**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ АСТРАХАНСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение  
Астраханской области «Астраханский колледж вычислительной техники»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **К защите допустить:** | | | | | | | |
| **Старший методист УМС** | | | | | | | |
|  |  | | | **Л.О.Земцова** | | | |
| **«** |  | **»** |  | | **20** |  | **г.** |

**курсовая работа**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| по МДК | 04.01 | Технология разработки и защиты баз данных | | | | | | |
|  | | | | | | | | |
|  | | | 6 семестр | | | | | |
| Тема курсовой работы | | | Разработка базы данных | | | | | |
|  | | | ***«***Комиссионный автосалон» | | | | | |
|  | | |  | | | | | |
| АКВТ.09.02.07.КР00.00ПЗ | | | | | | | | |
| Листов: | | | | 39 |  | | | |
|  | |  | | | | | | |
|  | |  | | | | | | |
| Выполнил | |  | | | | Студент гр. | | ПБ – 31 |
|  | |  | | | |  | | |
|  | |  | | | |  |  | |
| Проверил | |  | | | | Преподаватель | | |
|  | |  | | | | Мулеев Р.Р. | | |

**2022**

**Государственное бюджетное профессиональное образовательное учреждение**

**Астраханской области «Астраханский колледж вычислительной техники»**

|  |
| --- |
| УТВЕРЖДЕНО |
| Цикловой комиссией |
| специальности 09.02.07 |
| ПЦК \_\_\_\_\_\_\_\_\_ Ю.С. Андрианова |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
| Специальность | *09.02.07 Информационные системы и программирование* | | |
| МДК 04.01 | *Технология разработки и защиты баз данных* | | |
| Группа | *ПБ– 31* | | |
| **ЗАДАНИЕ** | | | |
| **на курсовое проектирование** | | | |
| ***[ДАННЫЕ УДАЛЕНЫ]*** | | | |
| (фамилия, имя, отчество студента) | | | |
| Тема курсового проекта | | ***Разработка БД*** | |
| ***«Комиссионный автосалон»*** | | | |

Содержание пояснительной записки курсового проекта

Содержание

Введение

1. Проектирование базы данных
2. Анализ и описание предметной области информационной системы
3. Проектирование концептуальной модели
4. Построение логической модели, схемы базы данных
5. Выбор СУБД
6. Реализация базы данных
7. Физическое проектирование
8. Написание исходного кода БД
9. Написание запросов, функций и хранимых процедур

Заключение

Список использованных источников

Приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Р.Р. Мулеев |
| Студент |  |  |  |

**Аннотация**

Данный документ на разработку базы данных имеет 39 страницы, 4 раздела, 3 рисунка, 5 источника использованной литературы. Таблицы и приложения отсутствуют.

**Содержание**

[Введение 5](#_Toc105341822)

[1. Проектирование БД 6](#_Toc105341823)

[1.1 Анализ и описание предметной области информационной системы 6](#_Toc105341824)

[1.2 Проектирование концептуальной модели 7](#_Toc105341825)

[1.3 Построение логической модели, схемы базы данных 8](#_Toc105341826)

[1.4 Выбор СУБД 15](#_Toc105341827)

[2 Реализация базы данных 17](#_Toc105341828)

[2.1 Физическое проектирование БД 17](#_Toc105341829)

[2.2 Написание исходного кода БД 17](#_Toc105341830)

[2.3 Написание запросов, функций и хранимых процедур 23](#_Toc105341831)

[Заключение 38](#_Toc105341832)

[Список используемой литературы 39](#_Toc105341833)

# Введение

Современные условия для работы по обслуживанию достигло того уровня, когда весь документооборот осуществляется на компьютерах, и вся информация храниться в специальных базах, потому растёт значимость информационных систем, которые позволяют обеспечить информационную поддержку процессов принятия решений. Базы данных являются одним из основных элементов большинства информационных систем.

База данных (далее БД) - это организованная структура, в которой в специальном формате хранится информация.

Система управления базой данных (далее СУБД) - это программа, с помощью которой в компьютер вводится информация, просматривается, сортируется и т.д.

БД «Комиссионный автосалон» предназначена для структуризации информации об клиентах, их автомобилей, договорах на продажу этих автомобилей и дилеров.

# Проектирование БД

## Анализ и описание предметной области информационной системы

Комиссионный автосалон - это место, где осуществляется торговля автотранспортом, через посредника, за определенное вознаграждение.

Целью издательства является:

* продажа поддержанных автомобилей;
* получение прибыли с продаж.

Предметная область представляет собой базу данных, направлена на информационную поддержку комиссионного автосалона. База данных даёт возможность проведение аналитики и создание отчётности по деятельности всей организации, а также каждого договора, автоматизирует некоторые процессы. Ввиду всех возможностей базы данных возникает необходимость в автоматизации информационных процессов.

Входные документы:

* документы автомобиля;
* документы клиента.

Выходные документы:

* договор на продажу.

База данных создана для сотрудников комиссионного автосалона, которые могут просматривать, вносить, изменять, удалять информацию в базе данных в полном объёме.

БД должна осуществлять хранение данных о:

* клиентах (код клиента, фамилия, имя, отчество, город, адрес, контактный телефон);
* дилерах (код дилера, фамилия, имя, отчество, фотография, домашний адрес, телефон);
* договорах (код договора, код клиента, код дилера, дата заключения договора, марка автомобиля, фото автомобиля, дата выпуска, пробег, дата продажи, цена продажи, размер комиссионных, примечание, текст договора в виде файла).

