МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Базы данных»

Тема: «Реализация базы данных салона автомобилей с использованием технологий резервного копирования и восстановления»

Исполнитель

студент 2 курса 1 группы Вакуленчик Владислав Сергеевич

(Ф.И.О.)

Руководитель старший преподаватель кафедры ИСиТ *Блинова Е.А.* (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой ­

Руководитель Блинова Е.А.

(подпись)

Минск 2023

**Оглавление**

[Введение 3](#_Toc134563032)

[1. Разбор аналогов и постановка задачи 4](#_Toc134563033)

[1.1 Обзор аналогичных решений 4](#_Toc134563034)

[1.2 Постановка задачи 8](#_Toc134563035)

[2. Разработка модели базы данных 9](#_Toc134563036)

[3. Разработка объектов базы данных 10](#_Toc134563037)

[3.1 Таблицы 10](#_Toc134563038)

[3.2 Пользователи 12](#_Toc134563039)

[3.3 Хранимые процедуры 14](#_Toc134563040)

[3.4 Индексы 14](#_Toc134563041)

[4. Описание процедур импорта и экспорта данных 16](#_Toc134563042)

[4.1 Описание процедуры импорта данных 16](#_Toc134563043)

[4.2 Описание процедуры экспорта данных 16](#_Toc134563044)

[5. Описание технологии и ее применения в базе данных 18](#_Toc134563045)

[6. Тестирование производительности 20](#_Toc134563046)

[7. Руководство пользователя 22](#_Toc134563047)

[Заключение 27](#_Toc134563048)

[Cписок источников 28](#_Toc134563049)

[ПРИЛОЖЕНИЕ А 29](#_Toc134563050)

[ПРИЛОЖЕНИЕ Б 31](#_Toc134563051)

## **Введение**

Для эффективного функционирования любой организации необходимо иметь быстрый и удобный доступ к информации, которая в настоящее время имеет высокую ценность. Базы данных часто выполняют роль распорядителей информации, обеспечивая надежное хранение структурированных данных и своевременный доступ к ним. Сегодня почти каждая организация нуждается в базе данных, которая отвечает их потребностям в хранении, управлении и администрировании данных.

За последние годы структуры данных стали более сложными. В дополнение к простым числам и текстовым строкам, информация может представляться в виде мультимедийных документов, графических изображений, хронологических рядов, процедурных или активных данных и многих других формах.

На рынке существует множество технологий доступа к данным и серверам баз данных, каждая из которых имеет свои отличительные особенности. Современные приложения для обработки данных ориентированы на работу с большим количеством пользователей, которые могут находиться удаленно от сервера базы данных.

Темой данного курсового проекта является разработка базы данных "Салона автомобилей", а также приложения, которое позволяет получать данные из базы данных, представлять их в определенном формате для просмотра пользователем, редактировать и выводить всю информацию.

Целью курсового проекта является разработка базы данных автосалона с выполнением следующих задач:

* Управление пользователями (добавление и изменение информации о пользователе, возможность удаления пользователя, вывод полной информации о пользователе);
* Обеспечение покупки авто (выбор модели, расцветки, комплектующих, отображение общей стоимости);
* Обеспечение отправки заявки на тест драйв;
* Анализ продаж автосалона (вывод информации о более/менее популярных моделях, удаление менее продаваемых товаров).

База данных "Салон автомобилей" существенно упрощает работу сотрудников магазина автомобилей и предоставляет возможность администратору своевременно вносить необходимые изменения.

1. Разбор аналогов и постановка задачи

Автомобильный салон – это место, где можно купить или продать автомобиль. В современном мире существует множество приложений, которые позволяют клиентам получать информацию о наличии автомобилей в салоне, их стоимости и характеристиках, а также управлять процессом покупки.

Примерами таких приложений являются приложение для поиска автомобилей по параметрам, приложение для онлайн-бронирования тест-драйва, приложение для отслеживания статуса заказа и оповещения о готовности автомобиля. Их суть состоит в том, чтобы помочь клиентам сформировать свой запрос, визуализировать информацию в виде графиков и диаграмм, а также осуществлять мониторинг процесса покупки и предоставление анализа результата по мере его достижения.

