**Реферат**

Пояснительная записка курсового проекта содержит 42 страницы, 8 рисунков, 3 источника литературы, 3 приложения.

БИБЛИОТЕКА, БАЗА ДАННЫХ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, SQL, T-SQL, MICROSOFT SQL SERVER.

Основными целями курсового проекта являются: проектирование базы данных «Библиотека».

Пояснительная записка состоит из введения, пяти разделов, заключения.

Во введении представлена общая информация, дающая представление о предстоящей работе, определены цели.

В первом разделе рассматривается архитектура базы данных.

Во втором разделе представлен процесс и результаты разработки объектов базы данных.

В третьем разделе описывается создание процедур для экспорта и импорта.

В четвертом разделе представляется тестирование базы данных.

В пятом разделе представлено руководство пользователя.

В заключении представлены итоги курсового проектирования и задачи, которые были решены в ходе проектирования и разработки базы данных.

Содержание

[Введение 6](#_Toc469487356)

[1. Разработка модели базы данных 7](#_Toc469487357)

[2. Разработка необходимых объектов 8](#_Toc469487358)

[2.1. Таблицы 8](#_Toc469487359)

[2.2. Процедуры 8](#_Toc469487360)

[2.3. Пользователи 8](#_Toc469487361)

[3. Описание процедур импорта и экспорта 9](#_Toc469487362)

[3.1. Процедура импорта 9](#_Toc469487363)

[3.2. Процедура экспорта 9](#_Toc469487364)

[4. Тестирование базы данных 10](#_Toc469487365)

[5. Руководство пользователя 11](#_Toc469487366)

[5.1. Клиент 11](#_Toc469487367)

[5.2. Менеджер 12](#_Toc469487368)

[5.3. Администратор 12](#_Toc469487369)

[5.4. Гость 12](#_Toc469487370)

[6. Разграничение прав 13](#_Toc469487376)

[Заключение 14](#_Toc469487377)

[Список использованных источников 15](#_Toc469487378)

[ПРИЛОЖЕНИЕ А 16](#_Toc469487379)

[ПРИЛОЖЕНИЕ Б 18](#_Toc469487380)

[ПРИЛОЖЕНИЕ В 22](#_Toc469487381)

[ПРИЛОЖЕНИЕ Г. 28](#_Toc469487382)

# Введение

Современное общество невозможно представить без компьютера. Компьютеры настолько широко и глубоко внедрились в нашу жизнь, что очень трудно назвать какую-либо сферу деятельности человека, где бы они ни использовались. В связи с этим серьезные требования предъявляются как к аппаратной части современных компьютеров, так и к используемому программному обеспечению. В основном именно программное обеспечение, или, иными словами, программные продукты, обеспечивают возможность широкого использования компьютеров. Стоит лишь переустановить программное обеспечение компьютера или добавить какой-либо новый программный продукт, и можно будет решать на этом компьютере совершенно новые задачи.

Следовательно, используемые программные продукты должны соответствовать определенным критериям, обеспечивающим надежность работы компьютера и удобство работы пользователя.

В соответствии с заданием курсового проектирования следует создать базу данных (БД) «Автосалон» с использование программного проекта Microsoft SQL Server.

Сегодня трудно себе представить сколько-нибудь значимую информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных. Концепции и технологии БД складывались постепенно и всегда были тесно связаны с развитием систем автоматизированной обработки информации.

Проектирование базы данных «Автосалон» ведется для упрощения повседневной работы менеджеров компании. А так же, как приложение для клиентов автосалона.

# Разработка модели базы данных

Для создания, хранения и управления данными, была использованная реляционная СУБД Microsoft SQL Server. Ниже на рисунке 1.1 представлена модель базы данных (таблицы, отмеченные на них первичных и вторичных ключей) курсового проекта. База данных была нормализована.

