроцессор, унаследованный от С, очень примитивен.
3. Плохая поддержка модульности

## Анализ среды разработки Visual Studio.

Visual Studio — полнофункциональная интегрированная среда разработки (IDE) программного обеспечения с поддержкой популярных языков программирования, среди которых С, C++, VB.NET, C#, F#, JavaScript, Python. Она позволяет создавать не только консольные приложения, но и приложения с графическим интерфейсом с помощью технологии WPF (Windows Presentation Foundation). Её функциональность может охватывать все этапы разработки программного обеспечения, предоставляя самые современные средства для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Редактор кода Visual Studio поддерживает подсветку синтаксиса, вставку фрагментов кода, отображение структуры и связанных функций. Встроенный отладчик Visual Studio используется для поиска и исправления ошибок в исходном коде, в том числе на низком аппаратном уровне. Инструменты диагностики Visual Studio позволяют узнать качество кода с точки зрения производительности и использования памяти. Дизайнер форм Visual Studio является незаменимым при разработке приложений с графическим интерфейсом, помогая спроектировать дизайн будущего приложения и работу каждого элемента интерфейса. Для командных проектов Visual Studio предлагает поддержку работы в группе, позволяя выполнять совместное редактирование и отладку любой части кода в режиме реального времени, а в качестве системы управления версиями использовать Git или Team Foundation. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense которая автоматически дополняет код по мере ввода. и даёт возможность простейшего редактирования кода. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно–ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Достоинства Visual Studio:

1. Поддержка множества языков при разработке. Visual Studio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE).
2. Меньше кода для написания. Для создания большинства приложений требуется приличное количество стандартного стереотипного кода. В Visual Studio такие детали устанавливаются автоматически.
3. Интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Применяемые Visual Studio автоматически параметры форматирования можно даже настраивать, что очень удобно в случаях, когда разработчик предпочитает другой стиль размещения скобок
4. Более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense, которая умеет перехватывать ошибки и предлагать правильные варианты; функции поиска и замены, которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте; функции автоматического добавления и удаления комментариев, позволяют разработчику работать быстро и эффективно.
5. Возможности отладки. Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

# ГЛАВА II. РАЗРАБОТКА БАЗЫ ДАННЫХ «ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ДЛЯ ГИБДД»

## Проектирование и разработка

Инфологическое проектирование – первый этап проектирования базы данных, подразумевает создание семантической модели предметной области. Описание должно быть максимально емким, чтобы можно было оценить глубину и корректность проработки проекта БД, и оно не должно быть привязано к конкретной СУБД.

В настоящий момент модель «сущность-связь», стала фактическим стандартом при инфологическом моделировании баз данных. Общепринятым стало сокращенное название ER-модель, большинство современных CASE-средств содержат инструментальные средства для описания данных в формализме этой модели. Эта модель в наибольшей степени согласуется с концепцией объектно-ориентированного проектирования, которая в настоящий момент, несомненно, является базовой для разработки сложных программных систем.

В основе ER-модели лежат следующие базовые понятия:

1. Сущность – это любой различимый объект, информацию о котором необходимо хранить в базе данных;

2. Атрибут – это свойство сущности или связи, получаемое путем наблюдения или измерения;

3. Связь – это отношение между 2 сущностями, каждая из которых относится к некоторому набору сущностей.

Логическое проектирование – второй этап проектирования базы данных, включает процесс конструирования общей информационной модели предприятия на основе отдельных моделей данных пользователей, которая является независимой от особенностей реально используемой СУБД и других физических условий. Этап логического проектирования делят на концептуальную и логическую часть. Концептуальное проектирование базы данных обычно предшествует этапу логического проектирования и подразумевает моделирование информации без использования какой-либо базовой модели данных. Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Одним из важных разделов создания базы данных с помощью любой СУБД является нормализация данных.

Формы нормализации данных:

1. Первая нормальная форма: переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов;

2. Вторая нормальная форма: переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо зависит от её потенциального ключа;

3. Третья нормальная форма: Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

Физическое проектирование – третий и последний этап создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например, реляционной, сетевой или иерархической. Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

В ходе проделанной работы, была разработана ER-модель (Приложение 1). в Microsoft Visio, а затем и физическая модель базы данных в СУБД MS SQL Server (Приложение 2).

