МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Разработка базы данных для приложения «Auto flea market»

Выполнил студент Левицкий В. С.

(Ф.И.О.)

Руководитель проекта ст. п. Блинова Е. А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

# Реферат

Пояснительная записка курсового проекта содержит 39 страниц пояснительной записки, 20 иллюстраций, 8 источников литературы, 5 приложений.

При разработке использовались: Oracle Database 12c, Oracle SQL Developer 19.2.1,

Цель курсового проекта – разработка базы данных, которая обеспечивает быстрое взаимодействие между базой данных и сервером, что позволит пользователю выставлять на продажу и находить б/у автомобили.

В первой главе проводится аналитический обзор схожих приложений по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы и описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования проекта.

В четвёртой главе описано тестирование, проверка работоспособности и анализ данных системы.

В заключении приведены результаты проделанной работы.

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Реферат

Лит.

Листов

1

БГТУ

# СОДЕРЖАНИЕ

Оглавление

[Реферат 2](#_Toc58846481)

[СОДЕРЖАНИЕ 3](#_Toc58846482)

[Введение 4](#_Toc58846483)

[1 Аналитический обзор схожих приложений 5](#_Toc58846484)

[2 Проектирование и разработка базы данных 8](#_Toc58846488)

[2.1 Спецификация функциональных требований 8](#_Toc58846489)

[2.2 Разработка модели базы данных 10](#_Toc58846490)

[2.3 Пользователи 12](#_Toc58846502)

[2.4 Хранимые процедуры 12](#_Toc58846503)

[2.4.1 Выборка данных для таблиц 12](#_Toc58846504)

[2.4.2 Удаление данных из таблиц 13](#_Toc58846505)

[2.4.3 Добавление данных в таблицы 13](#_Toc58846506)

[2.4.5 Экспорт и импорт таблиц в формат xml 13](#_Toc58846507)

[2.5 Индексы 13](#_Toc58846508)

[2.6 Пакеты 14](#_Toc58846509)

[2.7 Реализация последовательностей 16](#_Toc58846510)

[2.8 Триггеры 16](#_Toc58846511)

[2.9 Директории 17](#_Toc58846512)

[2.10 Реализация технологий 17](#_Toc58846513)

[3 Тестирование, проверка работоспособности и анализ данных 20](#_Toc58846514)

[Заключение 22](#_Toc58846515)

[Список литературы 23](#_Toc58846516)

[Приложение А 24](#_Toc58846517)

[Приложение Б 25](#_Toc58846518)

[Приложение В 29](#_Toc58846519)

[Приложение Г 31](#_Toc58846520)

[Приложение Д 37](#_Toc58846521)

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

В**ведение**

В данной записке приведено описание реляционной базы данных разработанных в соответствии с заданием на курсовое проектирование по теме «Auto flea market» по дисциплине «Базы данных».

Задачей данной курсовой работы является разработка базы данных предназначенной для приложения «Auto flea market». Приложение будет обладать следующими функционалом:

* Размещение автомобиля.
* Возможность добавления в избранное.
* Возможность оставлять комментарии.
* Возможность редактирования объявлений.
* Фильтр объявлений.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Для разработки и управления базой данных курсового проекта использовались система управления реляционными базами данных Oracle Database 12c, в связи с ее производительностью и надежностью, и графический интерфейс Oracle SQL Developer из-за интуитивной понятности и большого спектра возможности.

# 1 Аналитический обзор схожих приложений

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

Для составления технических требований к проекту был проведен анализ похожих программных средств. Из открытого источника (интернет) были найдены web-ресурсы предоставляющее схожий функционал. Один из примеров – сайт av.by [1].

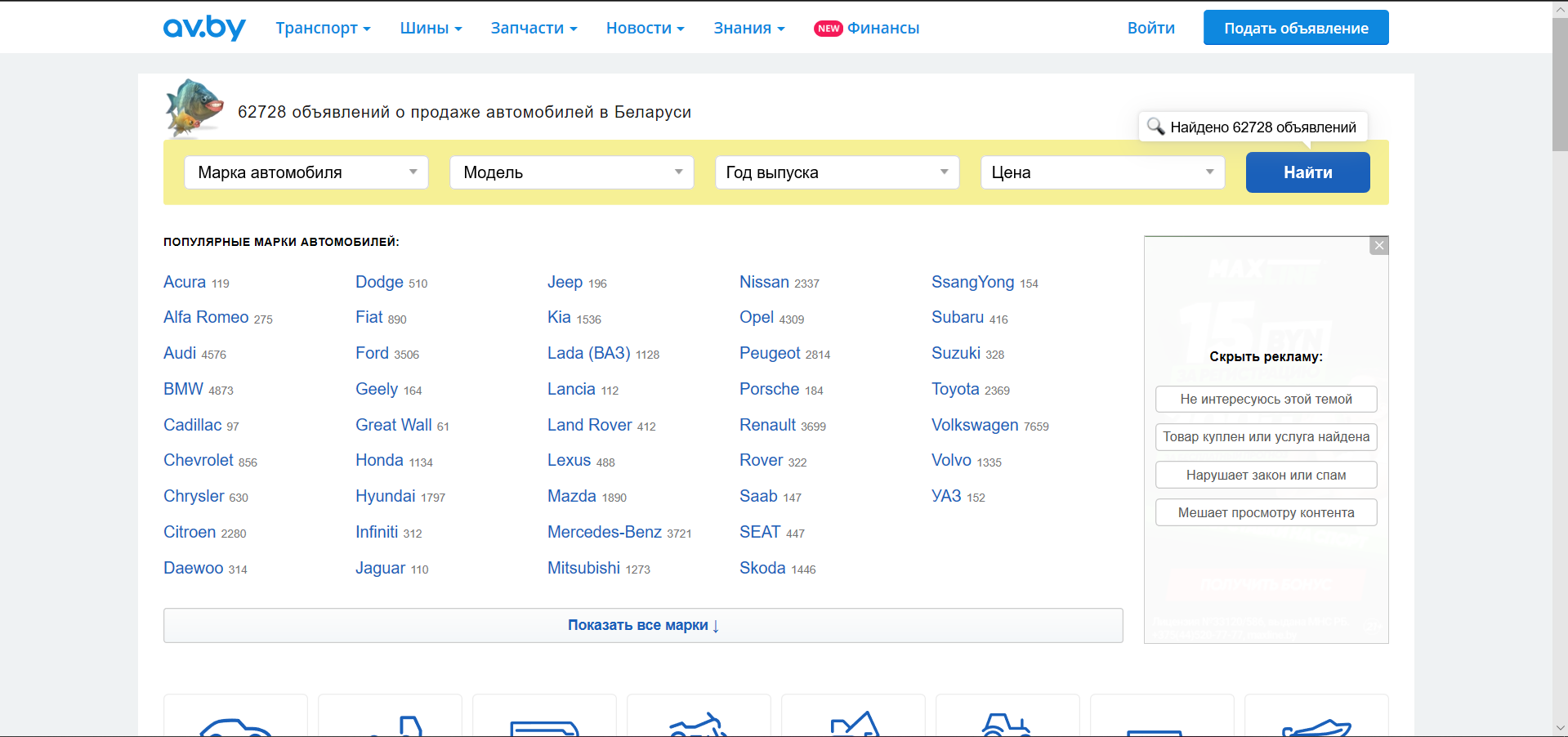


Рисунок 1.1 – Web-сайта av.by

Пример поиска авто по марке на сайте av.by.

