

**КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

Тема: Проектирование базы данных информационно-поисковой системы для ГИБДД.

Обучающийся группы: 3ИСП11-8 Потегаев Е.В.

*номер группы И.О. Фамилия*

Специальность: 09.02.07 Информационные системы и программирование

Руководитель / Куропаткина О.П.

*подпись И.О. Фамилия*

Москва

2021

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc65659645)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 4](#_Toc65659646)

[1.1. Описание предметной области 4](#_Toc65659647)

[1.2. Инфологическое проектирование 5](#_Toc65659648)

[1.3. Логическое проектирование базы данных 6](#_Toc65659649)

[1.4. Выбор СУБД 7](#_Toc65659650)

[ГЛАВА II. РАЗРАБОТКА БАЗЫ ДАННЫХ «ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ДЛЯ ГИБДД» 9](#_Toc65659651)

[2.1. Физическое проектирование 9](#_Toc65659652)

[2.2. Создание таблиц 9](#_Toc65659653)

[2.3. Определение условий целостности данных 11](#_Toc65659654)

[2.4. Создание запросов на выборку данных 12](#_Toc65659655)

[2.5. Создание представлений 13](#_Toc65659656)

[2.6. Создание триггеров. 15](#_Toc65659657)

[2.7. Создание хранимых процедур 19](#_Toc65659658)

[2.8. Создание пользовательских функций 21](#_Toc65659659)

[ЗАКЛЮЧЕНИЕ 23](#_Toc65659660)

[СПИСОК ЛИТЕРАТУРЫ 24](#_Toc65659661)

# ВВЕДЕНИЕ

Целью выполнения курсовой работы является закрепление полученных теоретических знаний и практических умений по общепрофессиональным и специальным дисциплинам специальности “Информационные системы и программирование” и разработки базы данных с использованием СУБД. В ходе выполнения данного курсового проекта необходимо разработать базу данных «ГИБДД». Информационно-поисковая система занимается хранением информации о транспортных средствах, участниках дорожного движения, их водительских удостоверениях и нарушениях.

Актуальность обусловлена развитием информационных технологий и необходимостью автоматизировать все процессы, для облегчения рабочего труда.

Для выполнения поставленной задачи необходимо:

1. Проанализировать предметную область;
2. Спроектировать базу данных;
3. Создать таблицы;
4. Создать связи между таблицами;
5. Поставить ограничения;
6. Создать запросы;
7. Сделать представления;
8. Создать триггеры;
9. Создать функции;
10. Сделать хранимые процедуры.

Объектом исследования является MS SQL-Server, так как именно в нём будет создана база данных и выполнятся поставленные задачи.

Предметом исследования является процесс автоматизации работы в ГИБДД, а именно, хранение данных, их анализ и работа с ними. Информационно-поисковая система позволяет выполнять все эти действия с меньшими усилиями.

# 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, мощность). 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 водителях;
2. Данные о водительских правах;
3. Данные о ТС;
4. Данные о регистрации ТС;
5. Данные о категориях ТС.

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

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

Использование данной информационной системы существенно помогает ускорить выполнение работы.

## Инфологическое проектирование

Инфологическая модель применяется на втором этапе проектирования БД, то есть после словесного описания предметной области.

Инфологическая модель должна включать такое формализованное описание предметной области, которое будет понятным не только для специалистов по базам данных, но и сторонних людей. Описание должно быть максимально емким, чтобы можно было оценить глубину и корректность проработки проекта БД, и оно не должно быть привязано к конкретной СУБД.

Выбор СУБД - это отдельная задача, для корректного ее решения необходимо иметь проект, который не привязан ни к какой конкретной СУБД. Инфологическое проектирование прежде всего связано с попыткой представления смыслового содержания базы данных, её предметной области в модели.

Реляционная модель данных в силу своей простоты и лаконичности не позволяет отобразить смысл предметной области, и он остается за рамками реляционной модели. Ранние теоретико-графовые модели в большей степени, чем реляционная модель, отображали идею предметной области. Они в явном виде определяли иерархические связи между объектами предметной области.