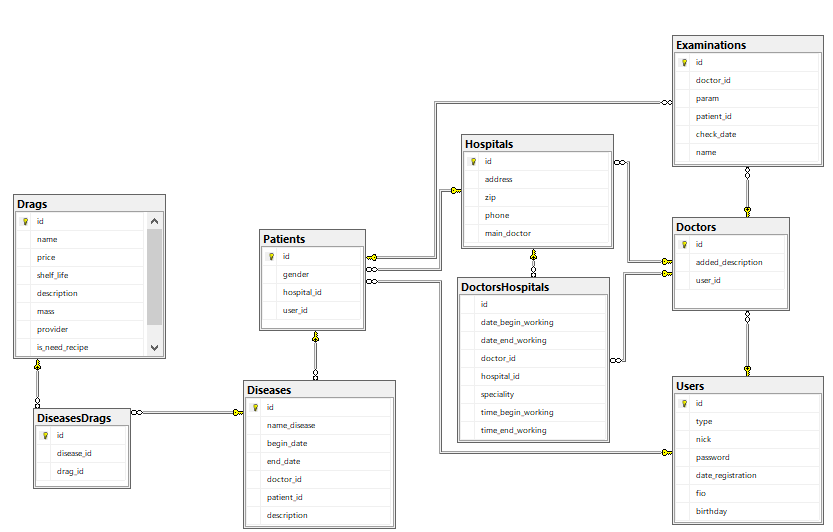


Рисунок 1.1 – Диаграмма базы данных

Более подробное описание назначения таблиц базы данных представлено ниже в разделе 2.1.

# 2. Разработка необходимых объектов

База данных данного курсового проекта содержит следующие объекты: таблиц, хранимых процедур, пользователей.

## 2.1. Таблицы

Данная база данных содержит 9 таблиц.

Таблица Users *–* хранит основные данные о пользователях приложения. В ней хранится логин, пароль, тип пользователя, дата регистрации, ФИО пользователя и дата рождения. Пользователи бывают 3 типов. Графическое описание представлено на рисунке 2.1 ниже.

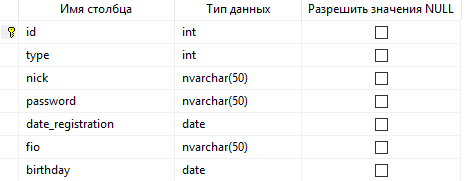


Рисунок 2.1. Таблица пользователей

Таблица Doctors *–* содержит информацию о докторах. Хранит описание о враче и вторичный ключ связанный с id пользователя. Графическое описание представлено на рисунке 2.2 ниже.

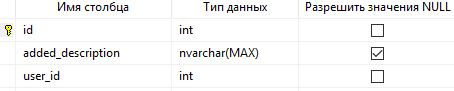


Рисунок 2.2. Таблица врачей

Таблица Hospitals *–* содержит информацию о больницах. В ней хранится адрес больницы, почтовый индекс больницы, телефон, вторичный ключ связанный с id доктора. Графическое описание представлено на рисунке 2.3 ниже.

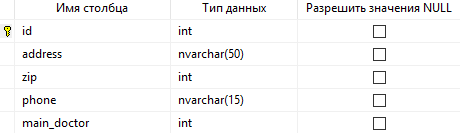


Рисунок 2.3. Таблица больниц

Таблица Patients *–* содержит информацию о пациентах. Хранит информацию о гендерной принадлежности, вторичный ключ связанный с id больницы, вторичный ключ связанный с id пользователя. Графическое описание представлено на рисунке 2.4 ниже.

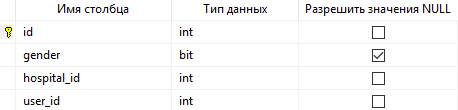


Рисунок 2.4. Таблица пользователей

Таблица Examinations *–* содержит информацию об обследованиях пациента. Хранит информацию о имени обследования, дата проведения обследования, параметры обследования (в виде xml), вторичный ключ связанный с id пациента, вторичный ключ связанный с доктором. Графическое описание представлено на рисунке 2.5 ниже.

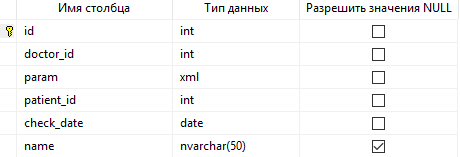


Рисунок 2.5. Таблица обследований

Таблица Diseases *­­–* содержит информацию о заболеваниях пациента. Хранит имя заболевания, дата начала заболевания, дата конца заболевания, его тестовое описание, вторичный ключ связанный с id пациента, вторичный ключ связанный с id доктора. Графическое описание представлено на рисунке 2.6 ниже.

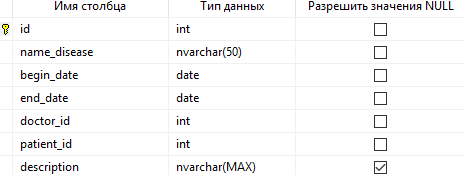


Рисунок 2.6. Таблица заболеваний

Таблиц Drags *–* содержит информацию о лекарствах. Хранит имя лекарства, цену, срок годности, описание, массу, имя поставщика и информацию о необходимости ресурса. Графическое описание представлено на рисунке 2.7 ниже.