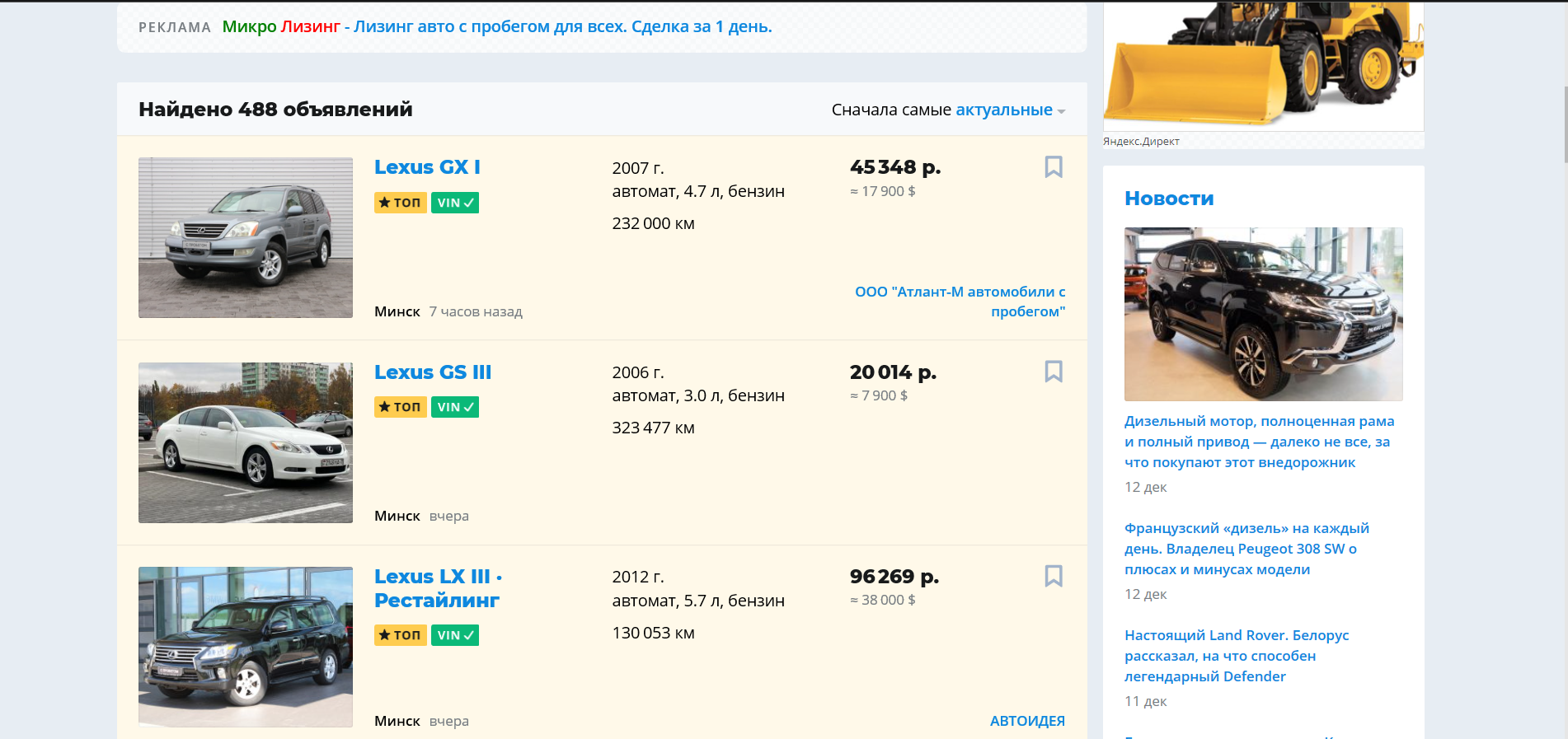


Рисунок 1.2 ­­– интерфейс выбора авто по марке

Функционально сайт включает в себя:

* Поиск на сайте по марке;
* Поиск на сайте по модели;
* Поиск на сайте запчастей для конкретной модели;
* Поиск на сайте техники для конкретных услуг;
* Оставить объявление о продаже;
* Просмотр актуальных новостей;
* Просмотр рецензий на автомобили;
* Просмотр доступных сервисов.

Следующим рассмотренным веб-сайт является ab.onliner.by[2].

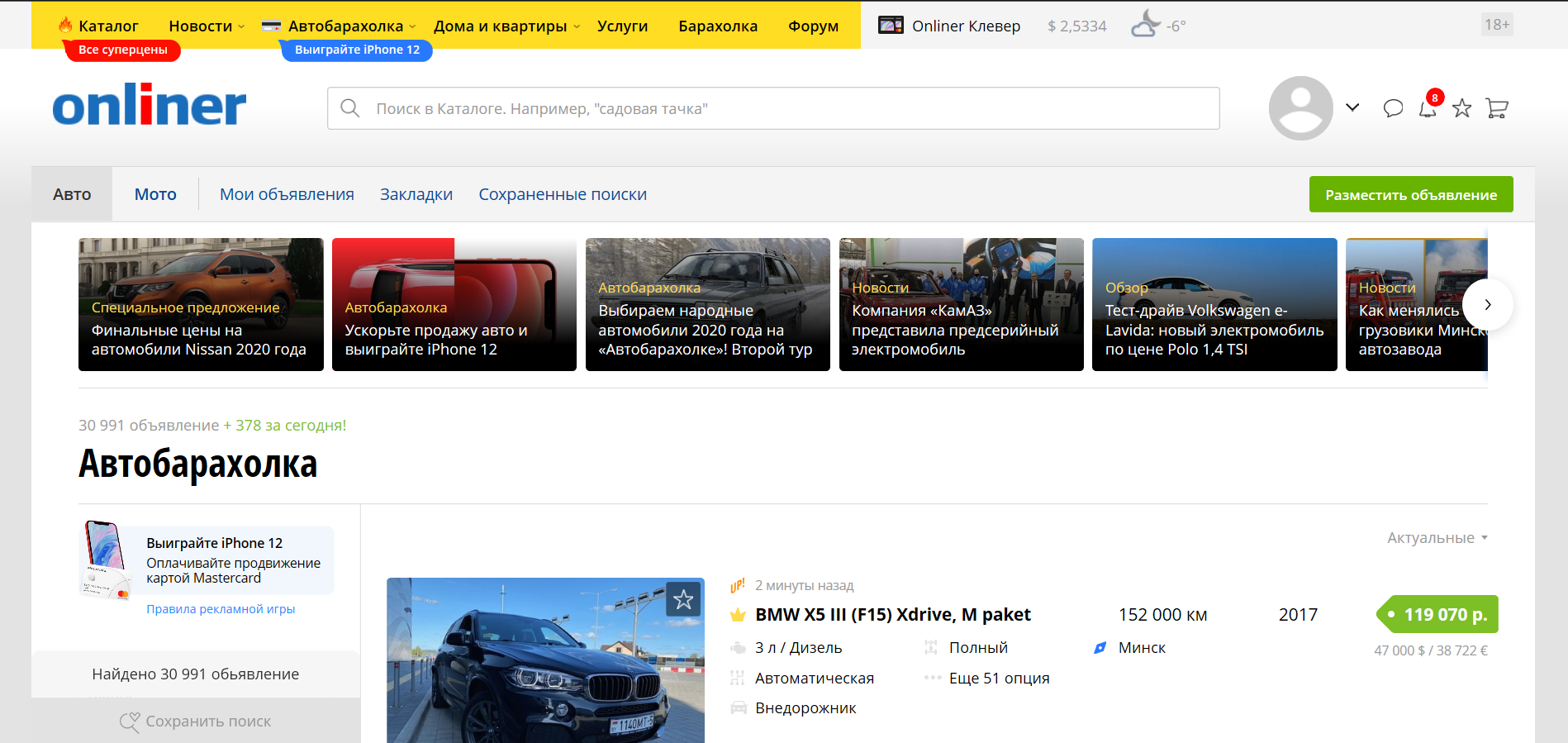


Рисунок 1.3 – Веб-сайт ab.onliner.by

Процесса выбора авто по марке на сайте ab.onliner.by.