## Создание представлений

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

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

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

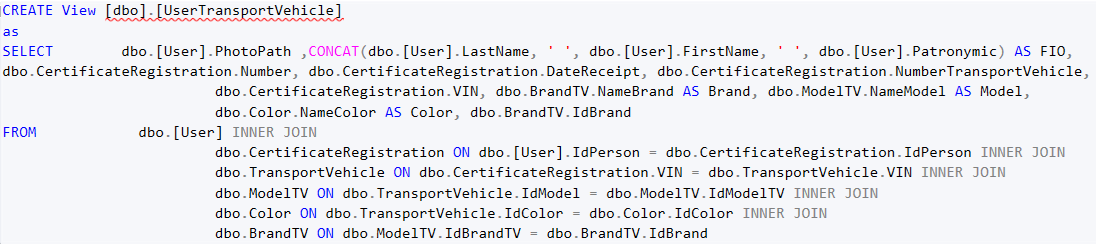


Рис. 2.7 «Запрос на создание представления который выведет информацию о людях и их транспортных средствах»



Рис. 2.8 «Результат работы представления»

## Создание триггеров

Триггер – это механизм, который вызывается, когда в указанной таблице происходит определенное действие. Каждый триггер имеет следующие основные составляющие: имя, действие и исполнение. Имя триггера может содержать максимум 128 символов. Действием триггера может быть или инструкция DML (INSERT, UPDATE или DELETE), или инструкция DDL. Таким образом, существует два типа триггеров: триггеры DML и триггеры DDL. Исполнительная составляющая триггера обычно состоит из хранимой процедуры или пакета.

Для создания триггера применяется выражение CREATE TRIGGER [Имя триггера]. Как правило, имя триггера отражает тип операций и имя таблицы, над которой производится операция. Каждый триггер ассоциируется с определенной таблицей или представлением, имя которых указывается после слова ON.

Также можно задать тип триггера с помощью двух дополнительных параметров: AFTER и INSTEAD OF. (Параметр FOR является синонимом параметра AFTER.) Триггеры типа AFTER вызываются после выполнения действия, запускающего триггер, а триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггер. Триггеры AFTER можно создавать только для таблиц, а триггеры INSTEAD OF - как для таблиц, так и для представлений.

Параметры INSERT, UPDATE и DELETE задают действие триггера. Под действием триггера имеется в виду инструкция Transact-SQL, которая запускает триггер. Допускается любая комбинация этих трех инструкций. Инструкция DELETE не разрешается, если используется параметр IF UPDATE.

При создании действия триггера обычно требуется указать, ссылается ли он на значение столбца до или после его изменения действием, запускающим триггер. По этой причине, для тестирования следствия инструкции, запускающей триггер, используются две специально именованные виртуальные таблицы:

1. deleted - содержит копии строк, удаленных из таблицы;
2. inserted - содержит копии строк, вставленных в таблицу.

Структура этих таблиц эквивалентна структуре таблицы, для которой определен триггер.

Таблица deleted используется в том случае, если в инструкции CREATE TRIGGER указывается предложение DELETE или UPDATE, а если в этой инструкции указывается предложение INSERT или UPDATE, то используется таблица inserted.

Инструкция UPDATE рассматривается, как инструкция DELETE, за которой следует инструкция INSERT. Поэтому для каждой инструкции UPDATE, выполненной в действии триггера, создается как таблица deleted, так и таблица inserted.

Таблицы inserted и deleted реализуются, используя управление версиями строк. Когда для таблицы с соответствующими триггерами выполняется инструкция DML (INSERT, UPDATE или DELETE), для всех изменений в этой таблице всегда создаются версии строк. Когда триггеру требуется информация из таблицы deleted, он обращается к данным в хранилище версий строк. В случае таблицы inserted, триггер обращается к самым последним версиям строк.

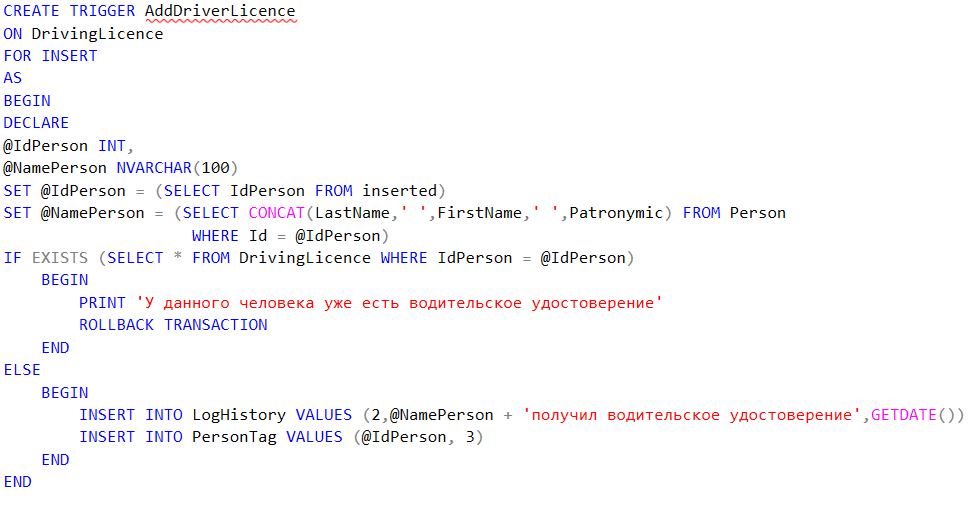


Рис. 2.11 «Создание триггера на добавление водительских прав»

Данный триггер срабатывает на попытку добавления записи в таблицу «DrivingLicence». Сначала при помощи IF EXISTS идёт проверка на наличие у этого человека водительских прав. Если у этого человека уже есть водительское удостоверение, он отменяет транзакцию. В обратном случае, запись проходит, а так же триггер добавляет в таблицу LogHistory(История) запись о том когда и кто, получил водительское удостоверение.