Необходимо предусмотреть:

* вывод информации о клиентах и предлагаемых ими автомобилях;
* вывод информации о деятельности дилеров (с перечислением договоров) и клиентах, которые они обслуживают;
* подсчет количества договоров, заключенных с каждым клиентом;
* подсчет количества договоров, обслуживаемых каждым дилером;
* выдачу данных дилера (дату заключения договора, данные клиента, отметку о продаже) обо всех договорах и договорах за некоторый промежуток времени или договорах, удовлетворяющих определенному условию.

## Проектирование концептуальной модели

Концептуальная модель - модель предметной области, состоящей из перечня взаимосвязанных понятий, используемых для описания этой области, вместе со свойствами и характеристиками, классификацией этих понятий, по типам, ситуациям, признакам в данной области и законов протекания процессов в ней. Концептуальная модель относится к моделям, которые формируются после процесса концептуализации или обобщения. Концептуальные модели часто являются абстракциями вещей в реальном мире.

Сущности:

1. Клиент - содержит информацию о клиенте, желающий продать автомобиль.

Атрибуты: ФИО, номер телефона, адрес.

1. Автомобиль - содержит информацию о автомобиле клиента.

Атрибуты: марка, телефон, модель, пробег.

1. Договор - содержит информацию о договоре.

Атрибуты: дата подписания, комиссионные, дата продажи.

1. Дилер - содержит информацию о дилере.

Атрибуты: ФИО, номер телефона, адрес.

Связи:

1. Клиент заключает договор. Клиент может заключить много договоров, а у договора может быть один клиент (связь 1 к многим).
2. Автомобиль принадлежит клиенты. Автомобиль может принадлежать только одному клиенту, а у клиента может быть много автомобилей (связь 1 к многим).
3. Дилер облаживает договор. Дилер может облаживать много договоров, а у договора может быть только один дилер (связь 1 к многим).
4. Автомобиль обосновывает договор. Автомобиль может обосновывать несколько договор, а у договора может быть только одby автомобиль (связь 1 к многим).

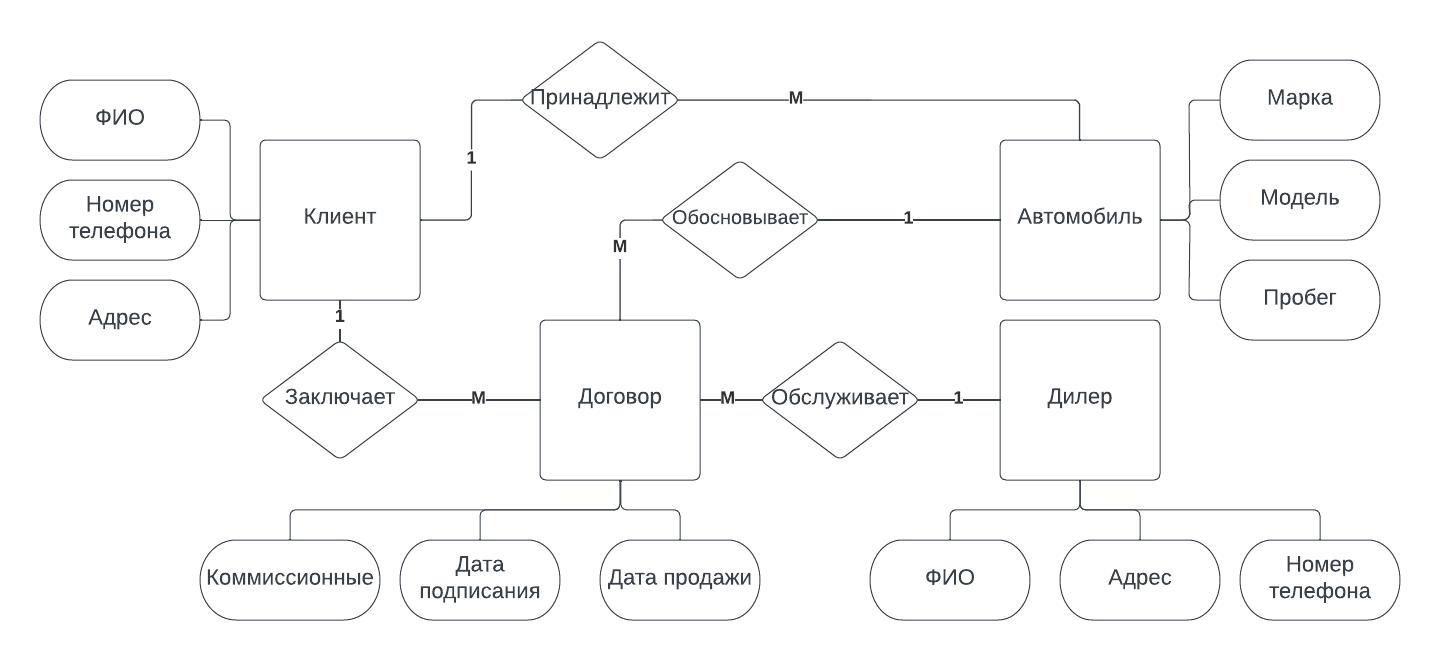


Рисунок 1 – Концептуальная модель

## Построение логической модели, схемы базы данных

Логические модели данных представляют собой абстрактную структуру области информации. Они часто носят схематический характер и чаще всего используются в бизнес-процессах, которые стремятся захватить вещи, важные для организации, и как они связаны друг с другом. После проверки и утверждения логическая модель данных может стать основой физической модели данных и сформировать дизайн базы данных. модели должны основываться на структурах, определенных в предыдущей концептуальной модели данных, поскольку это описывает семантику информационного контекста, которую логическая модель также должна отражать.