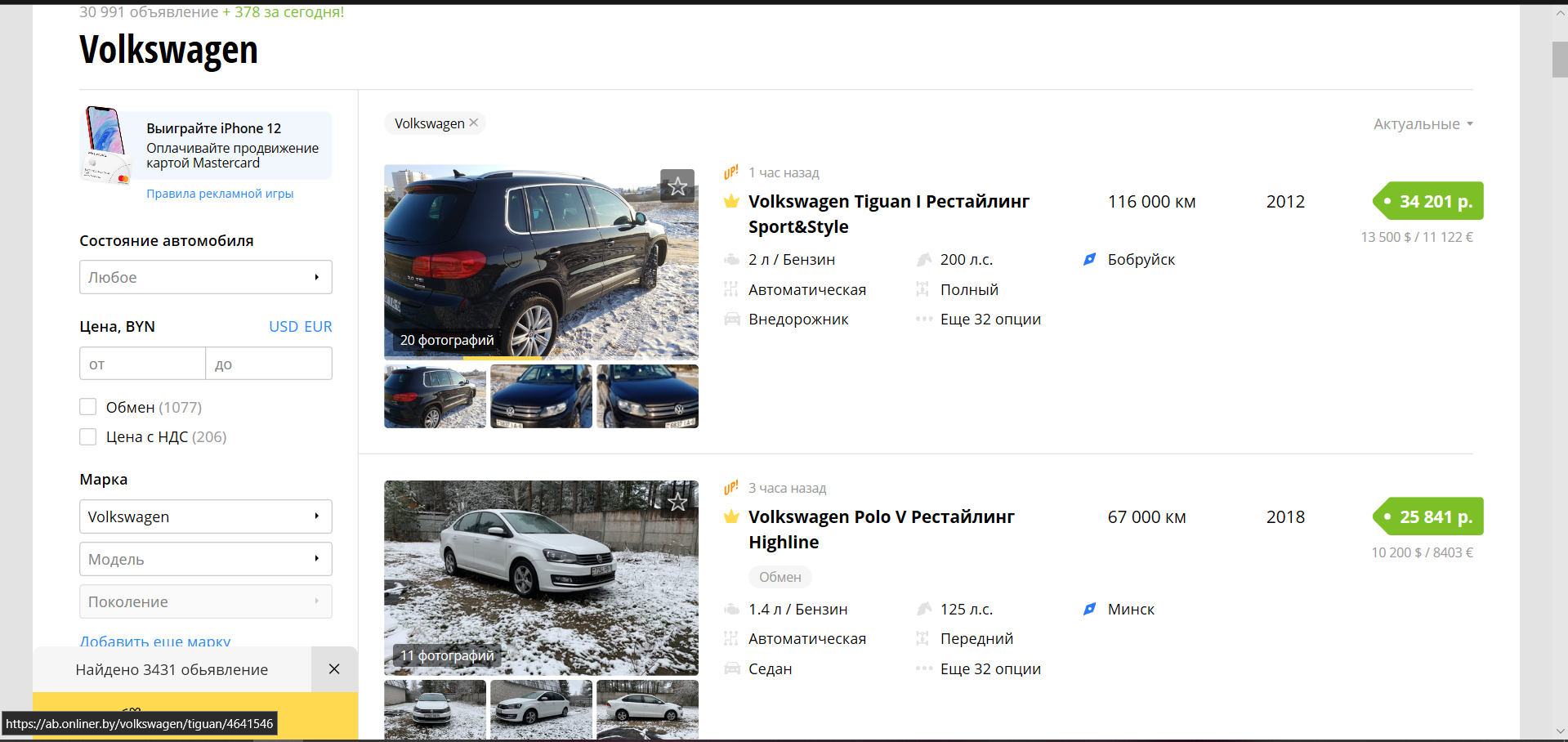


Рисунок 1.4 – Интерфейс выбора авто по определенным параметрам

Приложение Holidays дает следующие возможности:

* Поиск на сайте по марке;
* Поиск на сайте по модели;
* Поиск на сайте запчастей для конкретной модели;
* Поиск на сайте техники для конкретных услуг;
* Оставить объявление о продаже.

Все вышеперечисленные сайты обладают схожим функционалом для облегчения выполнения всех необходимых операций, связанных с поиском, покупкой и продажей авто.

Проанализировав данные сайты, я попробовал реализовать облегченный вариант регистрации и размещения объявления.

# 2 Проектирование и разработка базы данных

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Мингазов А. И.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

2 Проектирование и разработка базы данных

Лит.

Листов

1

БГТУ

При разработке курсового проекта понадобились следующие объекты:

1. Таблицы.
2. Хранимые процедуры.
3. Пакеты.
4. Последовательности.
5. Индексы.
6. Триггеры.
7. Хранимые директории.

## 2.1 Спецификация функциональных требований

Рассмотрим возможности, которые предоставляет разработанное программное средство. Для наглядности демонстрации использовались UML -диаграммы.

UML – уникальный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. [3]

На рисунке 2.1 показана UML-диаграмма, отображающая возможности приложения с точки зрения неавторизованного пользователя.

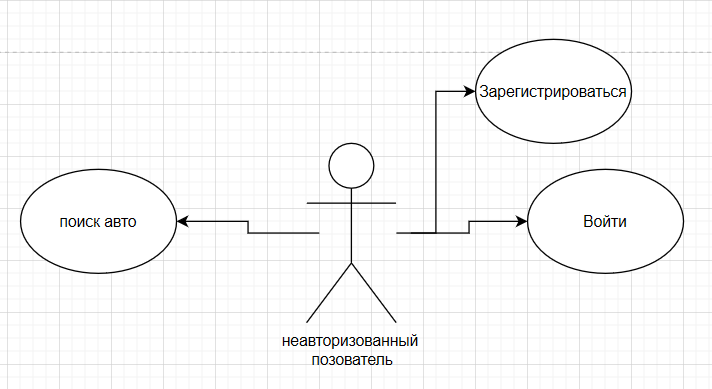


Рисунок 2.2.1 – UML диаграмма вариантов использования для неавторизованного пользователя

Неавторизованный пользователь имеет возможности: авторизоваться, зарегистрироваться, искать и просматривать объявления о продаже авто.

Если пользователь авторизуется, то будет иметь возможности, представленные на рисунке 2.2.

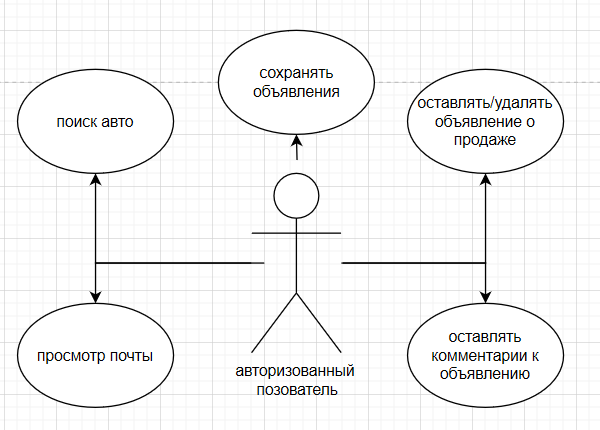


Рисунок 2.2.2 – UML диаграмма вариантов использования для авторизованного пользователя

Авторизованный пользователь может искать авто, оставлять/удалять/сохранять объявления, оставлять комментарии, просматривать сообщения от администрации и системы, выход.

Также в приложение можно войти как администратор. Тогда открываются следующие возможности: экспорт и импорт всех таблиц с информацией в XML, удаление объявлений любого пользователя, добавление новых брендов и моделей, поиск объявления, оставлять сообщения для пользователя, поиск по пользователям и выход из аккаунта. Диаграмма вариантов использования для администратора отображена на рисунке 2.3.

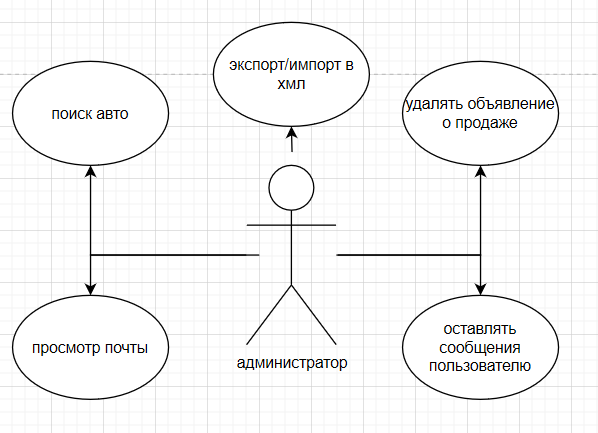


Рисунок 2.2.3 – UML диаграмма вариантов использования для администратора

## 2.2 Разработка модели базы данных

Для реализации базы данных для туристического агентства было разработано 10 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлен скрипт создания БД.

Логически можно вывести 10 основных таблиц: Brand, Model, Cars, Saved, Contacts, Users, Comments, Post, Pictures и Spares.

Таблица Users представляет перечень пользователей, состоит из столбцов:

−ID\_USER – идентификатор пользователя, тип int, первичный ключ;

− NAME – имя пользователя, тип varchar;

− LOGIN – уникальное имя пользователя для входа в систему, тип varchar;

− PASSWORD – пароль пользователя, тип varchar;

− PRIVILEGE – привилегии клиента, тип varchar.

Таблица Contacts представляет перечень контактных данных пользователя, состоит из столбцов:

−ID\_CONTACTS – идентификатор контакта, тип int, первичный ключ;

− ID\_USER– идентификатор пользователя, которому принадлежит данный контакт, тип int;

− CONTACT – контакт, тип varchar.

Таблица Brand представляет перечень брендов, состоит из столбцов:

−ID\_BRAND – идентификатор бренда, тип int, первичный ключ;

− BRAND – название бренда, тип varchar.

Таблица Model представляет перечень моделей, состоит из столбцов:

−ID\_MODEL– идентификатор модели, тип int, первичный ключ;

− ID\_BRAND – идентификатор бренда, к которому относится данная модель, тип int, внешний ключ;

− MODEL – название модели, тип varchar.