# **Обзор аналогичных решений**

Перед этапом создания своего приложения и разработки базы данных к нему, следует обратить внимание на аналогичные решения на рынке. Поскольку темой является автосалон, то первым аналогом будет являться салон Audi, представленный на рисунке 1.1.

|  |
| --- |
| audi |

Рисунок 1.1 – главная страница автосалона Audi;

Дизайн веб-сайта выполнен в не нагружающих светлых тонах, а в навигации присутствуют все основные элементы управления. Снизу справа выскакивает всплывающее окно с консультацией. При переходе в каталог через навигацию (рисунок 1.2), открывается отдельное окне автомобиля, которое представляет обширную информацию о модели.

|  |
| --- |
|  |

Рисунок 1.2 – переход в каталог всех авто;

Преимуществами данного веб-сайта является:

- разнообразная информация о каждом автомобиле, с предоставлением вида салона и возможностью просмотра его через 360-грудосов камеру;

- большой выбор сервиса поддержки;

- возможность заказать тест-драйв, посмотреть актуальные новости и ближайшее расположение сервисов обслуживания.

Недостатками веб-сайта является:

- страница модели долго загружается, в силу того, что она нагружена большим количеством тяжелых графических элементов;

- отсутствует бот в сервисе техподдержки, который помог бы моментально отвечать на все самые популярные вопросы.

Вывод: модели Audi пользуются спросом по всему миру, а веб-сайт корректно и в удобной форме предоставляет их услуги: информационные, новостные, сервиса обслуживания и заказа консультации или тест-драйва. Веб-сайт содержит данные о пользователе: почту, номер телефона и пароль.

Следующим аналогом является автосалон Bugatti. Несмотря на то, что на главной странице их веб-сайта преобладает пустое и незанятое пространство, благодаря правильно подобранной цветовой палитре, он выглядит стильно и элегантно (Рисунок 1.3).

|  |
| --- |
| 1 |

Рисунок 1.3 – Веб-сайт автосалона Bugatti;

Преимуществами сервиса является:

- отлично проработанная адаптация под мобильные устройства;

- простой и не нагруженный дизайн;

- быстрая прогрузка страниц;

Недостатки:

- отсутствие возможности заказать тест-драйв;

- нету точек расположения автосалонов или пунктов обслуживания;

- нету контактов связи техподдержки.

Вывод: в связи с тем, что автомобили Bugatti относятся к классу роскошных и дорогостоящий моделей, то веб-сайт их автосалона не предоставляет множество информации о покупке, обслуживании или консультации. Тем не менее, присутствует блок актуальных новостей и подробное описание каждой модели салона. К сожалению, веб-сайт не содержит регистрационной формы, однако может предоставить информацию о ближайших автосалонах компании.

Последним аналогом является автосалон Nissan. Их веб-сайт выполнен в белых и черных тонах. По навигации сверху на главной странице можно перейти к любой популярной модели. А если нужного авто там нету, то можно воспользоваться поиском. Справа снизу есть блок соцсетей техподдержки. Дизайн веб-сайта представлен на рисунке 1.4.

|  |
| --- |
| 2 |

Рисунок 1.4 – Веб-сайт автосалона Nissan;

Преимуществам веб-сайта является:

- обширный инструментарий для подборки идеальной модели;

- множество вариантов связи с техподдержкой;

- присутствуют в избытке как поддержанные и практичные автомобили, так и спортивные;

Недостатки:

- долгая прогрузка некоторых страниц из-за большого количества графических элементов.

Общий вывод по обзору аналогичных решений: в ходе обзора аналогичных решений были рассмотрены различные веб-сайты, предоставляющие услуги своих автосалонов. Некоторые из них хранят информацию о пользователях и автомобилях, а другие – о местоположении автосалонов. В данном курсовом проекте, планируется хранить информацию о пользовательских данных: почта, фамилия, пароль - о сведениях моделей авто: цвет, марка, модель, оборудование - а также создание нескольких ролей: администратор и авторизированный пользователь.

# **Постановка задачи**

Задача проекта: разработать архитектуру приложения, создать интерфейс, взаимодействие с которым будет понятно любому пользователю. Построить базу данных и интерфейс и выполнить тестирование готового продукта.