1. Город (City):
   1. Идентификатор города (CityId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название (Name):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Документы автомобиля (CarDocument):
   1. Идентификатор документов автомобиля (CarDocumentId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор автомобиля (CarId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Car (поле CarId).
  1. Паспорт транспортного средства (PTS):
* нулевое значение: not null;
* тип поля: nvarchar;
* уникальное поле;
* маска ввода: формат номера паспорта транспортного средства.
  1. ОСАГО(OSAGO):
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: формат номера ОСАГО;
* уникальное поле.
  1. Свидетельство о регистрации авто (SOP):
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: формат номера свидетельства о регистрации авто;
* уникальное поле.
  1. Документ о приобретении автомобиля (DKP):
* нулевое значение: not null;
* тип поля: varbinary.

1. Автомобиль (Car):
   1. Идентификатор автомобиля (CarId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор бренда (BrandId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Brand (поле BrandId).
  1. Модель (Model):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Год выпуска (YearRelease):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 1970.
  1. Пробег (Mileage):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: больше 0.
  1. Идентификатор клиента (ClientId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Client (поле ClientId).
  1. Идентификатор автомобиля (VIN):
* нулевое значение: not null;
* тип поля: nvarchar;
* уникальное поле.

1. Договор (Сontract):
   1. Идентификатор договора (СontractId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор автомобиля (CarId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Car (поле CarId).
  1. Идентификатор дилера (DealerId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Dealer (поле DealerId).
  1. Комиссия (Commission):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.
  1. Примечание (Note):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Файл договора (СontractFile):
* нулевое значение: not null;
* тип поля: varbinary.
  1. Сумма договора (PriceContact):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: больше 0.
  1. Идентификатор дилера (StatusId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Status (поле StatusId).
  1. Сумма продажи (PriceSale):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше или равно 0.
  1. Дата статуса (DateStatus):
* нулевое значение: not null;
* тип поля: date.
  1. Дата статуса (DateStartContract):
* нулевое значение: not null;
* тип поля: date.
  1. Дата статуса (DateFinishContract):
* нулевое значение: not null;
* тип поля: date.

1. Дилер (Dealer):
   1. Идентификатор дилера (DealerId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: 10 цифр.
  1. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута адрес.
  1. Фотография дилера (Image):
* нулевое значение: not null;
* тип поля: varbinary.

1. Клиент (Client):
   1. Идентификатор клиента (ClientId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: 10 цифр.
  1. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута адрес.

1. Статус договора (Status):
   1. Идентификатор статуса договора (StatusId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование (Name):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Документы клиента (ClientDocument):
   1. Идентификатор документов клиента (ClientDocumentId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор клиента (ClientId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Client (поле ClientId).
  1. Паспорт (Passport)
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: формат серии и номера паспорта.
  1. ИНН (INN)
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: формат номера ИНН.

1. Бренд (Brand):
   1. Идентификатор бренда (BrandId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название (Name):
* нулевое значение: not null;
* тип поля: nvarchar.