Таблица Car представляет информацию об авто, состоит из столбцов:

−ID\_CAR – идентификатор авто, тип int, первичный ключ;

− ID\_USER – идентификатор пользователя, тип int;

− ID\_MODEL – идентификатор модели авто, тип int;

− ID\_PIC – фотография, тип int;

− DESCRIPTION\_CAR – описание авто, тип varchar;

− PRICE – цена, тип int;

Таблица Spares представляет информацию о деталях, состоит из столбцов:

−ID\_SPARE – идентификатор детали, тип int, первичный ключ;

− ID\_USER – идентификатор пользователя, тип int;

− ID\_MODEL – идентификатор модели детали для авто, тип int;

− ID\_PIC – фотография, тип int;

− Spare – описание детали, тип varchar;

− PRICE – цена, тип int;

Таблица Pictures представляет фото и информацию об нем, состоит из столбцов:

−ID\_PICTURE – идентификатор картинки, тип int, первичный ключ;

− ID\_CAR – идентификатор авто, тип int;

− ID\_SPARE – идентификатор детали, тип int;

− PICTURE – фотография, тип blob;

Таблица Saved представляет информацию о сохраненных объявлениях, состоит из столбцов:

−ID\_SAVED – идентификатор сохраненного объявления, тип int, первичный ключ;

− ID\_CAR – идентификатор авто, тип int;

− ID\_SPARE – идентификатор детали, тип int;

− ID\_USER – идентификатор пользователя, тип int;

− DATE – дата сохранения, тип date;

Таблица POST представляет информацию о сообщенияхот администратора пользователю, состоит из столбцов:

−ID\_POST– идентификатор сообщения, тип int, первичный ключ;

− ID\_AUTO – идентификатор авто, тип int;

− ID\_SPARE – идентификатор детали, тип int;

− ID\_USER – идентификатор пользователя, тип int;

− POST – сообщение, тип varchar.

# 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 были использованы 3 пользователя. Первый пользователь – с##client – обычный пользователь, имеет доступ к чтению и записи в таблицах, к которым имеет доступ.

Второй пользователь – managerr – обладает более обширным перечнем прав. Имеет права для чтения из всех таблиц и изменения и записи во всех таблицах, к которым есть доступ у пользователя.

Третий пользователь – c##super – выполняет роль системы, которая держит на себе все объекты бд. Имеет все права.

## 2.4 Хранимые процедуры

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

При разработке курсового проекта было создано 36 процедур, хранящихся в 4 пакетах, для следующих целей:

1. Выборка данных для таблиц;

2. Удаление данных из таблиц;

3. Добавление данных в таблицы;

4. Изменение данных в таблицах;

5. Экспорт и импорт таблиц в формат xml;

Весь перечень созданных процедур будет представлен в Приложении В.

### 2.4.1 Выборка данных для таблиц

Для вывода данных из таблиц были написаны следующие процедуры: GET\_CAR, GET\_SPARES, GET\_USER, USER\_BY\_ID,SPARE\_BY\_ID, SPARE\_BY\_USER, CAR\_BY\_MODEL и другие. Поскольку вывод таблицы размером в 100000 строк занимает довольно много времени, было принято решение создать индексы, что позволило уменьшить время вывода таблицы до минимума.

### 2.4.2 Удаление данных из таблиц

Для управления базой данных были созданы процедуры удаления DEL\_CAR, DEL\_SPARE, DEL\_USERS и DEL\_PICTURE. Они были разработаны для удаления строк из таблиц: CAR, USERS, SPARE и PICTURE.

### 2.4.3 Добавление данных в таблицы

Для управления базой данных были созданы процедуры добавления новых строк: ADD\_CAR, ADD\_SPARE, ADD\_USER, ADD\_POST, ADD\_COMMENT, ADD\_PICTURE, ADD\_SAVED, ADD\_CONTACT, ADD\_BRAND, ADD\_MODEL. Они были разработаны для заполнения соответствующих таблиц.

**2.4.4 Изменение данных в таблицах**

Для управления базой данных были созданы процедуры обновления: UPDATE\_CAR, UPDATE\_SPARE. Они были разработаны для обновления данных в соответствующих таблицах.

### 2.4.5 Экспорт и импорт таблиц в формат xml

Для экспорта таблицы BRAND в формате xml[5] была разработана процедура EXPORT\_BRAND\_TO\_XML. Для импорта таблицы BRAND в формате xml разработана процедура IMPORT\_BRAND\_FROM\_XML. Данные процедуры приведены в приложении Г и Д соответственно.

## 2.5 Индексы

Индекс [6] – объект базы данных, создаваемый с целью повышения производительности поиска данных [6]. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. В связи с необходимостью выборки таблиц, заполненных минимум на 100 000 строк, было принято решение разработать 9 индекса для всех таблиц.

Список индексов, созданных в рамках курсового проекта приведены ниже:

* IX\_BRAND ON BRAND (BRAND, ID\_BRAND);
* IX\_MODEL ON MODEL\_ (MODEL\_, ID\_MODEL);
* IX\_CAR ON CAR (ID\_CAR, ID\_USER, ID\_MODEL, ID\_PIC);
* IX\_COMMENT ON COMMENT\_ (ID\_COMMENT, ID\_USER, ID\_CAR, ID\_SPARE);
* IX\_CONTACT ON CONTACTS (ID\_CONTACTS, ID\_USER);
* IX\_POST ON POST (ID\_POST, ID\_USER, ID\_AUTO, ID\_SPARES);
* IX\_SAVED ON SAVED (ID\_SAVED, ID\_CAR, ID\_SPARE, ID\_USER);
* IX\_SPARES ON SPARES (ID\_SPARES, ID\_USER, ID\_MODEL, ID\_PIC);

IX\_PICTURE ON PICTURES (ID\_PICTURE, ID\_CAR, ID\_SPARE);

Во время разработки использовалось 2 bitmap индекса для таблиц brand и model, так как данные таблицы не должны часто изменяться.

Для наглядной демонстрации повышения производительности поиска данных в таблице, ниже приведены рисунки с необходимой статистикой.





Рисунок 2.5.1 – Время и число операций чтения необходимые для поиска данных в таблице brand без индекса и с индексом

## 2.6 Пакеты

Пакет[5] Oracle PL/SQL — это объект схемы, который группирует логически связанные типы, элементы и подпрограммы. Пакеты обычно состоят из двух частей: спецификации и тела, хотя иногда тело не нужно.

Во время проектирования бд мне понадобилось создать 4 пакета для группировки процедур: USER\_PACK, ADMIN\_PACK, OBS\_PACK и XML\_PACKAGE.

USER\_PACK – хранит процедуры, которыми может пользоваться только пользователь (рисунок 2.6.1)

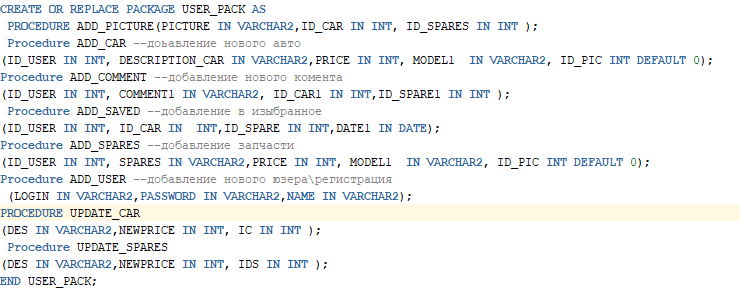


Рисунок 2.6.1 – спецификация пакета user\_pack

ADMIN\_PACK – хранит процедуры, которыми может пользоваться только администратор (рисунок 2.6.2)

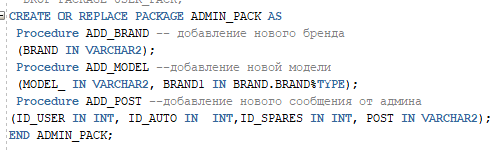


Рисунок 2.6.2 – спецификация пакета admin\_pack

OBS\_PACK – хранит процедуры, которыми могут пользоваться пользователь и администратор (рисунок 2.6.3)

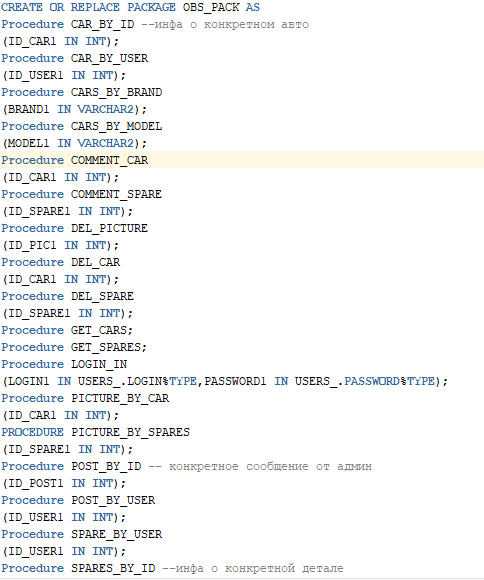




Рисунок 2.6.3 – спецификация пакета obs\_pack

XML\_PACKAGE – хранит процедуры, для работы с хмл (рисунок 2.6.4)

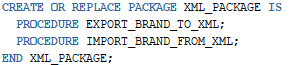


Рисунок 2.6.4 – спецификация пакета XML\_PACKAGE

## 2.7 Реализация последовательностей

Последовательность CREATE SEQUENCE [7] – это объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания.