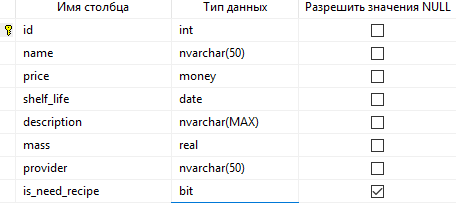


Рисунок 2.7. Таблица лекарств

Таблица DoctorsHospitals *–* содержит информацию о докторах, нанятых в больницах. Хранит информацию о дате начала работы, дате конца работы, название специальности, время начала смены, время конца смены, вторичный ключ для связи с id больницы, вторичный ключ для связи с id доктора. Графическое описание представлено на рисунке 2.8 ниже.

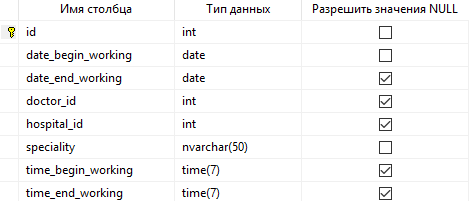


Рисунок 2.8. Таблица нанятых врачей

Таблица DiseasesDrags *–* содержит информацию о лекарствах, назначенных пациентам. Хранит вторичный ключ связанный с id болезни, вторичный ключ для связи с id лекарства. Графическое описание представлено на рисунке 2.9 ниже.

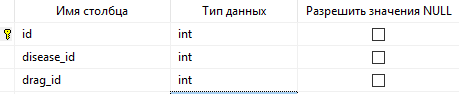


Рисунок 2.9. Таблица назначенных лекарств

T-sql код для создания таблиц представлен в приложении А.

## 2.2. Процедуры

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Их использование в курсовом проекте предназначено для ограничения доступа пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых некоторых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

Все процедуры курсовой работы можно условно поделить на два типа: возвращающие результат выборки из таблицы (рисунок 2.10) или выполняющие изменение данных в таблицах (использующие dml операторы), которые возвращают «0» при успешном завершении операции или код ошибки (рисунок 2.11).

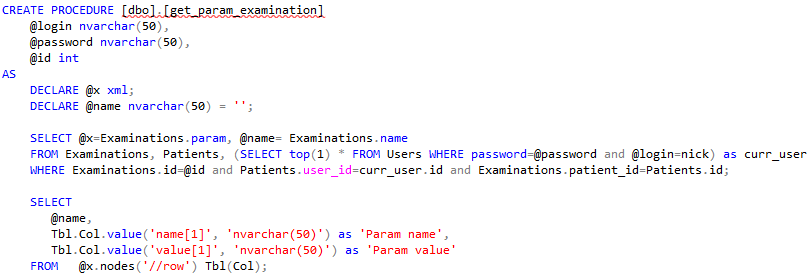


Рисунок 2.10. Процедура получения параметров обследования

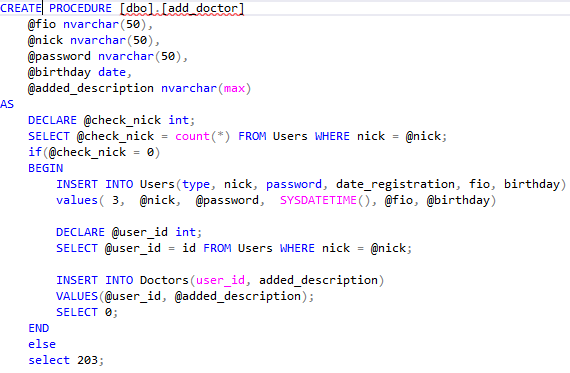


Рисунок 2.11. Процедура добавления доктора

Все исходные коды процедур находятся в приложении Б.

## 2.3. Пользователи

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. При проектировании базы данных понадобилось 4 пользователя (authorizator, user, doctor, reg). У каждого пользователь имеет свои права, предназначенных для выполнения соответствующих ему требованиям.

Пользователь authorizator предназначен для проверки на валидность данных при попытке входа под другим пользователем, и выдачей общедоступной информации. Доступны следующие процедуры: dbo.auth\_user, dbo.get\_drag\_by\_id, dbo.get\_drag\_by\_name, dbo.get\_time\_work\_by\_zip, dbo.get\_time\_work\_by\_fio, dbo.get\_time\_work\_by\_zip\_fio.

Пользователь user предназначен для получения информации о собственной истории болезней пациента и медицинских обследованиях. Доступны следующие процедуры: dbo.get\_inverse\_patient\_diseases, dbo.get\_drags\_by\_disease, dbo.get\_patient\_examinations, dbo.get\_inverse\_patient\_examinations, dbo.get\_time\_work\_by\_zip\_fio, dbo.get\_patient\_diseases.