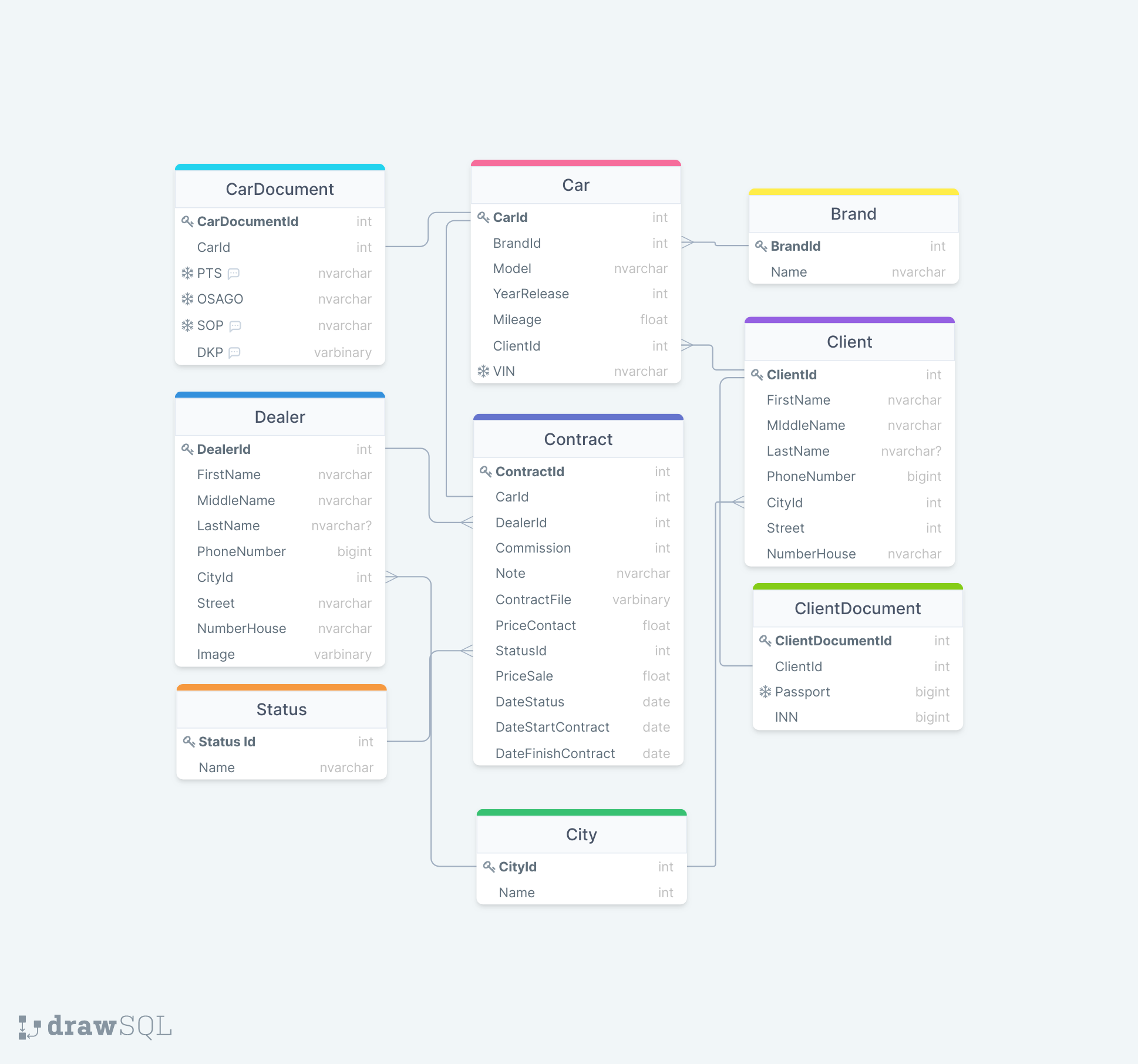


Рисунок 2 – Логическая модель

## Выбор СУБД

СУБД предназначена для централизованного управления базой данных в интересах всех работающих в этой системе. Используемые в настоящее время СУБД, обладающих средствами обеспечения целостности данных и надёжной безопасности, что даёт возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование.

В качестве средства реализации БД была выбрана система Microsoft SQL Server Management Studio 2019 (далее MSSQL).

MSSQL– это мощная и надежная система управления данными, обеспечивающая множество функций, защиту данных и высокую производительность для внедренных приложений-клиентов и локальных хранилищ данных. MSSQL предназначен для упрощенного развертывания и быстрого создания прототипов; его можно получить бесплатно и свободно распространять вместе с приложениями. Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах.

MSSQL упрощает развёртывание, управление и оптимизацию данных предприятия и аналитических приложений. Как платформа управления данными предприятия, он предоставляет единую консоль управления, которая позволяет администраторам данных, находящимся в любом месте вашей организации, отслеживать, управлять и настраивать все базы данных и связанные службы по всему предприятию. Он предоставляет расширяемую инфраструктуру управления, которая может быть легко запрограммирована при помощи SQL Management Objects, позволяя пользователям переделывать и расширять их среду управления и независимым поставщикам программных продуктов создавать дополнительные инструменты и функциональность для дальнейшего расширения возможностей, поставляемых по умолчанию.

Средством, связывающим клиента с сервером, является язык SQL - язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

# Реализация базы данных

## Физическое проектирование БД

Физическая модель данных представляет собой модель, в которой она описывает, как данные хранятся в памяти компьютера, как они разбросаны и упорядочены в памяти и как они будут извлечены из памяти.

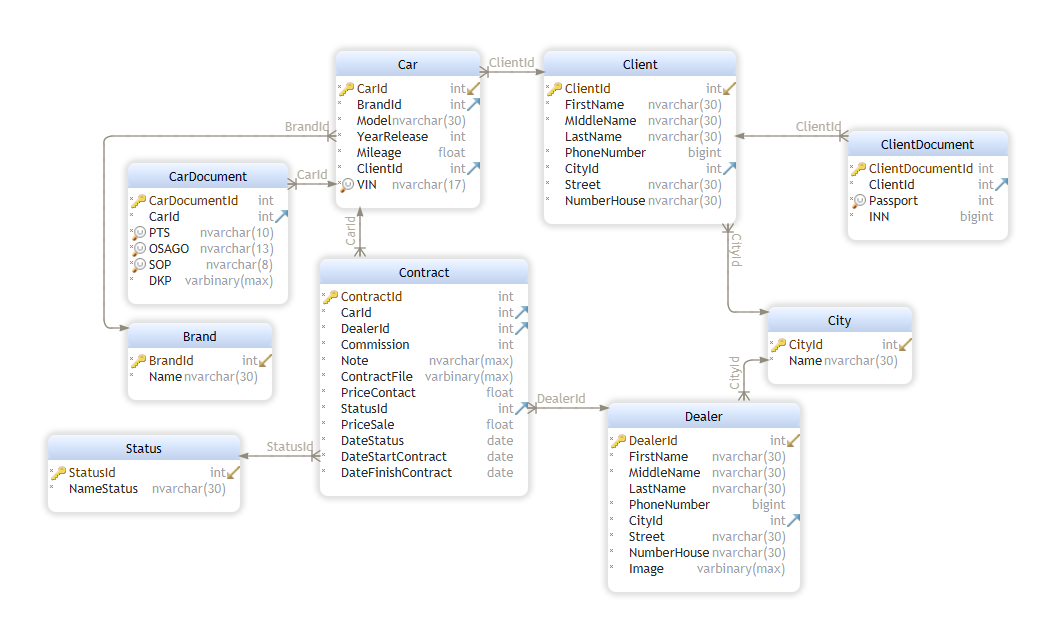


Рисунок 3 – Диаграмма базы данных

## Написание исходного кода БД

* 1. Car

Данная таблица создана с целью хранения информации о автомобиле.