В процессе разработки проекта были использованы 10 последовательностей: car\_s, user\_s, spare\_s, comment\_s, post\_s, bran\_s, model\_s, saved\_s, picture\_s, contact\_s

Примем создания последовательности приведен на рисунке 2.7.1

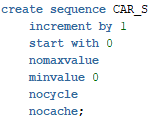


Рисунок 2.7.1 – создание последовательности

## 2.8 Триггеры

Триггеры PL/SQL - это именованные программные блоки, выполняемые в ответ на происходящие в базе данных Oracle события.

В процессе разработки проекта был создан триггер TR\_ADD\_LOGIN, который работает перед отработкой блока insert в таблицу users\_ и проверяет на наличие дублирование логинов (рисунок 2.8.1).

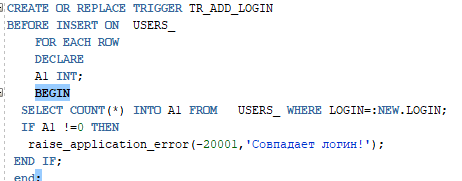


Рисунок 2.8.1 – создание триггера

## 2.9 Директории

Объект базы данных DIRECTORY - это ссылка на каталог сервера, на котором работает инстанс Oracle.

в ходе разработки бд были созданы 2 директории, для хранения xml и изображений: UTLDATA и PIC\_DIR.

Пример создания директории рисунок 2.9.1



Рисунок 2.9.1 – создание директории

## 2.10 Реализация технологий

В процессе разработки проекта была использована технология: мультимедийные типы данных в БД.

Oracle и PL/SQL поддерживают несколько разновидностей типов данных, предназначенных специально для работы с большими объектами (LOB[8], Large OBjects). Такие объекты позволяют хранить огромные (от 8 до 128 терабайт) объемы двоичных (например, графических) или текстовых данных.

В PL/SQL можно объявлять большие объекты четырех типов:

* BFILE— двоичный файл. Переменная этого типа содержит локатор файла, указывающий на файл операционной системы вне базы данных. Oracle интерпретирует содержимое файла как двоичные данные.
* BLOB— большой двоичный объект. Переменная этого типа содержит локатоp LOB, указывающий на большой двоичный объект, хранящийся в базе данных.
* CLOB — большой символьный объект. Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных в наборе символов базы данных.
* NLOB — большой символьный объект с поддержкой символов национальных языков (NLOB). Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных с национальным набором символов.

Большие объекты можно разделить на две категории: внутренние и внешние. Внутренние большие объекты (типы BLOB,CLOB и NCLOB) хранятся в базе данных и могут участвовать в транзакциях на сервере базы данных. Внешние большие объекты (тип BFILE) представляют двоичные данные, хранящиеся в файлах операционной системы вне таблиц базы данных. Они не могут участвовать в транзакциях, то есть вносимые в них изменения нельзя сохранить или отменить в зависимости от результата транзакции. Целостность данных обеспечивается только на уровне файловой системы. Кроме того, повторное чтение из BFILE может приводить к разным результатам — в отличие от внутренних больших объектов, соответствующих модели логической целостности чтения.

Код процедуры для добавления файла в бд:

create or replace Procedure ADD\_PICTURE --добавление картинки

(PICTURE IN VARCHAR2, ID\_CAR IN INT, ID\_SPARES IN INT )

IS

ID\_PICTURE INT;

L\_BLOB BLOB;

L\_BFILE BFILE;

BEGIN

SELECT PICTURE\_S.NEXTVAL INTO ID\_PICTURE FROM dual;

INSERT INTO PICTURES

( ID\_PICTURE,ID\_CAR, ID\_SPARE,PICTURE)

VALUES

(ID\_PICTURE,ID\_CAR,ID\_SPARES,EMPTY\_BLOB())

RETURN PICTURE INTO L\_BLOB;

L\_BFILE := BFILENAME('PIC\_DIR',PICTURE);

DBMS\_LOB.FILEOPEN(L\_BFILE,DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADFROMFILE(L\_BLOB,L\_BFILE, DBMS\_LOB.GETLENGTH(L\_BFILE));

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

Где: PICTURE IN VARCHAR2 – имя загружаемого файла,

ID\_CAR IN INT – ID авто, к которому относится данная картинка,

ID\_SPARES IN INT – ID детали, к которому относится данная картинка,

ID\_PICTURE INT – ID данной картинки,

L\_BLOB BLOB – переменная, для хранения картинки во время исполнения кода PL/SQL,

L\_BFILE BFILE – переменная, для открытия файл, находящегося в директории.

Вставка в таблицу PICTURE:  
INSERT INTO PICTURES

(ID\_PICTURE,ID\_CAR, ID\_SPARE,PICTURE)

VALUES

(ID\_PICTURE,ID\_CAR,ID\_SPARES,EMPTY\_BLOB())

RETURN PICTURE INTO L\_BLOB;

EMPTY\_BLOB() – функция, которая заполняет колонку picture пустым значение типа BLOB. Дальше мы возвращаем это значение из колонки picture в переменную L\_BLOB.

L\_BFILE := BFILENAME('PIC\_DIR',PICTURE) – получаем путь файла с помощью созданной директории, и имени файла.

DBMS\_LOB.FILEOPEN(L\_BFILE,DBMS\_LOB.FILE\_READONLY) – открываем поток для чтения данных из указанного пути.

DBMS\_LOB.LOADFROMFILE(L\_BLOB,L\_BFILE, DBMS\_LOB.GETLENGTH(L\_BFILE)) – записываем данные из фалайла в переменную L\_BLOB.

DBMS\_LOB.FILECLOSE(L\_BFILE) – закрываем поток.

# 3 Тестирование, проверка работоспособности и анализ данных

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

4 Тестирование, проверка работоспособности и анализ данных

Лит.

Листов

1

БГТУ

Разработанное программное средство полностью соответствует созданному для него техническому заданию, также оно использует дополнительно разработанный функционал.

Для проверки корректности работы программного средства, продемонстрируем следующий скрипт

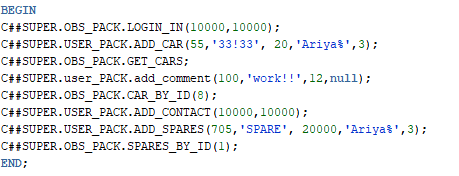


Рисунок 4.1 – Скрипт, демонстрирующий возможности пользователя

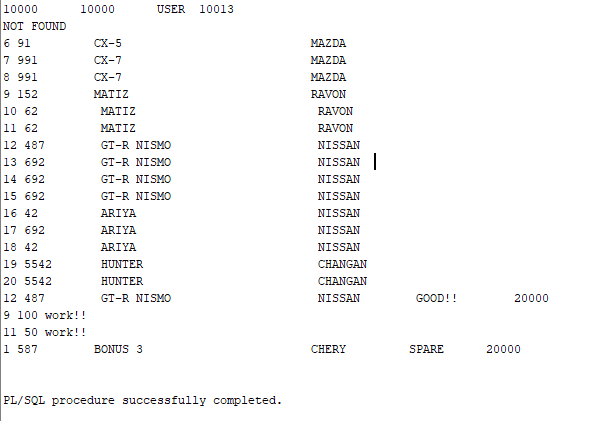


Рисунок 4.2 – вывод скрипта пользовотеля

Скрипт, демонстрирующий работоспособности администратора приведен ниже

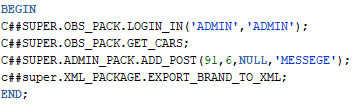


Рисунок 4.3 – Скрипт, демонстрирующий возможности администратора

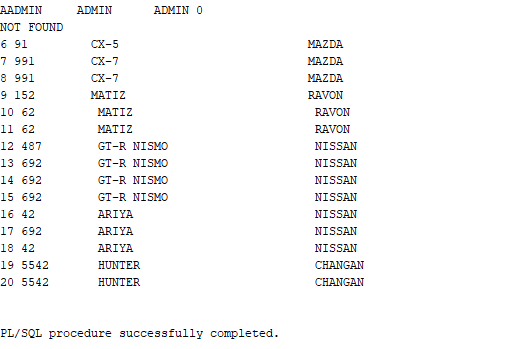


Рисунок 4.4 – вывод скрипта администратора

# Заключение

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Заключение

Лит.