Проблема представления смысла давно интересовала разработчиков, и в семидесятых годах было предложено несколько моделей данных, названных семантическими моделями. К ним можно отнести модель «сущность-связь».

У всех моделей были свои положительные и отрицательные стороны, но испытание временем выдержала только одна. И в настоящий момент именно 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. Кроме того, разработаны методы автоматического преобразования проекта БД из ER-модели в реляционную модель, соответствующую конкретной СУБД. [1]

В настоящий момент не существует единой общепринятой системы обозначений для ER-модели и разные CASE-системы используют разные графические нотации, но разобравшись в одной, можно легко понять и другие нотации.

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 базы данных

Логическое проектирование - это процесс конструирования общей информационной модели предприятия на основе отдельных моделей данных пользователей, которая является независимой от особенностей реально используемой СУБД и других физических условий.[3]

Одним из главных преимуществ этого этапа является то, что всегда можно взять черновой проект, отложить его в сторону и начать все заново или просто внести желаемые поправки. Гораздо легче менять те или иные детали на этапе проектирования, чем иметь дело с проблемами уже реализованной производственной базы данных, которая плохо спроектирована.

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. Примеры понятий - «сотрудник», «отдел», «проект», «зарплата». Примеры взаимосвязей между понятиями - «сотрудник числится ровно в одном отделе», «сотрудник может выполнять несколько проектов». Одним из важных разделов создания базы данных с помощью любой СУБД является нормализация данных.[6]

Нормализация 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.

## Выбор СУБД

Microsoft SQL Server представляет собой СУБД, обеспечивающую создание информационных систем с архитектурой “клиент-сервер”, в которой он играет роль сервера баз данных. SQL Server поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных, отличается простотой управления и использования. Эта СУБД обеспечивает высокую степень защиты данных, как от случайных потерь, так и от несанкционированного доступа, обладает развитыми средствами обработки данных и хорошим быстродействием. MS SQL Server позволяет хранить огромные объёмы данных. SQL Server добавляет к сетевым компонентам специальные сервисы, такие как OLEDB (Object Linking and Embedding-Database - связывание и внедрение объектов базы данных) и ODBC (Open Database Connectivity - совместимость от­крытых баз данных). С их помощью обеспечивается совместимость различных кли­ентских приложений при работе с сервером.[2]

Пользователь компьютера-клиента с помощью сетевых средств своей операционной системы может устанавливать связь с компьютером-сервером, где установлен SQL Server. На компьютерах-клиентах размещаются локаль­ные базы данных, работа с которыми ведется с помощью персональных СУБД (Access, Visual FoxPro) или языков программирования (Visual Basic, Delphi, C++ Builder, Visual C++). С их помощью через ODBC осу­ществляется доступ к базам дан­ных, размещенным на сервере.

В SQL Server используется понятие роли(role). Каждому пользователю может быть назначено произ­вольное число ролей. Например, пользователю может быть назначена роль Администратора.

Достоинства:

1. Бесплатная;

2. Является реляционной (Реляционная база данных представляет собой набор взаимосвязанных двухмерных таблиц (отношений)), полнофункциональной;

3. Используется для создания распределенных баз данных коллективного использования;

4. Реализует многопользовательский удаленный режим типа Клиент‑Сервер доступа к данным.

Недостатки:

1. Привязка к операционным системам Windows;

2. Сложный для освоения интерфейс.

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

## Физическое проектирование

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; 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.[7]

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных. Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например, реляционной, сетевой или иерархической. Однако, 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.[8]

Была разработана физическая модель базы данных в СУБД MS SQL Server (Приложение 1)

## Создание таблиц

MS SQL Server предоставляет 2 способа создания таблиц:

1. С помощью графического интерфейса;
2. С помощью запроса.