Листинг:

CREATE TABLE Car (

CarId INT identity(1,1) NOT NULL,

BrandId INT NOT NULL,

Model NVARCHAR(30) NOT NULL,

YearRelease INT NOT NULL,

Mileage FLOAT NOT NULL,

ClientId INT NOT NULL,

VIN NVarchar(17) NOT NULL

);

ALTER TABLE

Car ADD CONSTRAINT car\_carid\_primary PRIMARY KEY( CarId );

CREATE UNIQUE INDEX car\_vin\_unique ON Car ( VIN );

ALTER TABLE

Car ADD CONSTRAINT car\_clientid\_foreign FOREIGN KEY ( ClientId ) REFERENCES Client ( ClientId );

ALTER TABLE

Car ADD CONSTRAINT car\_brandid\_foreign FOREIGN KEY (BrandId ) REFERENCES Brand ( BrandId );

alter table

Car ADD CONSTRAINT Car\_check\_YearRelease CHECK (YearRelease > 1970)

alter table

Car ADD CONSTRAINT Car\_check\_Mileage CHECK (Mileage > 0)

alter table

Car ADD CONSTRAINT Car\_check\_VIN CHECK (DATALENGTH(CAST(VIN AS varchar(17))) = 17)

* 1. Client

Данная таблица создана с целью хранения информации о клиенте.

Листинг:

CREATE TABLE Client (

ClientId INT identity(1,1) NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MIddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NOT NULL,

PhoneNumber BIGINT NOT NULL,

CityId INT NOT NULL,

Street NVARCHAR(30) NOT NULL,

NumberHouse NVARCHAR(30) NOT NULL

);

ALTER TABLE

Client ADD CONSTRAINT client\_clientid\_primary PRIMARY KEY( ClientId );

ALTER TABLE

Client ADD CONSTRAINT client\_cityid\_foreign FOREIGN KEY( CityId ) REFERENCES City ( CityId );

alter table

Client ADD CONSTRAINT Client\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10)

* 1. Dealer

Данная таблица создана с целью хранения информации о дилере.

Листинг:

CREATE TABLE Dealer (

DealerId INT identity(1,1) NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MiddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NOT NULL,

PhoneNumber BIGINT NOT NULL,

CityId INT NOT NULL,

Street NVARCHAR(30) NOT NULL,

NumberHouse NVARCHAR(30) NOT NULL,

Image VARBINARY(MAX) NOT NULL

);

ALTER TABLE

Dealer ADD CONSTRAINT dealer\_dealerid\_primary PRIMARY KEY(DealerId);

ALTER TABLE

Dealer ADD CONSTRAINT dealer\_cityid\_foreign FOREIGN KEY(CityId) REFERENCES City (CityId);

alter table

Dealer ADD CONSTRAINT Dealer\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10)

* 1. City

Данная справочная таблица создана с целью хранения информации о городах

Листинг:

CREATE TABLE City(

CityId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL

);

ALTER TABLE

City ADD CONSTRAINT city\_cityid\_primary PRIMARY KEY(CityId);

* 1. Status

Данная справочная таблица создана с целью хранения информации о статусе договора.

Листинг:

CREATE TABLE Status(

StatusId INT identity(1,1) NOT NULL,

NameStatus NVARCHAR(30) NOT NULL

);

ALTER TABLE

Status ADD CONSTRAINT status\_statusid\_primary PRIMARY KEY(StatusId);

* 1. CarDocument

Данная таблица создана с целью хранения информации о документах на автомобиль.

Листинг:

CREATE TABLE CarDocument (

CarDocumentId INT identity(1,1) NOT NULL,

CarId INT NOT NULL,

PTS NVARCHAR(10) NOT NULL,

OSAGO NVARCHAR(13) NOT NULL,

SOP NVARCHAR(8) NOT NULL,

DKP VARBINARY(MAX) NOT NULL

);

ALTER TABLE

CarDocument ADD CONSTRAINT cardocument\_cardocumentid\_primary PRIMARY KEY( CarDocumentId );

CREATE UNIQUE INDEX cardocument\_pts\_unique ON

CarDocument ( PTS );

CREATE UNIQUE INDEX cardocument\_osago\_unique ON

CarDocument ( OSAGO );

CREATE UNIQUE INDEX cardocument\_sop\_unique ON

CarDocument ( SOP );

ALTER TABLE

CarDocument ADD CONSTRAINT cardocument\_carid\_foreign FOREIGN KEY( CarId ) REFERENCES Car ( CarId );

alter table

CarDocument ADD CONSTRAINT ClientDocument\_check\_PTS CHECK (PTS like '[0-9][0-9][А-Я][А-Я][0-9][0-9][0-9][0-9][0-9][0-9]')

alter table

CarDocument ADD CONSTRAINT ClientDocument\_check\_OSAGO CHECK (OSAGO like '[А-Я][А-Я][А-Я][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

alter table

CarDocument ADD CONSTRAINT ClientDocument\_check\_SOP CHECK (SOP like '[А-Я][0-9][0-9][0-9][А-Я][А-Я][0-9][0-9]')

* 1. ClientDocument

Данная таблица создана с целью хранения информации о документах клиента.

Листинг:

CREATE TABLE ClientDocument (

ClientDocumentId INT identity(1,1) NOT NULL,

ClientId INT NOT NULL,

Passport INT NOT NULL,

INN BIGINT NOT NULL

);

ALTER TABLE

ClientDocument ADD CONSTRAINT clientdocument\_clientdocumentid\_primary PRIMARY KEY( ClientDocumentId );

CREATE UNIQUE INDEX clientdocument\_passport\_unique ON

ClientDocument ( Passport );

ALTER TABLE

ClientDocument ADD CONSTRAINT clientdocument\_clientid\_foreign FOREIGN KEY( ClientId ) REFERENCES Client ( ClientId );

alter table

ClientDocument ADD CONSTRAINT ClientDocument\_check\_Passport CHECK (DATALENGTH(CAST(Passport AS varchar(10))) = 10)

alter table

ClientDocument ADD CONSTRAINT ClientDocument\_check\_INN CHECK (DATALENGTH(CAST(INN AS varchar(12))) = 12)

* 1. Сontract

Данная таблица создана с целью хранения информации о договорах.