Листов

1

БГТУ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Auto flea market». Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением/сайтом, которые помогут облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке выполнены следующие пункты:

* размещение автомобиля.
* возможность добавления в избранное.
* возможность оставлять комментарии.
* возможность редактирования объявлений.
* фильтр объявлений.
* технология работы с мультимедийными типами данных;
* импорт и экспорт XML.

Приложение прошло тестирование при использовании в БД большого количество данных.

Данный проект можно сравнить с аналогами, так как большинство из них несет схожий функционал. И исходя из этого я постарался реализовать более простое добавление объявлений, чтобы более выгодно выделить его на фоне аналогов. По сути, данный проект является показательным примером базы данных для различных площадок по продаже б/у авто, позволяя охватить необходимый минимум требований для их администрирования.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

# Список литературы

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Список литературы

Лит.

Листов

1

БГТУ

1. Сайт av.by – https://av.by/.
2. Сайт ab.onliner.by – https://ab.onliner.by/.
3. Интернет ресурс для проектирования и создания UML-диаграмм – https://app.diagrams.net
4. Сайт habr.com – https://habr.com/ru/post/256655/.
5. Сайт oracle doc – https://docs.oracle.com/cd/B19306\_01/appdev.102/b14261/packages.htm.
6. Сайт oracle-dba.ru – https://oracle-dba.ru/docs/architecture/indexes/.
7. Сайт oracleplsql.ru– https://oracleplsql.ru/sequences.html
8. Сайт oracle-patches.com - https://oracle-patches.com/db/sql/3472-большие-объекты-данных-в-pl-sql-blob,-clob,-nclob-и-bfile

# Приложение А

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

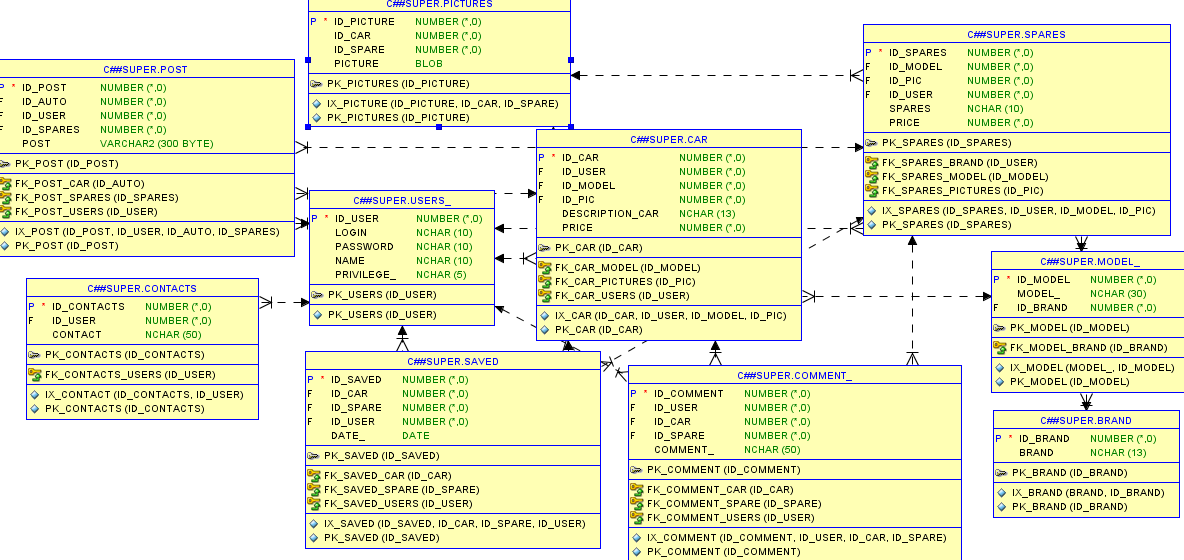
Приложение А

Лит.

Листов

1

БГТУ

****

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В.С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Б

Лит.

Листов

1

БГТУ

# Приложение Б

CREATE TABLE CAR (

id\_car int NOT NULL,

id\_user int NULL,

id\_model int NULL,

id\_pic int NULL,

description\_car nchar(13) NULL,

price int NULL,

CONSTRAINT PK\_car PRIMARY KEY (

id\_car

)

);

--DROP TABLE CAR;

ALTER TABLE CAR ADD CONSTRAINT FK\_car\_Model FOREIGN KEY(id\_model)

REFERENCES Model\_ (id\_model);

ALTER TABLE car ADD CONSTRAINT FK\_car\_Pictures FOREIGN KEY(id\_pic)

REFERENCES Pictures (id\_picture);

ALTER TABLE car ADD CONSTRAINT FK\_car\_Users FOREIGN KEY(id\_user)

REFERENCES Users\_ (id\_user);

CREATE TABLE Brand(

id\_brand int NOT NULL,

Brand nchar(13) NULL,

CONSTRAINT PK\_Brand PRIMARY KEY

(

id\_brand

));

CREATE TABLE Comment\_(

id\_comment int NOT NULL,

id\_user int NULL,

id\_car int null,

id\_spare int null,

Comment\_ nchar(50) NULL,

CONSTRAINT PK\_Comment PRIMARY KEY

(

id\_comment

));

ALTER TABLE Comment\_ ADD CONSTRAINT FK\_Comment\_CAR FOREIGN KEY(ID\_CAR)

REFERENCES CAR (ID\_CAR);

ALTER TABLE Comment\_ ADD CONSTRAINT FK\_Comment\_SPARE FOREIGN KEY(ID\_SPARE)

REFERENCES SPARES (ID\_SPARES);

ALTER TABLE Comment\_ ADD CONSTRAINT FK\_Comment\_Users FOREIGN KEY(id\_USER)

REFERENCES Users\_ (id\_user);

CREATE TABLE Contacts(

id\_contacts int NOT NULL,

id\_user int NULL,

Contact nchar(50) NULL,

CONSTRAINT PK\_Contacts PRIMARY KEY

(

id\_contacts

));

ALTER TABLE Contacts ADD CONSTRAINT FK\_Contacts\_Users FOREIGN KEY(id\_user)

REFERENCES Users\_ (id\_user);

CREATE TABLE Model\_(

id\_model int NOT NULL,

Model\_ nchar(30) NULL,

id\_brand int NULL,

CONSTRAINT PK\_Model PRIMARY KEY

(

id\_model

));

ALTER TABLE Model\_ ADD CONSTRAINT FK\_Model\_Brand FOREIGN KEY(id\_brand)

REFERENCES Brand (id\_brand);

CREATE TABLE Post(

id\_post int NOT NULL,

id\_AUTO int NULL,

id\_user int NULL,

id\_spares int NULL,

POST VARCHAR(300),

CONSTRAINT PK\_Post PRIMARY KEY

(

id\_post

));

--drop table Post;

ALTER TABLE Post ADD CONSTRAINT FK\_Post\_car FOREIGN KEY(id\_AUTO)

REFERENCES CAR (id\_car);

ALTER TABLE Post ADD CONSTRAINT FK\_Post\_Spares FOREIGN KEY(id\_spares)

REFERENCES Spares (id\_spares);

ALTER TABLE Post ADD CONSTRAINT FK\_Post\_Users FOREIGN KEY(id\_user)

REFERENCES Users\_ (id\_user);

CREATE TABLE Saved(

id\_saved int NOT NULL,

id\_CAR int NULL,

ID\_SPARE INT NULL,

id\_user int NULL,

Date\_ DATE NULL,

CONSTRAINT PK\_Saved PRIMARY KEY

(

id\_saved

));

--drop table Saved;

ALTER TABLE Saved ADD CONSTRAINT FK\_Saved\_CAR FOREIGN KEY(id\_CAR)

REFERENCES CAR (id\_CAR);

ALTER TABLE Saved ADD CONSTRAINT FK\_Saved\_SPARE FOREIGN KEY(id\_SPARE)

REFERENCES SPARES (id\_SPARES);

ALTER TABLE Saved ADD CONSTRAINT FK\_Saved\_Users FOREIGN KEY(id\_user)

REFERENCES Users\_ (id\_user);

CREATE TABLE Spares(

Id\_spares int NOT NULL,

Id\_model int NULL,

Id\_pic int NULL,

Id\_user int Null,

Spares nchar(10) NULL,

Price int NULL,

CONSTRAINT PK\_Spares PRIMARY KEY

(

Id\_spares

));

--drop table Spares;

ALTER TABLE Spares ADD CONSTRAINT FK\_Spares\_Brand FOREIGN KEY(ID\_user)

REFERENCES Users\_ (ID\_user);

ALTER TABLE Spares ADD CONSTRAINT FK\_Spares\_Model FOREIGN KEY(ID\_model)

REFERENCES Model\_ (ID\_model);

ALTER TABLE Spares ADD CONSTRAINT FK\_Spares\_Pictures FOREIGN KEY(ID\_pic)

REFERENCES Pictures (ID\_picture);

CREATE TABLE Users\_(

Id\_user int NOT NULL,

Login nchar(10) NULL,

Password nchar(10) NULL,

Name nchar(10) NULL,

PRIVILEGE\_ nchar(5),

CONSTRAINT PK\_Users PRIMARY KEY

(

Id\_user

));

INSERT INTO USERS\_(ID\_USER,LOGIN,PASSWORD,NAME,PRIVILEGE\_) VALUES(0,'ADMIN','ADMIN','AADMIN','ADMIN');

CREATE TABLE Pictures(

Id\_picture int NOT NULL,

picture BLOB NULL,

Id\_car int NULL,

Id\_spare int NULL,

CONSTRAINT PK\_Pictures PRIMARY KEY

(

Id\_picture

));

# Приложение В

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение В

Лит.