Пользователь doctor предназначен для получения информации о истории болезней пациентов и медицинских обследованиях, а также имеет возможность назначать лекарства, создавать записи об болезнях и обследованиях. Доступны следующие процедуры: dbo.select\_disease, dbo.add\_disease, dbo.change\_disease, dbo.appoint\_drag, dbo.add\_examination, dbo.add\_param, dbo. get\_doctor\_examination\_param, dbo.get\_doctor\_examinations, dbo.get\_exclude\_doctor\_examinations, dbo.get\_drags\_by\_disease\_doctor, dbo.get\_statistic\_by\_all\_time, dbo.get\_statistic\_by\_index, dbo.get\_statistic\_by\_year, dbo.get\_statistic\_by\_index\_year.

Пользователь reg предназначен для создания пациентов, врачей, привязка врачей к больнице и изменение основной больницы пациента. Доступны следующие процедуры: dbo.add\_patient, dbo.add\_doctor, dbo.take\_doctor, dbo.del\_doctor, dbo.change\_patient.

# 3. Описание процедур импорта и экспорта

Для экспорта и импорта использовалось API разработанной Microsoft – OLE DB. OLE DB ( Object Linking and Embedding, Database) — набор интерфейсов, основанных на COM, которые позволяют приложениям обращаться к данным, хранимым в разных источниках информации или хранилищах данных с помощью унифицированного доступа. Импорт и экспорт производится в формат .xls и .xml согласно заданию курсового проекта.

## 3.1. Процедура импорта

Часть скрипта создания процедуры импорта представлен на рисунке 3.1.

****

Рисунок 3.1 – Процедура импорта

Полный листинг процедуры импорта находится в Приложении B. Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся

## 3.2. Процедура экспорта

Скрипт создания процедуры экспорта представлен на рисунке 3.2.

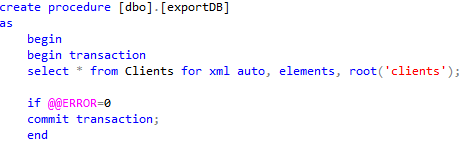
****

Рисунок 3.2 – Процедура экспорта

Для экспорта используется .xml – файл, откуда экспортируются данные.

# 4. Тестирование базы данных

Тестирование происходило с помощью утилиты с открытым исходным кодом – *SQLQueryStress.* Данная утилита позволяет задавать количество повторов запросов, а так же количество потоков (клиентов). Пример работы программы представлен на рисунке 4.1.

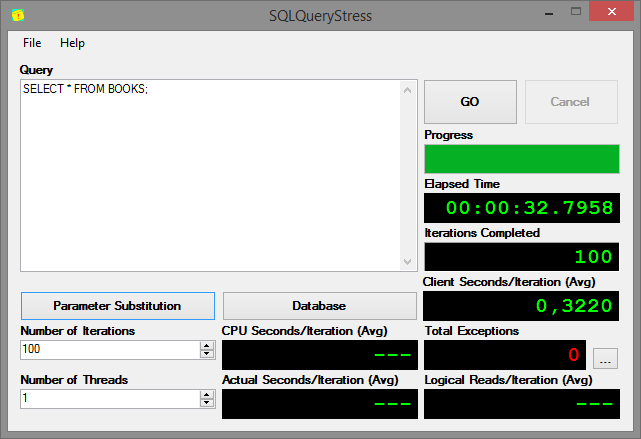


Рисунок 4.1 – Пример работы утилиты *SQLQueryStress*

В ходе тестирования не было выявлено каких-либо замедлений в обработке данных, даже не смотря на большое количество записей (100 тысяч строк). Вследствие чего необходимость в индексе отпала сама собой. Результаты можно улучшить путем улучшения характеристик самого сервера, т.к. данное тестирование проводилось на обычной пользовательской машине, где производительность ниже, чем у сервера.

# 5. Руководство пользователя

Поскольку в базе данных несколько пользователей, то каждый выполняет определенную функцию.

## 5.1. Клиент

Панель клиента представляет собой окно с элементами меню, кнопками пагинации результатов запроса и панелью вывод. Ее вид представлен на рисунке 5.1.1.

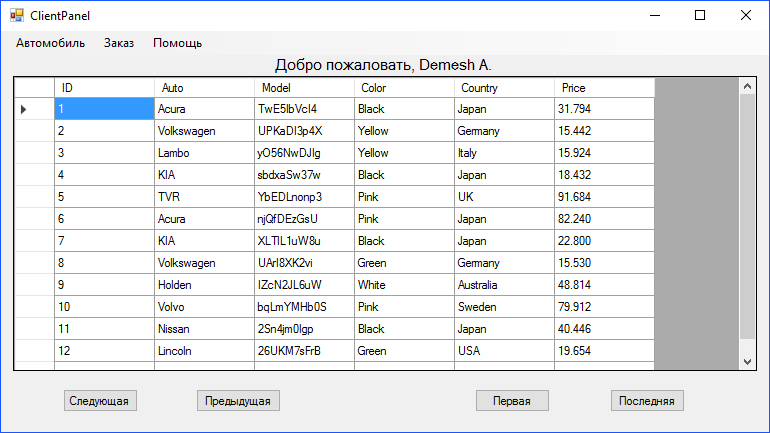


Рисунок 5.1.1 – Панель клиента

Клиент автосалона может просматривать доступные автомобили, используя меню в верхней части приложения. Для удобства была реализована пагинация(из-за большого числа данных).