Листинг:

CREATE TABLE Сontract (

СontractId INT identity(1,1) NOT NULL,

CarId INT NOT NULL,

DealerId INT NOT NULL,

Commission INT NOT NULL,

Note NVARCHAR(Max) NOT NULL,

СontractFile VARBINARY(MAX) NOT NULL,

PriceContact FLOAT NOT NULL,

StatusId INT NOT NULL,

PriceSale FLOAT default 0 NOT NULL,

DateStatus DATE NOT NULL,

DateStartContract DATE NOT NULL,

DateFinishContract DATE NOT NULL

);

ALTER TABLE

Сontract ADD CONSTRAINT Сontract\_Сontractid\_primary PRIMARY KEY( СontractId );

ALTER TABLE

Сontract ADD CONSTRAINT Сontract\_carid\_foreign FOREIGN KEY( CarId ) REFERENCES Car ( CarId );

ALTER TABLE

Сontract ADD CONSTRAINT Сontract\_dealerid\_foreign FOREIGN KEY( DealerId ) REFERENCES Dealer ( DealerId );

ALTER TABLE

Сontract ADD CONSTRAINT Сontract\_statusid\_foreign FOREIGN KEY( StatusId ) REFERENCES Status ( StatusId );

alter table

Сontract ADD CONSTRAINT Сontract\_check\_Commission CHECK (Commission > 0)

alter table

Сontract ADD CONSTRAINT Сontract\_check\_PriceContact CHECK (PriceContact > 0)

alter table

Сontract ADD CONSTRAINT Сontract\_check\_PriceSale CHECK (PriceSale >= 0)

* 1. Brand

Данная таблица создана с целью хранения информации о брендах.

Листинг:

CREATE TABLE Brand (

BrandId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL

);

ALTER TABLE

Brand ADD CONSTRAINT brand\_brandid\_primary PRIMARY KEY( BrandId );

## Написание запросов, функций и хранимых процедур

1. CarAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Car и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Car по названию и типу.

Листинг процедуры:

create PROCEDURE CarAction (

@Action tinyint,

@CarId INT ,

@BrandId INT ,

@Model NVARCHAR(30) ,

@YearRelease INT ,

@Mileage FLOAT ,

@ClientId INT ,

@VIN NVARCHAR(17)

)

As

Begin

if(@Action = 1)

delete Car where CarId = @CarId

else

if(@Action = 2)

insert Car (BrandId,Model,YearRelease,Mileage,ClientId,VIN)

values (@BrandId,@Model,@YearRelease,@Mileage,@ClientId,@VIN)

else

if(@Action = 3)

UPDATE [dbo].[Car]

SET BrandId = @BrandId

,Model = @Model

,YearRelease = @YearRelease

,Mileage = @Mileage

,ClientId = @ClientId

,VIN = @VIN

where CarId = @CarId

end

Пример использования:

* Удаление: execute CarAction 1,2,0,'',0,0,0,null

При удалении важно указать первым (@Action) 1 и вторым (@CarId) существующий CarId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute CarAction 2,1,1,'',12,213,43,’SD2133GCEG2143FVD’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute CarAction 3,1,1,'',12,213,43,’SD2133GCEG2143FVD’

При обновлении важно указать первым (@Action) 3 и вторым (@CarId) существующий CarId, а остальное заполнить нужными значениями.

1. ClientAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Client и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Client по названию и типу.

Листинг процедуры:

create PROCEDURE ClientAction (

@Action tinyint,

@ClientId INT ,

@FirstName NVARCHAR(30),

@MIddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@PhoneNumber BIGINT,

@CityId INT,

@Street NVARCHAR(30),

@NumberHouse NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Client where ClientId = @ClientId

else

if(@Action = 2)

insert Client (FirstName,MIddleName,LastName,PhoneNumber ,CityId ,Street,NumberHouse)

values (@FirstName,@MIddleName,@LastName,@PhoneNumber ,@CityId ,@Street,@NumberHouse)

else

if(@Action = 3)

UPDATE [dbo].[Client]

SET FirstName = @FirstName

,MIddleName = @MIddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

,CityId = @CityId

,Street = @Street

,NumberHouse = @NumberHouse

where ClientId = @ClientId

end

Пример использования:

* Удаление: execute ClientAction 1,2,’’,’’,’’,NULL,0,’’,’’

При удалении важно указать первым (@Action) 1 и вторым (@ClientID) существующий ClientId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ClientAction 2,1, ’qw’,’er’,’ty’,89123454344,0,’ds’,’rwe’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ClientAction 3,1, ’qw’,’er’,’ty’,89123454344,0,’ds’,’rwe’

При обновлении важно указать первым (@Action) 3 и вторым (@ClientId) существующий ClientId, а остальное заполнить нужными значениями.

1. DealerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Dealer и включает следующие переменные: @Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Dealer по названию и типу.

Листинг процедуры:

create PROCEDURE DealerAction (

@Action tinyint,

@DealerId INT ,

@FirstName NVARCHAR(30),

@MiddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@PhoneNumber BIGINT,

@CityId INT,

@Street NVARCHAR(30),

@NumberHouse NVARCHAR(30),

@Image VARBINARY(MAX)

)

As

Begin

if(@Action = 1)

delete Dealer where DealerId = @DealerId

else

if(@Action = 2)

insert Dealer (FirstName, MiddleName, LastName, PhoneNumber, CityId, Street, NumberHouse, Image)

values (@FirstName, @MiddleName, @LastName, @PhoneNumber, @CityId, @Street, @NumberHouse, @Image)

else

if(@Action = 3)

UPDATE [dbo].[Dealer]

SET FirstName = @FirstName

,MIddleName = @MIddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

,CityId = @CityId

,Street = @Street

,NumberHouse = @NumberHouse

,Image = @Image

where DealerId = @DealerId

end

Пример использования:

* Удаление: execute DealerAction 1,2, ’’,’’,’’,NULL,0,’’,’’

При удалении важно указать первым (@Action) 1 и вторым (@DealerID) существующий DealerId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute DealerAction 2,1,’qw’,’er’,’ty’,89123454344,0,’ds’,’rwe’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute DealerAction 3,1, ’qw’,’er’,’ty’,89123454344,0,’ds’,’rwe’

При обновлении важно указать первым (@Action) 3 и вторым (@DealerId) существующий DealerId, а остальное заполнить нужными значениями.

1. CityAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу City и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице City по названию и типу.

Листинг процедуры:

Create PROCEDURE CityAction (

@Action tinyint,

@CityId INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete City where CityId = @CityId

else

if(@Action = 2)

insert City (Name)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[City]

SET Name = @Name

where CityId = @CityId

end

Пример использования:

* Удаление: execute CityAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@CityID) существующий CityId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute CityAction 2,1,’Астрахань’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute CityAction 3,1,’Казань’

При обновление важно указать первым (@Action) 3 и вторым (@CityId) существующий CityId, а остальное заполнить нужными значениями.

1. StatusAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Status и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Status по названию и типу.

Листинг процедуры:

Create PROCEDURE StatusAction (

@Action tinyint,

@StatusId INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Status where StatusId = @StatusId

else

if(@Action = 2)

insert Status (NameStatus)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[Status]

SET Name = @Name

where StatusId = @StatusId

end

Пример использования:

* Удаление: execute StatusAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@StatusId) существующий StatusId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute StatusOrdersAction 2,1,’’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute StatusAction 3,1,’’

При обновлении важно указать первым (@Action) 3 и вторым (@StatusId) существующий StatusId, а остальное заполнить нужными значениями.

1. CarDocumentAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу CarDocument и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице CarDocument по названию и типу.

Листинг процедуры:

create PROCEDURE CarDocumentAction (

@Action tinyint,

@CarDocumentId INT ,

@CarId INT,

@PTS NVARCHAR(30),

@OSAGO NVARCHAR(30),

@SOP NVARCHAR(30),

@DKP INT

)

As

Begin

if(@Action = 1)

delete CarDocument where CarDocumentId = @CarDocumentId

else

if(@Action = 2)

insert CarDocument (CarId,PTS,OSAGO,SOP,DKP)

values (@CarId,@PTS,@OSAGO,@SOP,@DKP)

else

if(@Action = 3)

UPDATE [dbo].[CarDocument]

SET CarId = @CarId

,PTS = @PTS

,OSAGO = @OSAGO

,SOP = @SOP

,DKP = @DKP

where CarDocumentId = @CarDocumentId

end

Пример использования:

* Удаление: execute CarDocumentAction 1,2,0,null, null, null, null

При удалении важно указать первым (@Action) 1 и вторым (@CarDocumentID) существующий CarDocumentId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute CarDocumentAction 2,1, 1, ’qw423423’, ’21fr3223212’, ’sad32432’, ’sdasd’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute CarDocumentAction 3,1, 1, ’qw423423’, ’21fr3223212’, ’sad32432’, ’sdasd’

При обновлении важно указать первым (@Action) 3 и вторым (@CarDocumentId) существующий CarDocumentId, а остальное заполнить нужными значениями.

1. ClientDocumentAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу ClientDocument и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице ClientDocument по названию и типу.

Листинг процедуры:

Create PROCEDURE ClientDocumentAction (

@Action tinyint,

@ClientDocumentId INT ,

@ClientId INT,

@Passport INT,

@INN BIGINT

)

As

Begin

if(@Action = 1)

delete ClientDocument where ClientDocumentId = @ClientDocumentId

else

if(@Action = 2)

insert ClientDocument (ClientId,Passport,INN)

values (@ClientId,@Passport,@INN)

else

if(@Action = 3)

UPDATE [dbo].[ClientDocument]

SET ClientId = @ClientId

,Passport = @Passport

,INN = @INN

where ClientDocumentId = @ClientDocumentId

end

Пример использования:

* Удаление: execute ClientDocumentAction 1,2,0,0,0

При удалении важно указать первым (@Action) 1 и вторым (@ClientDocumentId) существующий ClientDocumentId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ClientDocumentAction 2,1,1,1221121212,111111111111

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ClientDocumentAction 3, 1,1,1221121212,111111111111

При обновлении важно указать первым (@Action) 3 и вторым (@ClientDocumentId) существующий ClientDocumentId, а остальное заполнить нужными значениями.

1. СontractAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Сontract и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Сontract по названию и типу.