Листов

1

БГТУ

create or replace PACKAGE ADMIN\_PACK AS – пакет для хранения процедур админа

Procedure ADD\_BRAND -- добавление нового бренда

(BRAND IN VARCHAR2);

Procedure ADD\_MODEL --добавление новой модели

(MODEL\_ IN VARCHAR2, BRAND1 IN BRAND.BRAND%TYPE);

Procedure ADD\_POST --добавление нового сообщения от админа

(ID\_USER IN INT, ID\_AUTO IN INT,ID\_SPARES IN INT, POST IN VARCHAR2);

END ADMIN\_PACK;

create or replace PACKAGE XML\_PACKAGE IS – пакет для работы с хмл

PROCEDURE EXPORT\_BRAND\_TO\_XML;

PROCEDURE IMPORT\_BRAND\_FROM\_XML;

END XML\_PACKAGE;

create or replace PACKAGE OBS\_PACK AS – пакет с общими процедурами

Procedure CAR\_BY\_ID --инфа о конкретном авто

(ID\_CAR1 IN INT);

Procedure CAR\_BY\_USER – информация об авто опред пользователя

(ID\_USER1 IN INT);

Procedure CARS\_BY\_BRAND – список авто определенного бренда

(BRAND1 IN VARCHAR2);

Procedure CARS\_BY\_MODEL – список авто определенной модели

(MODEL1 IN VARCHAR2);

Procedure COMMENT\_CAR – список комментариев к опред авто

(ID\_CAR1 IN INT);

Procedure COMMENT\_SPARE– список комментариев к опред детали

(ID\_SPARE1 IN INT);

Procedure DEL\_PICTURE – удалить картинку

(ID\_PIC1 IN INT);

Procedure DEL\_CAR – удалить машину

(ID\_CAR1 IN INT);

Procedure DEL\_SPARE – удалить деталь

(ID\_SPARE1 IN INT);

Procedure GET\_CARS; -- список всех машин

Procedure GET\_SPARES; -- список всех деталей

Procedure LOGIN\_IN -- вход

(LOGIN1 IN USERS\_.LOGIN%TYPE,PASSWORD1 IN USERS\_.PASSWORD%TYPE);

Procedure PICTURE\_BY\_CAR – картинки к определенной машине

(ID\_CAR1 IN INT);

PROCEDURE PICTURE\_BY\_SPARES – картинки к опред детале

(ID\_SPARE1 IN INT);

Procedure POST\_BY\_ID -- конкретное сообщение от админ

(ID\_POST1 IN INT);

Procedure POST\_BY\_USER – почта пользователя

(ID\_USER1 IN INT);

Procedure SPARE\_BY\_USER – деталь пользователя

(ID\_USER1 IN INT);

Procedure SPARES\_BY\_ID --инфа о конкретной детале

(IDS IN INT);

Procedure USER\_BY\_ID –информация о пользователе

(IDU IN INT);

Procedure USERS\_CONTACTS – контакты пользователя

(ID\_USER1 IN INT);

END OBS\_PACK;

create or replace PACKAGE USER\_PACK AS –пакет для процедур пользователя

PROCEDURE ADD\_PICTURE(PICTURE IN VARCHAR2,ID\_CAR IN INT, ID\_SPARES IN INT ); -- добавление картинки

Procedure ADD\_CAR --добавление нового авто

(ID\_USER IN INT, DESCRIPTION\_CAR IN VARCHAR2,PRICE IN INT, MODEL1 IN VARCHAR2, ID\_PIC INT DEFAULT 0);

Procedure ADD\_COMMENT --добавление нового комента

(ID\_USER IN INT, COMMENT1 IN VARCHAR2, ID\_CAR1 IN INT,ID\_SPARE1 IN INT );

Procedure ADD\_SAVED --добавление в изыбранное

(ID\_USER IN INT, ID\_CAR IN INT,ID\_SPARE IN INT,DATE1 IN DATE);

Procedure ADD\_SPARES --добавление запчасти

(ID\_USER IN INT, SPARES IN VARCHAR2,PRICE IN INT, MODEL1 IN VARCHAR2, ID\_PIC INT DEFAULT 0);

Procedure ADD\_USER --добавление нового юзера\регистрация

(LOGIN IN VARCHAR2,PASSWORD IN VARCHAR2,NAME IN VARCHAR2);

PROCEDURE UPDATE\_CAR – обновление поста о машине

(DES IN VARCHAR2,NEWPRICE IN INT, IC IN INT );

Procedure UPDATE\_SPARES – обновление поста о детале

(DES IN VARCHAR2,NEWPRICE IN INT, IDS IN INT );

Procedure ADD\_CONTACT--добавление контактов

(CONTACT IN VARCHAR2,ID\_USER IN VARCHAR2);

# Приложение Г

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Г

Лит.

Листов

1

БГТУ

<brand xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:nonamespaceSchemaLocation="http://www.oracle.com/Employee.xsd">