Так же, клиенту доступен поиск автомобилей по критериям и просмотр собственных покупок.

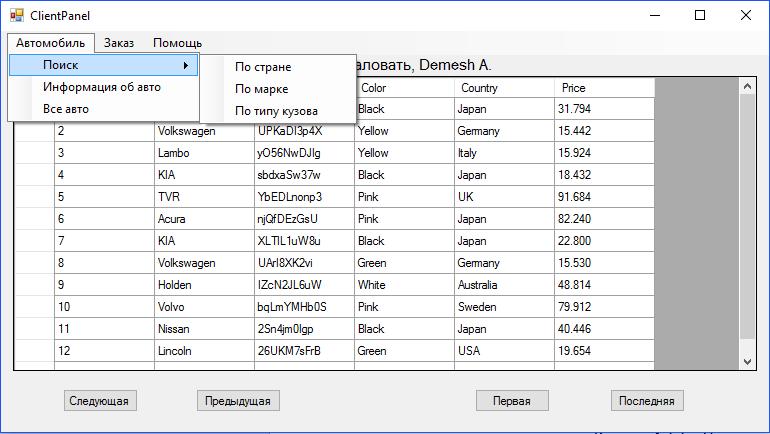


Рис. 5.1.2 – Режимы поиска

## 5.2. Менеджер

Менеджер может просматривать списки клиентов автосалона и их заказы. Так же, ему доступна функция добавления акта покупки в базу данных и добавления клиента. Панель менеджера представлена на рисунке 5.2.1

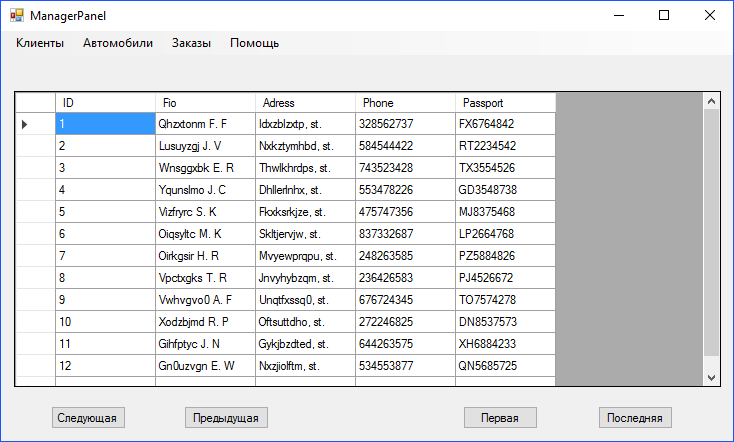


Рисунок 5.2.1 – Панель менеджера

## 5.3. Администратор

Администратор базы данных имеет привилегии на добавление и удаление менеджеров, клиентов, заказов и автомобилей. У него нет графического интерфейса, поэтому все операции выполняются через хранимые процедуры БД.

## 5.4. Гость

Роль гостя используется лишь на этапе авторизации клиентов и менеджеров, для проверки правильности параметров и существования их в базе данных



# Разграничение прав

Для разграничения уровней доступа к БД были использованы роли и пользователи базы данных. Роль – это набор операции с объектами БД, которые доступны определенному пользователю. Пользователь БД – объект базы данных, которые может выполнять операции с элементами базы данных.

# Заключение

В данном курсовом проекте была разработана база данных «Автосалон». Программа полностью автоматизирует труд менеджера автосалона и организует надежное хранение и обработку данных.

В соответствии с полученным результатом работы базы данных можно сделать вывод, что разработанная БД работает верно, а требования технического задания выполнены в полном объеме.

Выполненные цели:

* учет покупок автомобилей;
* учет клиентов компании;
* учет автомобилей;
* оформление процедур покупки.

# Список использованных источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Microsoft OLE DB [Электронный ресурс]/ msdn.microsoft.com – Режим доступа: <https://msdn.microsoft.com/en-us/library/ms722784(v=vs.85).aspx> Дата доступа 10.11.2016.
3. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.