Создание таблиц с помощью графического интерфейса является более быстрым и удобным способом для начинающего пользователя. Для этого нажимаем правой кнопкой мыши по объекту «Базы данных», затем открываем нужную базу данных и щелкаем правой кнопкой мыши по пункту «Таблицы», и выбираем «Создать таблицу». Или можно создавать таблицы прямо в диаграмме базы данных. Для этого щёлкаем правой кнопкой мыши по пустому месту и выбираем «Создать таблицу». После этого введите название создаваемой таблицы. Далее откроется конструктор таблиц, в котором будет всего три колонки:

1. Имя столбца;
2. Тип данных;
3. Разрешить значение NULL.

Второй способ – создание таблицы с помощью запроса CREATE TABLE. Этот способ немного сложнее, но он более гибкий и удобный для продвинутых пользователей.

Создадим несколько основных сущностей для базы данных:

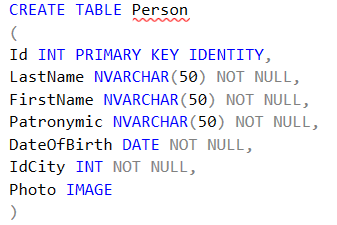


Рис. 2.1 «Создание таблицы Person запросом»

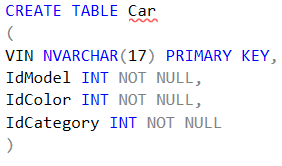


Рис. 2.2 «Создание таблицы Car»

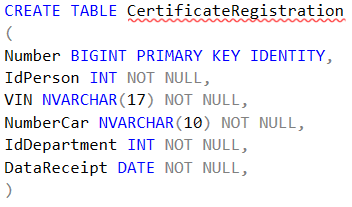


Рис. 2.3 «Создание таблицы CertificateRegistration»

## Определение условий целостности данных

Целостность базы данных - соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Целостность данных поддерживается с помощью ограничений, которые не являются обязательными для таблицы.[4]

В MS SQL Server есть 4 основных ограничения:

1. PRIMARY KEY - набор полей (1 или более), значения которых образуют уникальную комбинацию и используются для однозначной идентификации записи в таблице. Не может содержать значение NULL;
2. CHECK - используется для ограничения множества значений, которые могут быть помещены в данный столбец;
3. UNIQUE - обеспечивает отсутствие дубликатов в столбце или наборе столбцов. Отличается от PRIMARY KEY тем что может содержать уникальное значение NULL;
4. FOREIGN KEY - защищает от действий, которые могут нарушить связи между таблицами. FOREIGN KEY в одной таблице указывает на PRIMARY KEY в другой. Поэтому данное ограничение нацелено на то, чтобы не было записей FOREIGN KEY, которым не отвечают записи PRIMARY KEY. Таким образом, FOREIGN KEY поддерживает ссылочную целостность данных.



Рис. 2.2 «Создание ограничение UNIQUE на столбец Photo в таблице Person»

## Создание запросов на выборку данных

Оператор языка SQL, SELECT - основная строительная конструкция для создания любого, простого или сложного запроса к базе данных. Без него, как и без фундамента для постройки, невозможно получить ни одну выборку данных из базы.

Самый простой запрос с оператором SELECT для выбора всех столбцов таблицы имеет следующий синтаксис:



Рис. 2.3 «Запрос на выборку всех значений в таблице Person»



Рис. 2.4 «Результат простого запроса SELECT»

Далее создадим запрос на более сложную выборку данных из базы данных с использованием агрегатной функции COUNT, которая возвращает количество найденных записей.