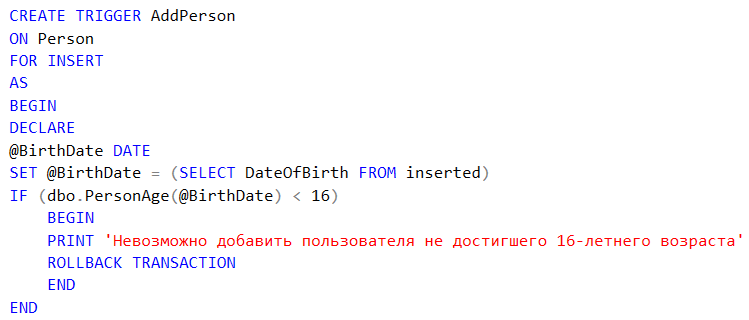


Рис. 2.14 «Создание триггера на добавление водителя»

Этот триггер срабатывает на добавление записи в таблицу «User». Его функция заключается в проверке возраста вносимого в базу человека. В данном триггере используется функция «PersonAge» для подсчёта возраста. В случае если возраст меньше 16 лет (минимальный возраст получения водительского удостоверения), триггер не даёт внести в базу запись.

На практике же триггеры используются очень часто, так как они помогают предотвратить случайные или ненужные ошибки.

## Создание хранимых процедур

Хранимые процедуры используются для сохранения на сервере повторно используемого кода, например, Вы написали некий алгоритм, последовательный расчет или многошаговую SQL инструкцию, и чтобы каждый раз не выполнять все инструкции, входящие в данный алгоритм, Вы можете оформить его в виде хранимой процедуры. При этом, когда Вы создаете процедуру SQL, сервер компилирует код, а потом, при каждом запуске этой процедуры SQL сервер уже не будет повторно его компилировать.

Для того чтобы запустить хранимую процедуру в SQL Server, необходимо перед ее названием написать команду EXECUTE, также возможно сокращенное написание данной команды EXEC. Вызвать хранимую процедуру в инструкции SELECT, например, как функцию уже не получится, т.е. процедуры запускаются отдельно.

В хранимых процедурах, в отличие от функций, уже можно выполнять операции модификации данных такие как: INSERT, UPDATE, DELETE. Также в процедурах можно использовать SQL инструкции практически любого типа, например, CREATE TABLE для создания таблиц или EXECUTE, т.е. вызов других процедур. Исключение составляет несколько типов инструкций таких как: создание или изменение функций, представлений, триггеров, создание схем и еще несколько других подобных инструкций, например, также нельзя в хранимой процедуре переключать контекст подключения к базе данных (USE).

Хранимая процедура может иметь входные параметры и выходные параметры, она может возвращать табличные данные, может не возвращать ничего, только выполнять заложенные в ней инструкции.

Хранимые процедуры очень полезны, они помогают нам автоматизировать или упростить многие операции, например, Вам постоянно требуется формировать различные сложные аналитические отчеты с использованием сводных таблиц, т.е. оператора PIVOT. Чтобы упростить формирование запросов с этим оператором (как Вы знаете, у PIVOT синтаксис достаточно сложен), Вы можете написать процедуру, которая будет Вам динамически формировать сводные отчеты, например, в материале «Динамический PIVOT в T-SQL» представлен пример реализации данной возможности в виде хранимой процедуры.

Преимущества хранимых процедур:

1. Снижение сетевого трафика между клиентами и сервером;

2. Большая безопасность;

3. Более легкое обслуживание;

4. Повышение производительности.

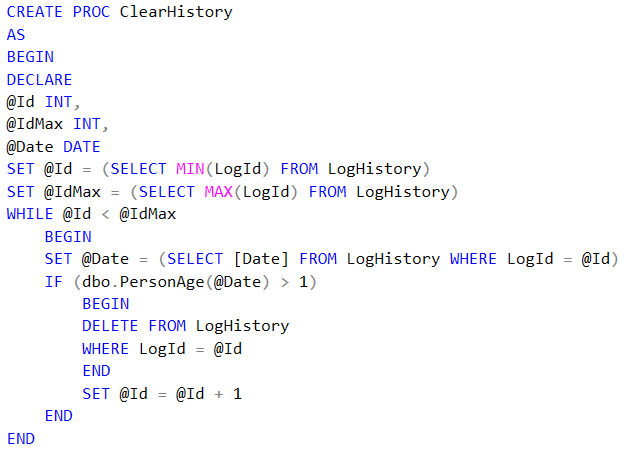


Рис. 2.16 «Создание процедуры на очищение истории»

Данная процедура используется для очистки таблицы «LogHistory». Она смотрит на дату добавления, и удаляет все записи старее 1 года. Это позволяет освободить пространство на диске для уменьшения размера базы.