# ПРИЛОЖЕНИЕ А

**SQL – скрипты созданных таблиц**

create table Clients(

Client\_key int primary key not null,

Fio varchar(50) not null,

Passport varchar(15) not null,

Adress varchar(50) not null,

Phone varchar(10) not null,

Login varchar(50) not null unique,

Password varchar(20) not null

);

create table BodyType(

Body\_type int primary key not null,

Title varchar(10) not null

);

create table LocateType(

Locate\_type int primary key not null,

Title varchar(10) not null

);

create table Mark(

Mark\_key int primary key not null,

Country varchar(15) not null,

Title varchar(10) not null

);

create table TechData(

Tech\_key int primary key not null,

BodyType int not null,

Doors int not null,

Sits int not null,

EngineType int not null,

LocateEngine varchar(10) not null,

Value decimal(4,2) not null,

constraint bodytype\_fk foreign key(BodyType) references BodyType(Body\_type),

constraint enginetype\_fk foreign key(EngineType) references EngineType(Engine\_type)

);

create table Automobile(

Auto\_key int primary key not null,

Automobile int not null,

Model varchar(10) not null,

Color varchar(10) not null,

IsAvailb bit not null,

Price varchar(10) not null,

Tech\_data int not null,

constraint auto\_fk foreign key(Automobile) references Mark(Mark\_key),

constraint tech\_fk foreign key(Tech\_data) references TechData(Tech\_key)

);

create table BuyAct(

Buy\_key int not null,

Car int not null,

Client int not null,

BuyDate date not null,

DelivDate date not null,

PayType varchar(10) not null,

constraint client\_fk foreign key(Client) references Clients(Client\_key),

constraint car\_fk foreign key(Car) references Automobile(Auto\_key)

);

# ПРИЛОЖЕНИЕ Б

**Процедуры**

––Вывод всех автомобилей

create procedure [dbo].[all\_cars]

@pagenumb int,

@rowsatpage int

as

begin

select Automobile.Auto\_key as [ID], Mark.Title as [Name], Automobile.Model as [Model],

Automobile.Color as [Color], Mark.Country as [Country], Automobile.IsAvailb as [Availible],

Automobile.Price as [Price] from Automobile inner join Mark on Automobile.Automobile = Mark.Mark\_key

order by Auto\_key

offset((@pagenumb-1)\*@rowsatpage) rows

fetch next @rowsatpage rows only

end

–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

––Авторизация клиентов

create procedure [dbo].[check\_user]

@login varchar(50), @password varchar(32)

as

begin

declare @ID int;

if not exists (select Clients.Client\_key from Clients where Login = @login and Password = @password)

return(0);

else

(select Clients.Client\_key from Clients where Login = @login and Password = @password);

end

–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

––Импорт данных для таблицы Clients

create procedure [dbo].[importClients]

as

begin

CREATE TABLE XMLwithOpenXML

(

Id INT IDENTITY PRIMARY KEY,

XMLData XML,

LoadedDateTime DATETIME

)

INSERT INTO XMLwithOpenXML(XMLData, LoadedDateTime)

SELECT CONVERT(XML, BulkColumn) AS BulkColumn, GETDATE()

FROM OPENROWSET(BULK 'E:\XMLs\clients.xml', SINGLE\_BLOB) AS x;

DECLARE @XML AS XML, @hDoc AS INT, @SQL NVARCHAR (MAX)

SELECT @XML = XMLData FROM XMLwithOpenXML

EXEC sp\_xml\_preparedocument @hDoc OUTPUT, @XML

insert into Clients

SELECT id, FIO, Passport, Adress, Phone, Logn, Pass

FROM OPENXML(@hDoc, 'ArrayOfClient/Client')

WITH

(

id [int] 'id',

FIO [varchar](50) 'Fio',

Passport [varchar](15) 'Passport',

Adress [varchar](50) 'Adress',

Phone [varchar](10) 'Phone',

Logn [varchar](50) 'login',

Pass [varchar](32) 'pass'

)

EXEC sp\_xml\_removedocument @hDoc

drop table XMLwithOpenXML;

end

-Добавление клиента

create procedure [dbo].[add\_client]

@fio varchar(50),

@passport varchar(15),

@adress varchar(50),

@phone varchar(10),

@login varchar(10),

@password varchar(20)

as

begin

if (@fio is null) or (@passport is null) or (@adress is null) or (@phone is null) or (@login is null) or (@password is null)

throw 50007, 'One of parameters is NULL. Please, correct it and try again', 1;

Declare @id int, @pass varchar(32);

set @id = DATEPART(SS, GETDATE())\*5+DATEPART(DAYOFYEAR,GETDATE())-DATEPART(MINUTE,GETDATE()) + DATEPART(MCS, GETDATE())\*2;