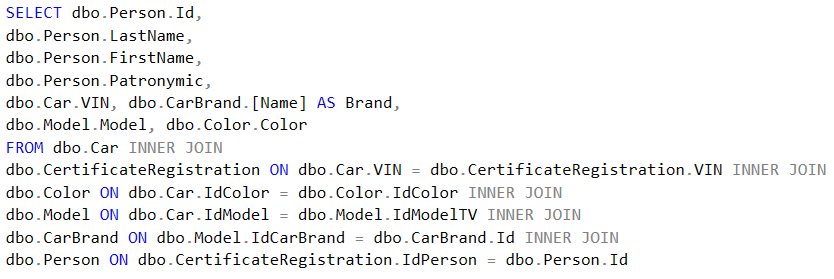


Рис. 2.5 «Запрос на выборку данных о людях и принадлежащим им автомобилях»

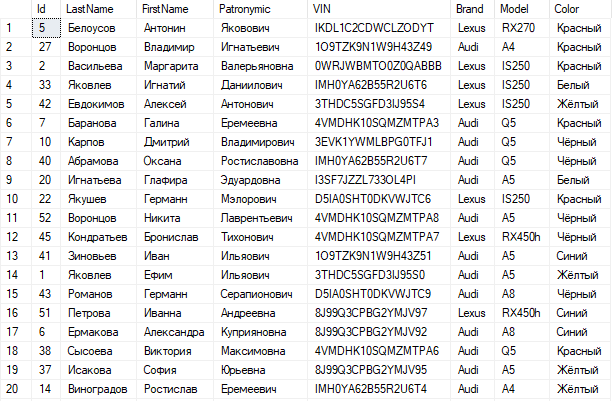


Рис. 2.6 «Результат сложного запроса SELECT»

## 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.[10]

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.

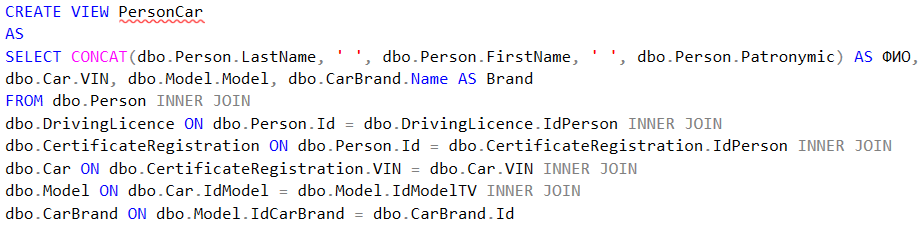


Рис. 2.7 «Запрос на создание представления который выведет информацию о людях и их транспортным средствам»



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.[13]

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.

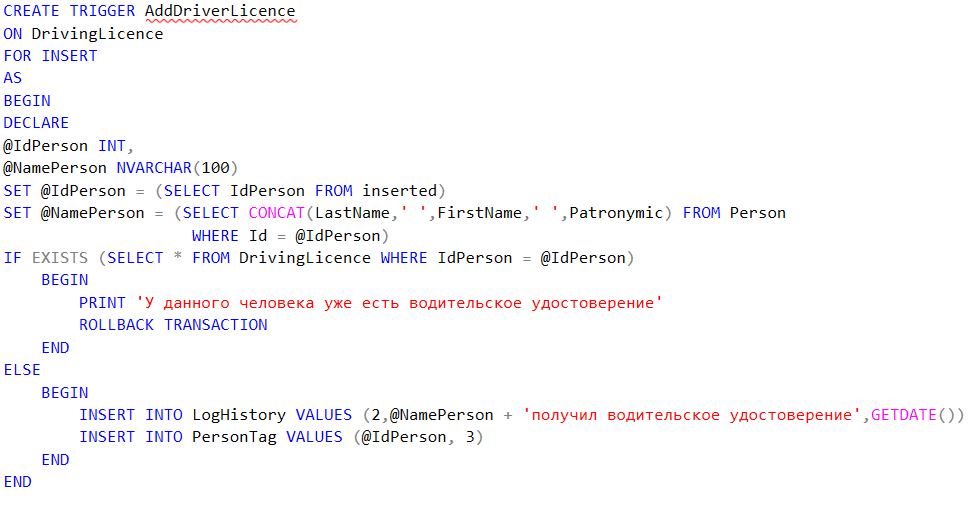


Рис. 2.11 «Создание триггера на добавление водительских прав»

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, онотменяет транзакцию. В обратном случае, запись проходит, а так же триггер добавляет в таблицу LogHistory(История) запись о том когда и кто, получил водительское удостоверение, и добавляет ему тег «Новичок» в таблице PersonTag.