Листинг процедуры:

Create PROCEDURE СontractAction (

@Action tinyint,

@СontractId INT ,

@CarId INT,

@DealerId INT,

@Commission INT,

@Note NVARCHAR(Max),

@СontractFile VARBINARY(MAX),

@PriceContact FLOAT,

@StatusId INT,

@PriceSale FLOAT,

@DateStatus DATE,

@DateStartContract DATE,

@DateFinishContract DATE

)

As

Begin

if(@Action = 1)

delete Сontract where СontractId = @СontractId

else

if(@Action = 2)

insert Сontract (CarId,DealerId,Commission,Note,СontractFile,PriceContact,StatusId,PriceSale,DateStatus,DateStartContract,DateFinishContract)

values (@CarId,@DealerId,@Commission,@Note,@СontractFile,@PriceContact,@StatusId,@PriceSale,@DateStatus,@DateStartContract,@DateFinishContract)

else

if(@Action = 3)

UPDATE [dbo].[Сontract]

SET CarId = @CarId

,DealerId = @DealerId

,Commission = @Commission

,Note = @Note

,СontractFile = @СontractFile

,PriceContact = @PriceContact

,StatusId = @StatusId

,PriceSale = @PriceSale

,DateStatus = @DateStatus

,DateStartContract = @DateStartContract

,DateFinishContract = @DateFinishContract

where СontractId = @СontractId

end

Пример использования:

* Удаление: execute СontractAction 1,2,0,0,0,’’,’’,0,0,0,0,null,null,null

При удалении важно указать первым (@Action) 1 и вторым (@СontractId) существующий СontractId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute СontractAction 2,1,1,1,1,’ad’,’file’,21,2,1,1,’12.12.2003’, ’12.12.2003’, ’12.12.2003’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute СontractAction 3,1,1,1,1,’ad’,’file’,21,2,1,1,’12.12.2003’, ’12.12.2003’, ’12.12.2003’

При обновлении важно указать первым (@Action) 3 и вторым (@СontractId) существующий СontractId, а остальное заполнить нужными значениями.

1. BrandAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Brand и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Brand по названию и типу.

Листинг процедуры:

Create PROCEDURE BrandAction (

@Action tinyint,

@BrandId INT ,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Brand where BrandId = @BrandId

else

if(@Action = 2)

insert Brand (Name)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[Brand]

SET Name = @Name

where BrandId = @BrandId

end

Пример использования:

* Удаление: execute BrandAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@BrandId) существующий BrandId, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute BrandAction 2,1, ‘BMW’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute BrandAction 3,1, ‘DMW’

При обновлении важно указать первым (@Action) 3 и вторым (@BrandId) существующий BrandId, а остальное заполнить нужными значениями.

1. ClientCountСar

Описание: процедура создана для вывода информации по количеству машин у клиента.

Листинг:

Create PROCEDURE ClientCountСar

As

Begin

select c.\*,CountСar from Client c

inner join (select

Car.ClientId,

Count(ct.CarId) as CountСar

from Car

inner join Сontract ct on Car.CarId = ct.CarId

group by Car.ClientId) a on a.ClientId = c.ClientId

end

Выходные данные: все поля таблицы Client, CountСar.

1. DealerCountСontract

Описание: процедура создана для вывода информации по количеству договоров у дилеров.

Листинг:

Create PROCEDURE DealerCountСontract

As

Begin

select D.\*,CountСontract from Dealer D

left join (select

d.DealerId,

Count(ct.СontractId) as CountСontract

from Dealer d

left join Сontract ct on d.DealerId = ct.DealerId

where StatusId = 1

group by d.DealerId

) a on a.DealerId = D.DealerId

end

Выходные данные: все поля таблицы Dealer, CountСontract

1. DealerClient

Описание: процедура создана для вывода информации по всем договорам.

Листинг:

Create PROCEDURE DealerClient

As

Begin

select D.\*, c.\* from Dealer D

inner join Сontract ct on ct.DealerId = D.DealerId

inner join Car on Car.CarId = ct.CarId

inner join Client c on c.ClientId = Car.ClientId

order by D.DealerId

end

Выходные данные: все поля таблицы Dealer, все поля таблицы Client.

1. СontractDealer

Описание: процедура создана для вывода информации по всем договорам за определённый период.

Листинг:

Create PROCEDURE СontractDealer

(

@DateStartContractStart date = '01.01.1970',

@DateStartContractFinish date

)

as

Begin

select D.DealerId, DateStartContract, c.\*, StatusId from Dealer D

inner join Сontract ct on ct.DealerId = D.DealerId

inner join Car on Car.CarId = ct.CarId

inner join Client c on c.ClientId = Car.ClientId

where DateStartContract between '' and ''

order by D.DealerId

end

Выходные данные: все поля таблицы Dealer, DateStartContract, все поля таблицы Client, StatusId.

1. checkInsertСontract

Триггер создан как ограничение таблицы Сontract на вставку и обновление полей с условием: дата подписания договора не может больше даты выполнения договора.

Листинг:

create trigger checkInsertСontract on Сontract

for insert, update

as

declare @DateStartContract date, @DateFinishContract date

Select

@DateStartContract=DateStartContract,

@DateFinishContract=DateFinishContract

from inserted

if (@DateFinishContract >= @DateStartContract)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

# Заключение

В данной курсовой работе был проведен детальный анализ предметной области комиссионный автосалон. Раскрыто само понятие БД и способы его построения в соответствии с основными этапами проектирования, осуществляемыми в рамках современных информационных технологий.

В конечном итоге, на основе MSSQL была разработана база данных для издательства.

Разрабатываемая БД поможет отслеживать информацию о каждом договоре, его составляющих (автомобиль, дилер и т.д.).

Использование БД в реальном автосалоне позволит автоматизировать деятельность сотрудников автосалоне. Создаст благоприятные условия труда для сотрудников, позволит всегда отражать актуальную информацию.

Поставленная цель курсовой работы достигнута, задачи решены в полном объеме.

# Список используемой литературы

1. <https://1avtoyurist.ru/kuplya-prodazha/prodazha/komissionnaya-prodazha-avtotransporta.html>
2. <https://prava.expert/avtoyurist/kuplya-prodazha/cherez-avtosalon-podvodnye-kamni-i-poryadok.html>
3. <https://auto.ru/docs/dkp/>
4. <https://info-comp.ru/programmirovanie/412-directory-transact-sql.html>
5. <https://accesshelp.ru/baza-dannyh-access-avtosalon/>