--set @pas = convert(varchar(32),HASHBYTES('MD5',@password),2);

begin transaction

insert into Clients values(

@id,

@fio,

@passport,

@adress,

@phone,

@login,

(select convert(varchar(32),HASHBYTES('MD5',@password),2)));

if @@ERROR = 0

begin

commit transaction;

print 'Client was added';

end;

else

rollback transaction;

end

# ПРИЛОЖЕНИЕ В

**Наиболее важные функции**

**Клиент**

private void FillAllCars()

{

this.dataGridView1.Rows.Clear();

this.dataGridView1.Columns.Clear();

using (SqlConnection connect = new SqlConnection(connectString))

{

string query = "all\_cars";

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

SqlParameter logparam = new SqlParameter {

ParameterName = "@pagenumb",

Value = page,

SqlDbType = SqlDbType.Int };

command.Parameters.Add(logparam);

SqlParameter logparam1 = new SqlParameter {

ParameterName = "@rowsatpage",

Value = rowsAtPage,

SqlDbType = SqlDbType.Int };

command.Parameters.Add(logparam1);

var result = command.ExecuteReader();

if (result != null)

{

this.dataGridView1.Columns.Add("id", "ID");

this.dataGridView1.Columns["id"].Width = 100;

this.dataGridView1.Columns.Add("car", "Auto");

this.dataGridView1.Columns["car"].Width = 100;

this.dataGridView1.Columns.Add("model", "Model");

this.dataGridView1.Columns["model"].Width = 100;

this.dataGridView1.Columns.Add("color", "Color");

this.dataGridView1.Columns["color"].Width = 100;

this.dataGridView1.Columns.Add("country", "Country");

this.dataGridView1.Columns["country"].Width = 100;

this.dataGridView1.Columns.Add("price", "Price");

this.dataGridView1.Columns["price"].Width = 100;

while (result.Read()) {

this.dataGridView1.Rows.Add(result[0].ToString(), result[1].ToString(), result[2].ToString(), result[3].ToString(), result[4].ToString(), result[6].ToString());

} } else

MessageBox.Show("Error!");

}

}

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

{

temp te = new temp();

DialogResult res = te.ShowDialog();

if (res != DialogResult.OK)

MessageBox.Show("Введите страну еще раз.");

this.dataGridView1.Rows.Clear();

this.dataGridView1.Columns.Clear();

using (SqlConnection connect = new SqlConnection(connectString))

{

string query = "country\_cars";

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

SqlParameter logparam = new SqlParameter

{

ParameterName = "@country",

Value = Program.q,

SqlDbType = SqlDbType.VarChar

};

command.Parameters.Add(logparam);

var result = command.ExecuteReader();

if (result != null)

{

this.dataGridView1.Columns.Add("id", "ID");

this.dataGridView1.Columns["id"].Width = 100;

this.dataGridView1.Columns.Add("car", "Auto");

this.dataGridView1.Columns["car"].Width = 100;

this.dataGridView1.Columns.Add("model", "Model");

this.dataGridView1.Columns["model"].Width = 100;

this.dataGridView1.Columns.Add("country", "Country");

this.dataGridView1.Columns["country"].Width = 100;

this.dataGridView1.Columns.Add("color", "Color");

this.dataGridView1.Columns["color"].Width = 100;

this.dataGridView1.Columns.Add("price", "Price");

this.dataGridView1.Columns["price"].Width = 100;

while (result.Read())

{

this.dataGridView1.Rows.Add(result[0].ToString(), result[1].ToString(), result[2].ToString(), result[3].ToString(), result[4].ToString(), result[5].ToString());

}

}

else

MessageBox.Show("Error!"); }

}

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

{

FillAllCars();

string query = "count\_cars";

using (SqlConnection connect = new SqlConnection(connectString))

{

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

var result = command.ExecuteScalar();

count = (int)result;

MessageBox.Show(count.ToString());

}

}

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

if (count != 0)

{

page++;

FillAllCars();

}

}

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

if (count != 0)

{

page--;

FillAllCars();

}

}

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

if (count != 0)

{

page = 1;

FillAllCars();

}

}

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

if (count != 0) {

page = count / rowsAtPage;

if (count % rowsAtPage > 0) // if remainder is more than zero

{

page += 1;

}

FillAllCars(); } }

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

{

this.dataGridView1.Rows.Clear();

this.dataGridView1.Columns.Clear();

string query = "all\_types";

using (SqlConnection connect = new SqlConnection(connectString))

{

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

var result = command.ExecuteReader();

if (result != null)