## Создание пользовательских функций

Аналогично функциям в языках программирования, определяемые пользователем функции SQL Server, представляют собой подпрограммы, которые принимают параметры, выполняют действие, например, сложные вычисления, и возвращают результат этого действия в виде значения. Возвращаемое значение может быть либо единичным скалярным значением, либо результирующим набором.

Типы функций:

1. Скалярная функция- возвращает одно значение типа данных, заданного в предложении RETURNS. Для встроенной скалярной функции возвращаемое скалярное значение является результатом одной инструкции. Скалярная функция из нескольких инструкций имеет текст может содержать последовательность инструкций Transact-SQL, возвращающих одно значение. Такие функции могут возвращать любые типы данных, кроме TEXT, NTEXT, IMAGE, CURSOR и TIMESTAMP.
2. Функции с табличными значениями - возвращают данные типа TABLE. Встроенная функция с табличным значением не имеет текста, таблица является результирующим набором одной инструкции.
3. Системные функции – нужны для выполнения различных операций. Их нельзя изменить.

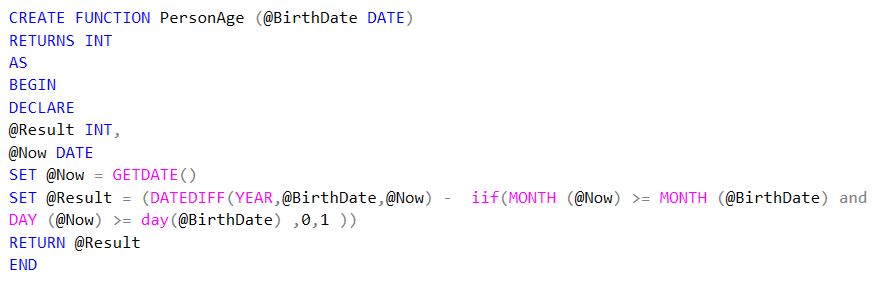


Рис. 2.16 «Создание функции разницы лет»

Данная функция может использоваться, для подсчёта разницы даты в годах, между текущим временем и принимаемым значением даты

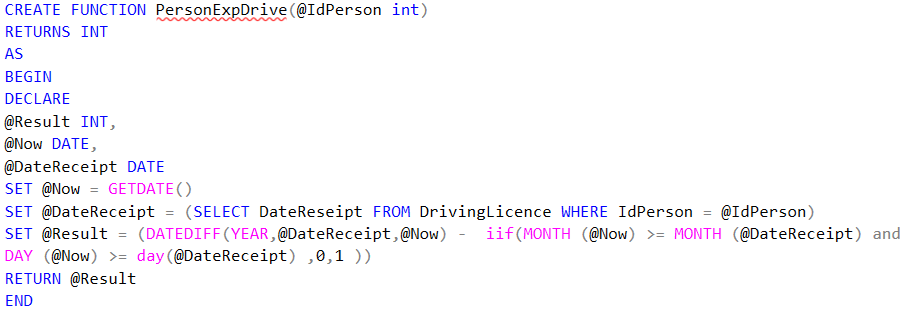


Рис. 2.17 «Создание функции для подсчёта водительского стажа»

Данная функция используется для подсчёта водительского стажа в годах.

## Создание пользовательского приложения

Передо мной стояла задача, создать визуальный интерфейс для удобного представления и легкого поиска информации об участниках дорожного движения, которым будут пользоваться только сотрудники ГИБДД.

Сначала сотрудника встретит форма авторизации.

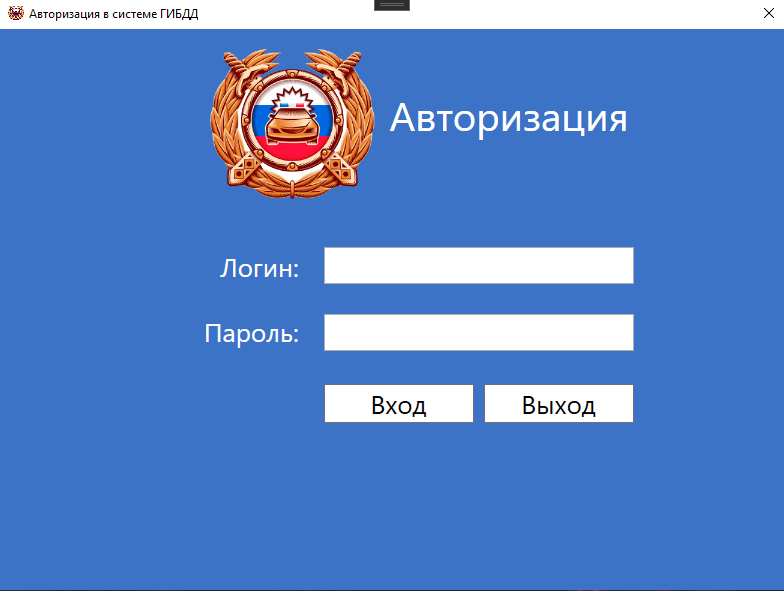


Рис 2.1 «Форма авторизации»

Если все поля заполнены, то приложение обратится с запросом к таблице к «Employee» где хранятся данные для входа всех сотрудников. Если введённые данные не совпадают с хранимыми в базе данных, приложение выдаст ошибку «Неправильный логин или пароль». Если данные совпадают с хранимыми в таблице, приложение откроет основную форму и занесёт в базу данных в таблицу «Log» запись о входе сотрудника в систему (ФИО, время).