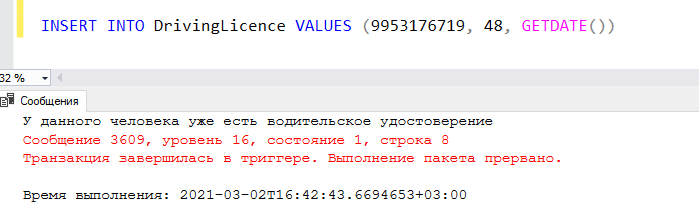


Рис. 2.12 «Результат работы триггера»

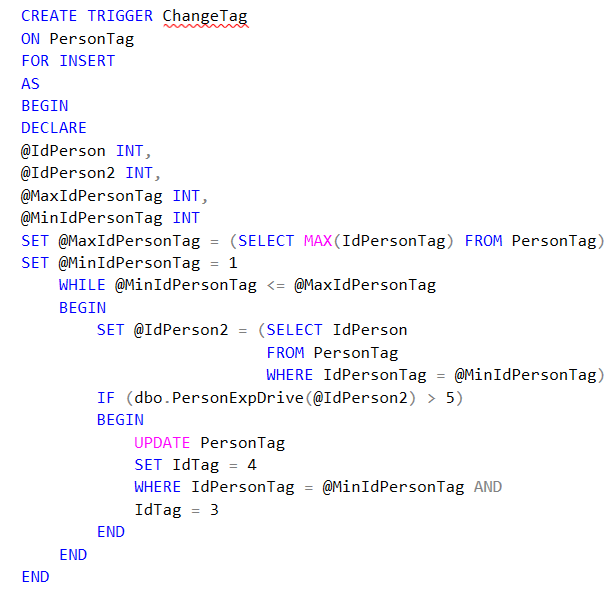


Рис. 2.13 «Создание триггера на обновление тэга»

Данный триггер срабатывает на добавление записи в таблицу «PersonTag». Его функция заключается в том чтобы менять тег водителя с «Новичок» на «Опытный», если его стаж вождения больше 5 лет. Для этого я использовал цикл WHILE.

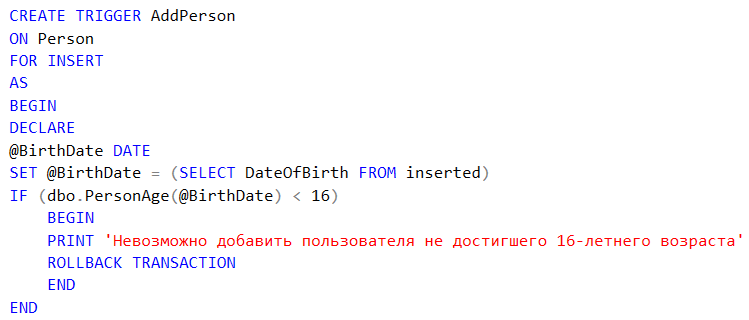


Рис. 2.14 «Создание триггера на добавление водителя»

Этот триггер срабатывает на добавление записи в таблицу «Person». 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.