{

this.dataGridView1.Columns.Add("types", "Types");

this.dataGridView1.Columns["types"].Width = 100;

while (result.Read())

{

this.dataGridView1.Rows.Add(result[0].ToString());

}

}

}

}

**Менеджер**

private void FillAllActs()

{

this.dataGridView1.Rows.Clear();

this.dataGridView1.Columns.Clear();

using (SqlConnection connect = new SqlConnection(connectString))

{

string query = "all\_acts";

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

SqlParameter logparam = new SqlParameter

{

ParameterName = "@pagenumb",

Value = page,

SqlDbType = SqlDbType.Int

};

command.Parameters.Add(logparam);

SqlParameter logparam1 = new SqlParameter

{

ParameterName = "@rowsatpage",

Value = rowsAtPage,

SqlDbType = SqlDbType.Int

};

command.Parameters.Add(logparam1);

var result = command.ExecuteReader();

if (result != null)

{

this.dataGridView1.Columns.Add("id", "ID");

this.dataGridView1.Columns["id"].Width = 100;

this.dataGridView1.Columns.Add("car", "CarID");

this.dataGridView1.Columns["car"].Width = 100;

this.dataGridView1.Columns.Add("client", "ClientID");

this.dataGridView1.Columns["client"].Width = 100;

this.dataGridView1.Columns.Add("date", "Date");

this.dataGridView1.Columns["date"].Width = 100;

this.dataGridView1.Columns.Add("pay", "Pay Type");

this.dataGridView1.Columns["pay"].Width = 100;

while (result.Read())

{

this.dataGridView1.Rows.Add(result[0].ToString(), result[1].ToString(), result[2].ToString(), result[3].ToString(), result[4].ToString());

}

}

else

MessageBox.Show("Error!");

}

}

private void всеКлиентыToolStripMenuItem\_Click(object sender, EventArgs e) {

type = "client";

FillAllClients();

string query = "count\_clients";

using (SqlConnection connect = new SqlConnection(connectString)) {

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

var result = command.ExecuteScalar();

count = (int)result;

MessageBox.Show(count.ToString()); }

}

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

{

this.dataGridView1.Rows.Clear();

this.dataGridView1.Columns.Clear();

string query = "all\_mark";

using (SqlConnection connect = new SqlConnection(connectString))

{

connect.Open();

SqlCommand command = new SqlCommand(query, connect);

command.CommandType = CommandType.StoredProcedure;

var result = command.ExecuteReader();

if (result != null)

{

this.dataGridView1.Columns.Add("titl", "Title");

this.dataGridView1.Columns["titl"].Width = 100;

this.dataGridView1.Columns.Add("country", "Country");

this.dataGridView1.Columns["country"].Width = 100;

while (result.Read())

{

this.dataGridView1.Rows.Add(result[0].ToString(),result[1].ToString());

}

}

}

}

# ПРИЛОЖЕНИЕ Г.

Генератор данных

Random rnd = new Random();

string st = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

List<Client> techs = new List<Client>();

for (int i = 1; i <= 100000; i++)

{

string fio = "";

int n = rnd.Next(0, 26);

fio = st.Substring(n, 1);

for (int j = 0; j < 7; j++)

{

n = rnd.Next(27, 53);

string s1 = st.Substring(n, 1);

fio += s1;

}

fio += " ";

n = rnd.Next(0, 23);

fio += st.Substring(n, 1);

fio += ". ";

n = rnd.Next(0, 23);

fio += st.Substring(n, 1);

string passp = "";

for (int j = 0; j < 2; j++)

{

n = rnd.Next(0, 26);

string s1 = st.Substring(n, 1);

passp += s1;

}

for (int j = 0; j < 7; j++)

{

n = rnd.Next(54, 61);

string s1 = st.Substring(n, 1);

passp += s1;

}

string address = "";

n = rnd.Next(0, 23);

address += st.Substring(n, 1);

for (int j = 0; j < 9; j++)

{

n = rnd.Next(27, 53);

string s1 = st.Substring(n, 1);

address += s1;

}

address += ", st.";

string phone = "";

for (int j = 0; j < 9; j++)

{

n = rnd.Next(54, 61);

string s1 = st.Substring(n, 1);

phone += s1;

}

string login = "";

for (int j = 0; j < 10; j++)

{

n = rnd.Next(0, 61);

string s1 = st.Substring(n, 1);

login += s1;

}

string pass = "";

for (int j = 0; j < 10; j++)

{

n = rnd.Next(0, 61);

string s1 = st.Substring(n, 1);

pass += s1;

}

Client tec = new Client(i, fio, passp, address, phone, login, CaclMD5.CalculateMD5Hash(pass));

techs.Add(tec);

}