Следующая форма представляет собой окно, в котором навигация реализуется с помощью перехода по страницам. Слева находится навигационное меню, а на стартовой странице находится актуальная дата и время, а также кнопки для перехода на самые используемые страницы: «Поиск», «Водительские удостоверения», «Штрафы».

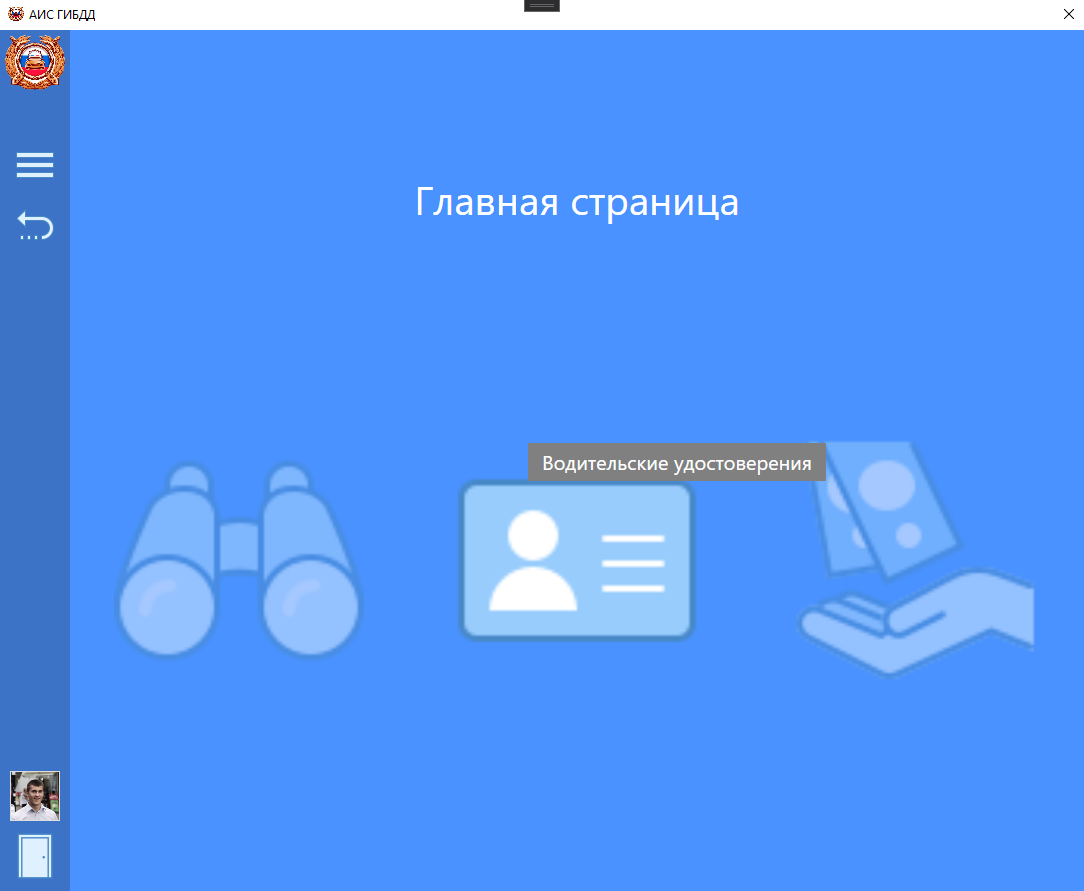


Рис.1.1 «Стартовая страница»