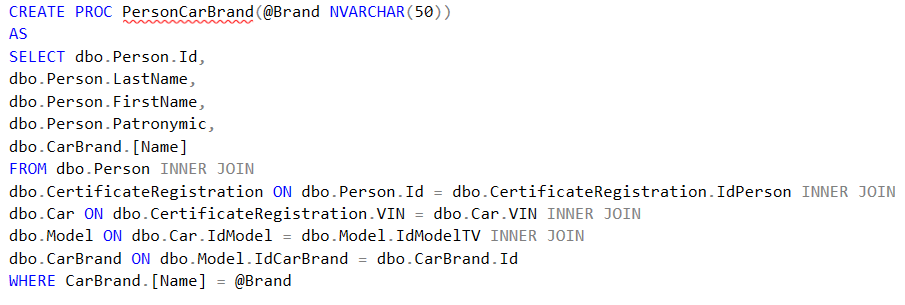


Рис. 2.14 «Создание процедуры которая выводит всех владельцев авто по заданной марке»



Рис. 2.15 «Результат работы процедуры по марке LEXUS»

Данная процедура принимает значение (марку автомобиля) типа NVARCHAR, и возвращает таблицу с ФИО владельцев автомобилей марки «LEXUS»

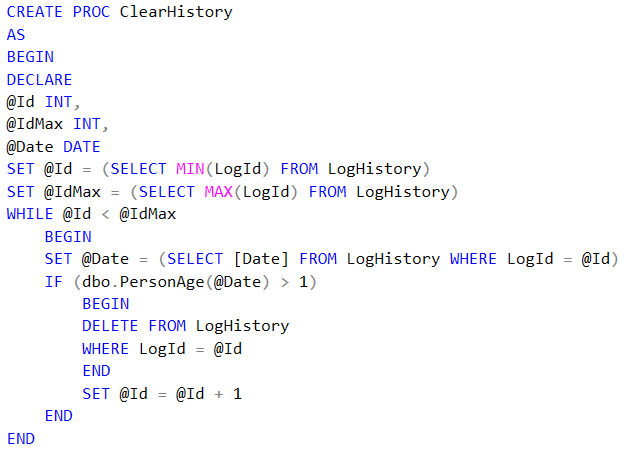


Рис. 2.16 «Создание процедуры на очищение истории»

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.

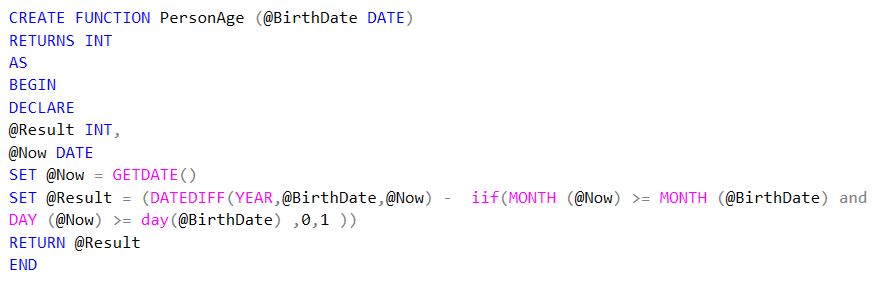


Рис. 2.16 «Создание функции разницы лет»

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

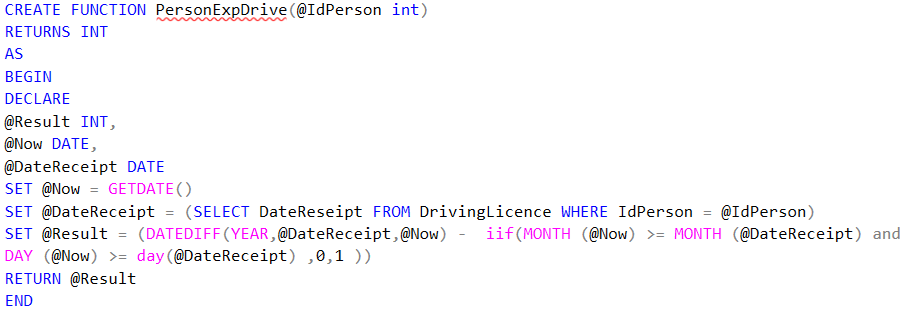


Рис. 2.17 «Создание функции для подсчёта водительского стажа»

Данная функция используется для подсчёта водительского стажа в годах.

# 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. Диаграмма базы данных в СУБД MS SQL Server