<BRAND>

<ID>1</ID>

<BRAND>ACURA </BRAND>

</BRAND>

<BRAND>

<ID>2</ID>

<BRAND>ALFA ROMEO </BRAND>

</BRAND>

<BRAND>

<ID>3</ID>

<BRAND>ASTON MARTIN </BRAND>

</BRAND>

<BRAND>

<ID>4</ID>

<BRAND>AUDI </BRAND>

</BRAND>

<BRAND>

<ID>5</ID>

<BRAND>BENTLEY </BRAND>

</BRAND>

<BRAND>

<ID>6</ID>

<BRAND>BMW </BRAND>

</BRAND>

<BRAND>

<ID>7</ID>

<BRAND>CADILLAC </BRAND>

</BRAND>

<BRAND>

<ID>8</ID>

<BRAND>CHANGAN </BRAND>

</BRAND>

<BRAND>

<ID>9</ID>

<BRAND>CHERY </BRAND>

</BRAND>

<BRAND>

<ID>10</ID>

<BRAND>CHEVROLET </BRAND>

</BRAND>

<BRAND>

<ID>11</ID>

<BRAND>CHRYSLER </BRAND>

</BRAND>

<BRAND>

<ID>12</ID>

<BRAND>CITROEN </BRAND>

</BRAND>

<BRAND>

<ID>13</ID>

<BRAND>DOGE </BRAND>

</BRAND>

<BRAND>

<ID>14</ID>

<BRAND>FIATE </BRAND>

</BRAND>

<BRAND>

<ID>15</ID>

<BRAND>FORD </BRAND>

</BRAND>

<BRAND>

<ID>16</ID>

<BRAND>GELLY </BRAND>

</BRAND>

<BRAND>

<ID>17</ID>

<BRAND>HAFEI </BRAND>

</BRAND>

<BRAND>

<ID>18</ID>

<BRAND>HAIMA </BRAND>

</BRAND>

<BRAND>

<ID>19</ID>

<BRAND>HAVAL </BRAND>

</BRAND>

<BRAND>

<ID>20</ID>

<BRAND>HONDA </BRAND>

</BRAND>

<BRAND>

<ID>21</ID>

<BRAND>HUMMER </BRAND>

</BRAND>

<BRAND>

<ID>22</ID>

<BRAND>HYUNDAI </BRAND>

</BRAND>

<BRAND>

<ID>23</ID>

<BRAND>INFINITY </BRAND>

</BRAND>

<BRAND>

<ID>24</ID>

<BRAND>JAGUAR </BRAND>

</BRAND>

<BRAND>

<ID>25</ID>

<BRAND>JEEP </BRAND>

</BRAND>

<BRAND>

<ID>26</ID>

<BRAND>KIA </BRAND>

</BRAND>

<BRAND>

<ID>27</ID>

<BRAND>LADA </BRAND>

</BRAND>

<BRAND>

<ID>28</ID>

<BRAND>LANCIA </BRAND>

</BRAND>

<BRAND>

<ID>29</ID>

<BRAND>LAND ROVER </BRAND>

</BRAND>

<BRAND>

<ID>30</ID>

<BRAND>LEXUS </BRAND>

</BRAND>

<BRAND>

<ID>31</ID>

<BRAND>LIFAN </BRAND>

</BRAND>

<BRAND>

<ID>32</ID>

<BRAND>LINCOLN </BRAND>

</BRAND>

<BRAND>

<ID>33</ID>

<BRAND>MASERATI </BRAND>

</BRAND>

<BRAND>

<ID>34</ID>

<BRAND>MAZDA </BRAND>

</BRAND>

<BRAND>

<ID>35</ID>

<BRAND>MERCEDES-BENZ</BRAND>

</BRAND>

<BRAND>

<ID>36</ID>

<BRAND>MITSUBISHI </BRAND>

</BRAND>

<BRAND>

<ID>37</ID>

<BRAND>NISSAN </BRAND>

</BRAND>

<BRAND>

<ID>38</ID>

<BRAND>OLDSMOBILE </BRAND>

</BRAND>

<BRAND>

<ID>39</ID>

<BRAND>OPEL </BRAND>

</BRAND>

<BRAND>

<ID>40</ID>

<BRAND>PEOGEOT </BRAND>

</BRAND>

<BRAND>

<ID>41</ID>

<BRAND>PORSCHE </BRAND>

</BRAND>

<BRAND>

<ID>42</ID>

<BRAND>RAVON </BRAND>

</BRAND>

<BRAND>

<ID>43</ID>

<BRAND>RENAULT </BRAND>

</BRAND>

<BRAND>

<ID>44</ID>

<BRAND>ROVER </BRAND>

</BRAND>

<BRAND>

<ID>45</ID>

<BRAND>SAAB </BRAND>

</BRAND>

<BRAND>

<ID>46</ID>

<BRAND>SAIPA </BRAND>

</BRAND>

<BRAND>

<ID>47</ID>

<BRAND>SANTANA </BRAND>

</BRAND>

<BRAND>

<ID>48</ID>

<BRAND>SEAT </BRAND>

</BRAND>

<BRAND>

<ID>49</ID>

<BRAND>SKODA </BRAND>

</BRAND>

<BRAND>

<ID>50</ID>

<BRAND>SMART </BRAND>

</BRAND>

<BRAND>

<ID>51</ID>

<BRAND>SUBARU </BRAND>

</BRAND>

<BRAND>

<ID>52</ID>

<BRAND>SUZUKI </BRAND>

</BRAND>

<BRAND>

<ID>53</ID>

<BRAND>TATA </BRAND>

</BRAND>

<BRAND>

<ID>54</ID>

<BRAND>TESLA </BRAND>

</BRAND>

<BRAND>

<ID>55</ID>

<BRAND>TOYOTA </BRAND>

</BRAND>

<BRAND>

<ID>56</ID>

<BRAND>VOLKSWAGEN </BRAND>

</BRAND>

<BRAND>

<ID>57</ID>

<BRAND>VOLVO </BRAND>

</BRAND>

<BRAND>

<ID>58</ID>

<BRAND>VORTEX </BRAND>

</BRAND>

<BRAND>

<ID>59</ID>

<BRAND>ГАЗ </BRAND>

</BRAND>

<BRAND>

<ID>60</ID>

<BRAND>ЗАЗ </BRAND>

</BRAND>

<BRAND>

<ID>61</ID>

<BRAND>МОСКВИЧ </BRAND>

</BRAND>

<BRAND>

<ID>62</ID>

<BRAND>УАЗ </BRAND>

</BRAND>

<BRAND>

<ID>63</ID>

<BRAND>ЭКСКЛЮЗИВ </BRAND>

</BRAND>

</brand>

# Приложение Д

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Левицкий В. С.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Д

Лит.

Листов

1

БГТУ

-- 101 IMPORT ERROR 102 EXPORT ERROR

CREATE OR REPLACE DIRECTORY UTLDATA AS 'C:\DB\XML\';

--DROP DIRECTORY UTLDATA;

--------------------------------------------------------------------------------

--DROP PACKAGE XML\_PACKAGE;

CREATE OR REPLACE PACKAGE XML\_PACKAGE IS

PROCEDURE EXPORT\_BRAND\_TO\_XML;

PROCEDURE IMPORT\_BRAND\_FROM\_XML;

END XML\_PACKAGE;

--------------------------------------------------------------------------------

CREATE OR REPLACE PACKAGE BODY XML\_PACKAGE IS

--------------------------------------------------------------------------------

PROCEDURE EXPORT\_BRAND\_TO\_XML

IS

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("USERS",

XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",

'http://www.oracle.com/Employee.xsd' AS "xsi:nonamespaceSchemaLocation"),

XMLAGG(XMLELEMENT("BRAND",

XMLELEMENT("ID",U.ID\_BRAND),

XMLELEMENT("BRAND",U.BRAND)

))

) FROM BRAND U;

BEGIN

OPEN XMLCUR;

LOOP

FETCH XMLCUR INTO XDATA;

EXIT WHEN XMLCUR%NOTFOUND;

END LOOP;

CLOSE XMLCUR;

DOC := DBMS\_XMLDOM.NewDOMDocument(XDATA);

DBMS\_XMLDOM.WRITETOFILE(DOC, 'UTLDATA/BRAND.xml');

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20102, 'EXPORT XML ERROR');

END EXPORT\_BRAND\_TO\_XML;

--------------------------------------------------------------------------------

PROCEDURE IMPORT\_BRAND\_FROM\_XML

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('UTLDATA', 'BRAND.xml');

L\_DEST\_OFFSET INTEGER := 1;

L\_SRC\_OFFSET INTEGER := 1;

L\_BFILE\_CSID NUMBER := 0;

L\_LANG\_CONTEXT INTEGER := 0;

L\_WARNING INTEGER := 0;

P DBMS\_XMLPARSER.PARSER;

V\_DOC DBMS\_XMLDOM.DOMDOCUMENT;

V\_ROOT\_ELEMENT DBMS\_XMLDOM.DOMELEMENT;

V\_CHILD\_NODES DBMS\_XMLDOM.DOMNODELIST;

V\_CURRENT\_NODE DBMS\_XMLDOM.DOMNODE;

U BRAND%ROWTYPE;

BEGIN

DBMS\_LOB.CREATETEMPORARY (L\_CLOB, TRUE);

DBMS\_LOB.FILEOPEN(L\_BFILE, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADCLOBFROMFILE (DEST\_LOB => L\_CLOB, SRC\_BFILE => L\_BFILE, AMOUNT => DBMS\_LOB.LOBMAXSIZE,

DEST\_OFFSET => L\_DEST\_OFFSET, SRC\_OFFSET => L\_SRC\_OFFSET, BFILE\_CSID => L\_BFILE\_CSID,

LANG\_CONTEXT => L\_LANG\_CONTEXT, WARNING => L\_WARNING);

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

-- Create XML Parser.

P := DBMS\_XMLPARSER.NEWPARSER;

-- Parse XML into DOM object

DBMS\_XMLPARSER.PARSECLOB(P,L\_CLOB);

-- Document Element

V\_DOC := DBMS\_XMLPARSER.GETDOCUMENT(P);

-- Root element

V\_ROOT\_ELEMENT := DBMS\_XMLDOM.Getdocumentelement(v\_Doc);

--- return Dbms\_Xmldom.Domnodelist

V\_CHILD\_NODES := DBMS\_XMLDOM.GETCHILDRENBYTAGNAME(V\_ROOT\_ELEMENT,'\*');

FOR i IN 0 .. DBMS\_XMLDOM.GETLENGTH(V\_CHILD\_NODES) - 1

LOOP

-- <USER> Node.

V\_CURRENT\_NODE := DBMS\_XMLDOM.ITEM(V\_CHILD\_NODES,i);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,

'ID\_BRAND/text()',U.ID\_BRAND);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,

'BRAND/text()',U.BRAND);

INSERT INTO BRAND(ID\_BRAND,BRAND)

VALUES(U.ID\_BRAND,U.BRAND) ;

END LOOP;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

RAISE\_APPLICATION\_ERROR(-20101, 'IMPORT XML ERROR'|| SQLERRM);

END IMPORT\_BRAND\_FROM\_XML;

--------------------------------------------------------------------------------

END XML\_PACKAGE;

--------------------------------------------------------------------------------

BEGIN XML\_PACKAGE.EXPORT\_BRAND\_TO\_XML;END;