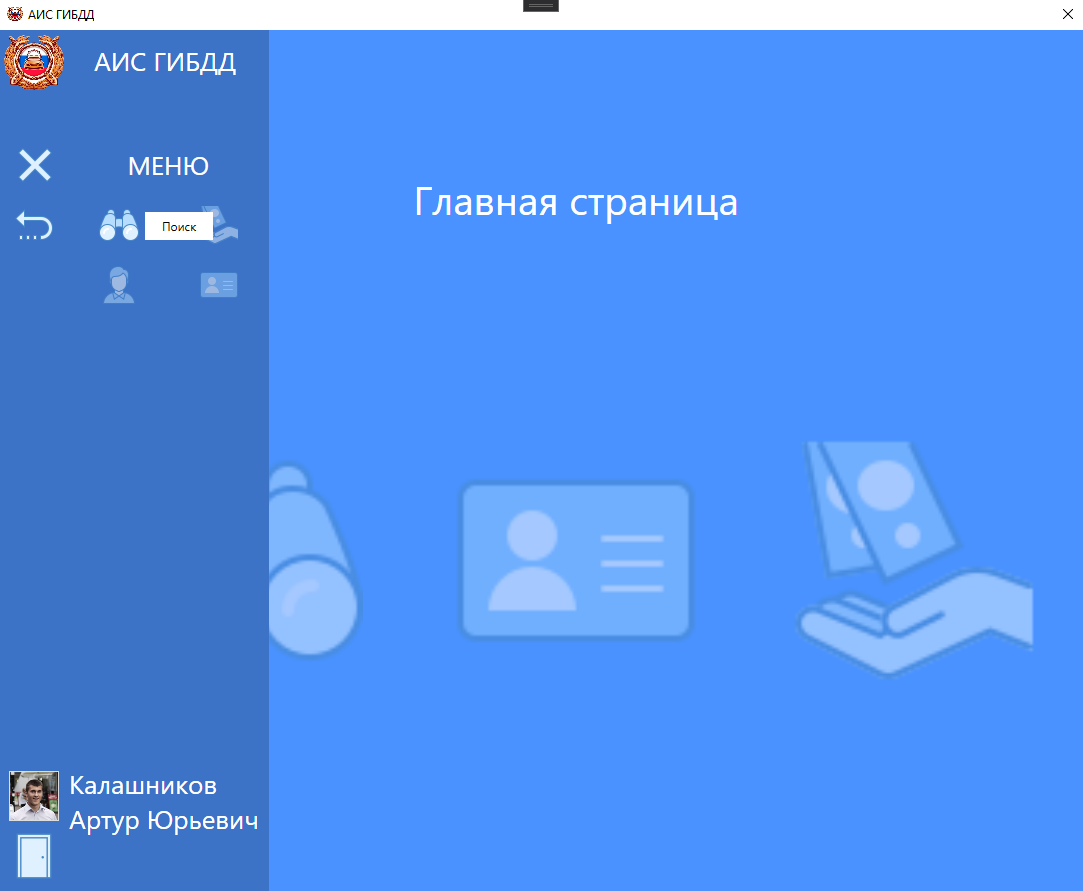


Рис 1.1 «Открытое меню»

При нажатии на кнопку «Поиск» в меню или на стартовой странице, нас перекинет на страницу с информацией о транспортных средствах, и их владельцах выведенные в «ListView». Поиск осуществляется в режиме реального времени, по ФИО водителя, гос. номеру или VIN транспортного средства (далее ТС). Присутствует возможность фильтра по марке, модели и цвету ТС, реализованная с помощью объекта «ComboBox», данные о существующих марках и моделях загружается из БД.

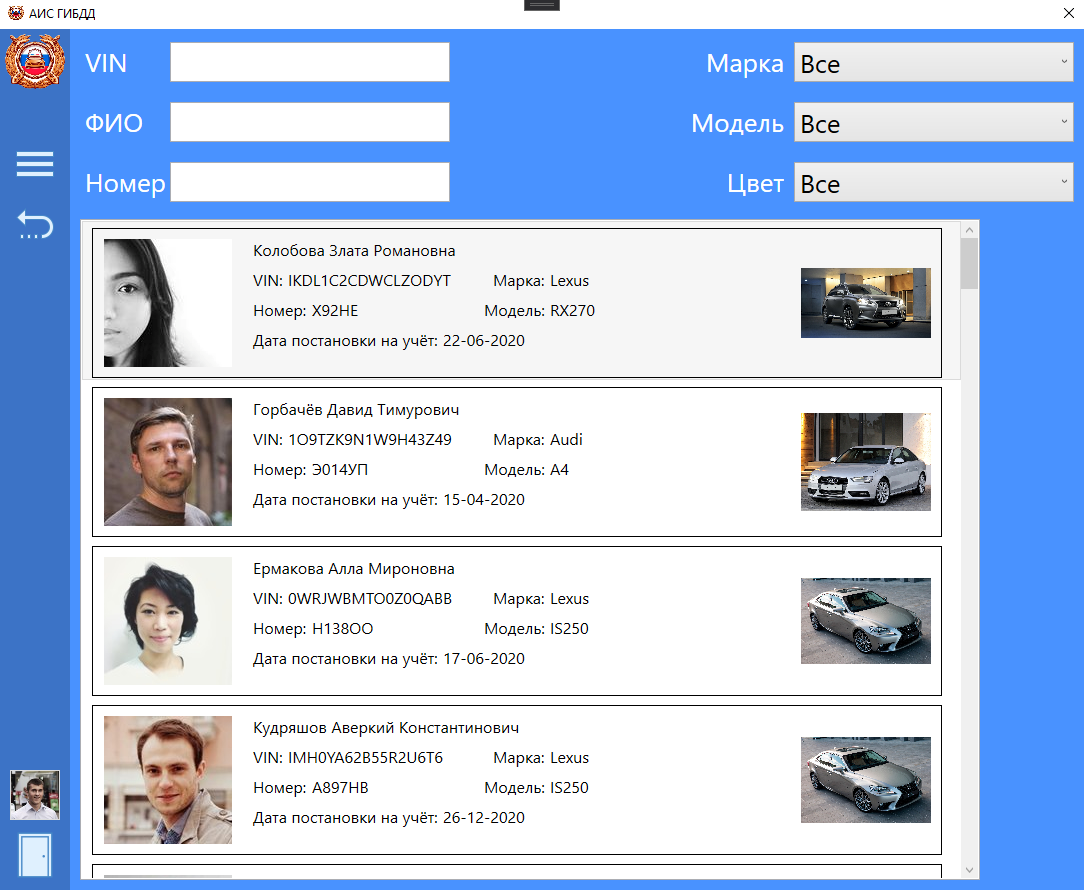


Рис. 1.1 «Страница поиска»

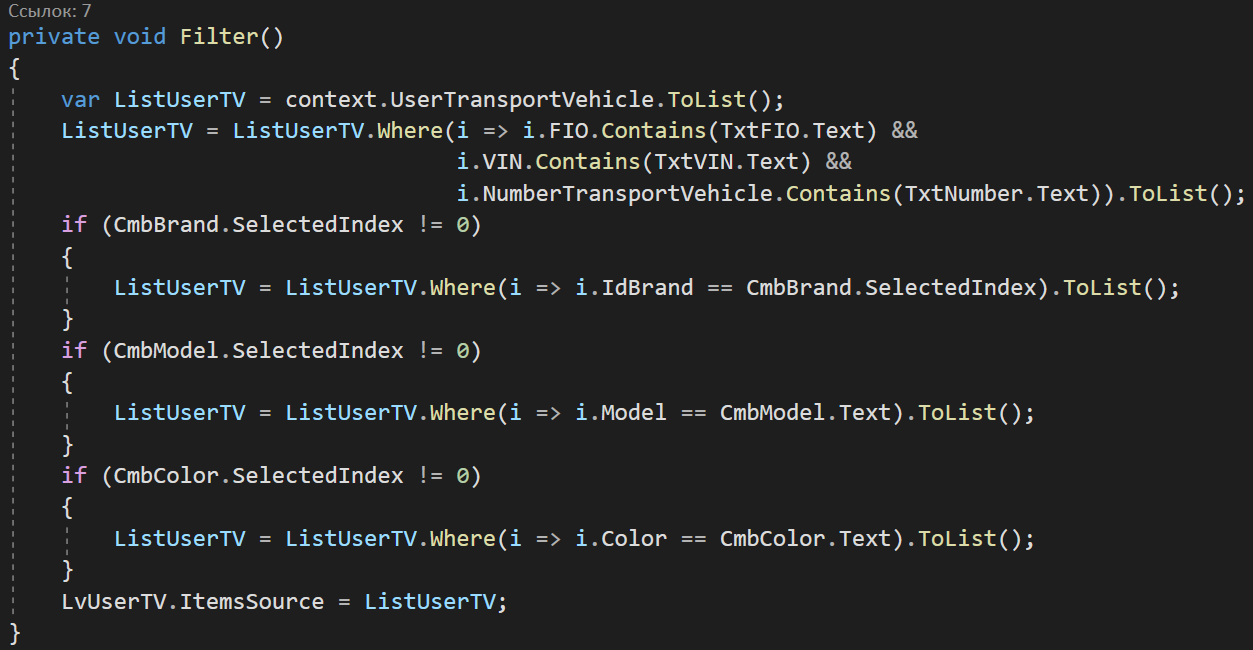


Рис. 1.1 «Реализация поиска на С»

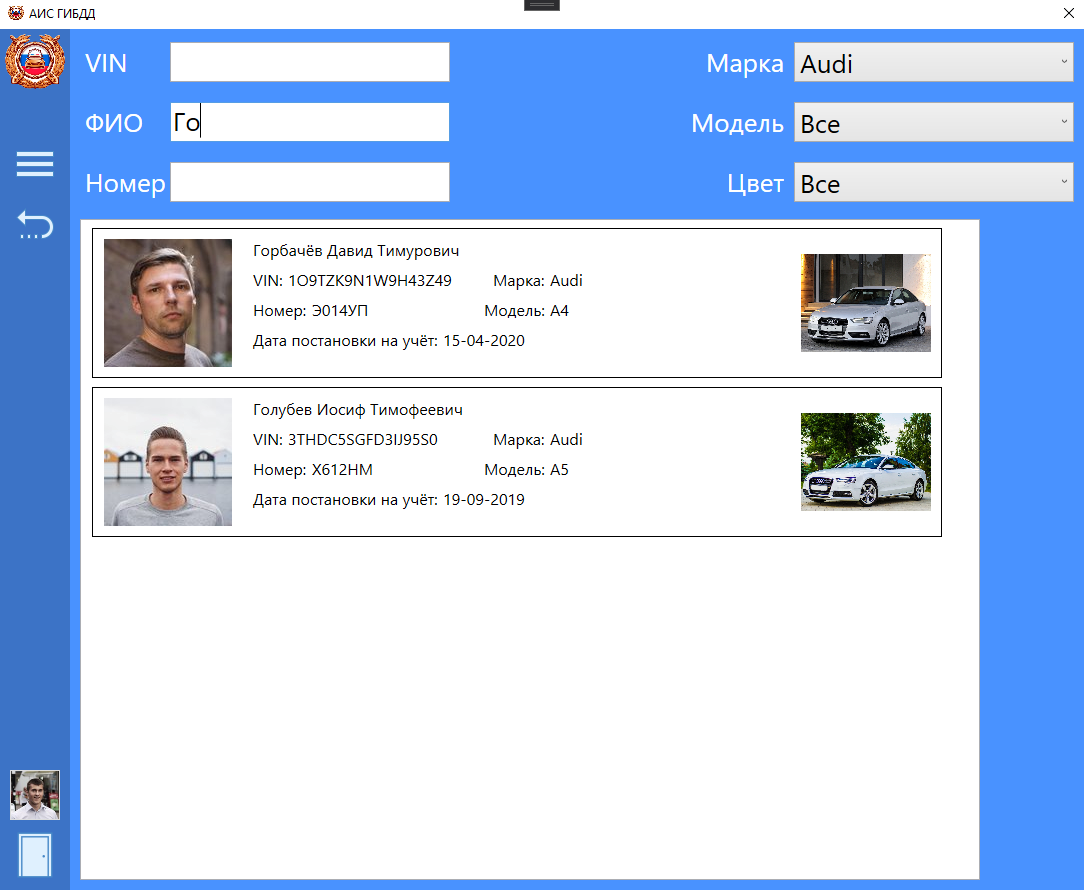


Рис. 1.1 «Пример поиска»

Через меню

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были закреплены полученные знания и выполнены все поставленные задачи. В данной работе была создана база данных для информационно-поисковой системы для ГИБДД, которая значительно упрощает работу с данными, помогает в ведении учёта информации об участниках дорожного движения, транспортных средствах, водительских удостоверений с целью оптимизации рабочего времени, быстрого поиска нужного человека или транспортного средства по одному из атрибутов: ФИО владельца, государственный номер транспортного средства, номер водительского удостоверения.

В базе данных сведения из каждого источника сохраняются в отдельной таблице. При работе с данными из нескольких таблиц устанавливаются связи между таблицами. Для поиска и отбора данных, удовлетворяющих определённым условиям, создаётся запрос. Запросы также позволяют обновить или удалить одновременно несколько записей, выполнить встроенные и специальные сообщения.

Применение разработанной базы данных позволит упростить труд персонала, а также найти любую интересующую информацию.

Для разработки базы данных использовалась система управления базами данных MS SQL 2019.

В работе применялись знания, полученные за время обучения и информация из дополнительных источников.

Готовый продукт обеспечивает эффективность хранения и обработку данных, что является очень важной составляющей работы автоматизированной информационно-поисковой системы. База данных может быть расширена и дополнена новыми данными, таблицами и запросами.

# СПИСОК ЛИТЕРАТУРЫ

1. Капрова Т. С. Базы данных: модели, разработка, реализация / Капрова Т. С. – СПб: издательство «Экзамен», 2016 – 304 с.
2. Лазицкас Е. А. Базы данных и системы управления базами данных / Зигумёникова И. Н., Гилевский П. Г – Москва: издательство «Учебное пособие», 2016 – 266 с.;
3. Кузнецов. С. Введение в реляционные базы данных / Москва: издательство «Экзамен»,2016 – 248 с.;
4. Сирант О. В. Работа с базами данных / Санкт - Питербург: издательство «Питер»,2016 – 150 с.;
5. Кумскова И. Базы данных / Санкт - Питербург: издательство «Питер»,2018 – 400 с.;
6. Нестеров. С. А. Базы данных. Учебник и практикум / Москва: издательство «МИФ»,2017 – 230 с.;
7. Медведкова И. Е. Базы данных / Бугаев Ю. В., Чикунов С. В Москва: издательство «Clever»,2018 – 105 с.;
8. Маркин А. В Программирование на SQL Учебное пособие для СПО / Москва: издательство «Clever»,2019 – 435 с.;

Интернет источники

1. Metanit.com Сайт о программировании [Электронный ресурс]. Режим доступа: https://metanit.com/;
2. javarush.ru Сайт по программированию [Электронный ресурс]. Режим доступа: https://javarush.ru;
3. Статья о sql server, запросы [Электронный ресурс]. Режим доступа: https://www.npk-kaluga.ru/MakeQuery\_MSSQL.htm;
4. Статья о запросах sql server [Электронный ресурс]. Режим доступа: https://itvdn.com/ru/blog/article/m-sql;
5. Официальный сайт Microsoft, урок запросов и создание баз данных [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/sql/t-sql/lesson-1-c;
6. Работа с БД [Электронный ресурс]. Режим доступа: https://professorweb.ru/.

Приложение №1. ER-модель базы данных в Microsoft Visio



Приложение №2. Диаграмма базы данных в СУБД MS SQL Server