Функционально должны быть выполнены следующие задачи (полный список задач изображен на Use Case диаграмме, представленной на рисунке 1.5):

* разработка приложения для взаимодействия с БД;
* регистрация новых пользователей и их авторизация;
* возможность оформить заказ автомобиля;
* возможность осуществлять поиск автомобиля;

|  |
| --- |
|  |

Рисунок 1.5 – Use Case диаграмма проекта;

Должны быть выполнены следующие требования:

* доступ к данным должен осуществляться только через соответствующие процедуры;
* должен быть проведен импорт данных из XML файлов, экспорт данных в формат XML;
* необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице;
* применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.

Форма реализации продукта – десктоп приложение.

Десктоп приложение должно быть реализовано с использованием языка С#. Технология для реализации – WPF. База данных приложения должна взаимодействовать с Oracle Database 21c.

# **Разработка модели базы данных**

Первым этапом курсового проекта будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуру базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

Диаграмма базы данных, спроектированная в ходе разработки показана на рисунке 2.1.

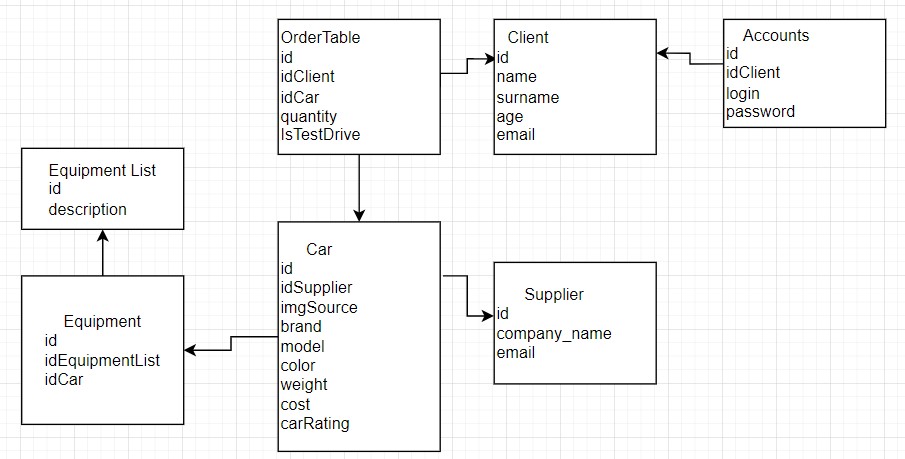


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 7 таблиц, взаимосвязанных между собой внешними и прямыми ключами:

* Car;
* Equipment;
* EquipmentList;
* Supplier;
* Order;
* Client;

- Account;

# **Разработка объектов базы данных**

При разработке приложения для курсового проекта была использована созданная база данных DB13 в рамках СУБД Oracle.

# **Таблицы**

Для реализации базы данных «Магазин автомобилей» было разработано 7 таблиц: Car, Equipment, EquipmentList, Supplier, Order, Client, Account.

Таблица Clients представляет список клиентов, состоит из столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы Client

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор клиента, первичный ключ | int |
| surname | фамилия клиента | varchar2 |
| name | имя клиента | varchar2 |
| age | возраст | int |
| email | почта | varchar2 |

Таблица Account представляет список учётных записей клиентов в системе, таблица состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы Account

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| login | логин пользователя | varchar2 |
| password | пароль пользователя | varchar2 |
| idClient | идентификатор клиента, внешний ключ для связи с таблицей Client | int |

Таблица Supplier представляет список поставщиков автомобилей, состоит из столбцов (таблица 3.3):

Таблица 3.3 – Столбцы таблицы Supplier

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор поставщика, первичный ключ | int |
| company name | название компании | varchar2 |
| email | почта поставщика | varchar2 |

Таблица Orders представляет заказы клиента, состоит из столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы OrderTable

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | Идентификатор заказа | int |
| quantity | Количество товара | int |
| idClient | Идентификатор клиента, внешний ключ для связи с таблицей Client | int |
| idCar | Идентификатор автомобиля, внешний ключ для связи с таблицей Car | int |
| IsTestDrive | 1 – если авто записано на тест-драйв, 0 – если нет | int |

Таблица Cars представляет список автомобилей, которые будет искать и заказывать пользователь через разработанное приложение. Данная таблица состоит из столбцов (таблица 3.5):

Таблица 3.5 – Столбцы таблицы Car

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор автомобиля, тип, первичный ключ | int |
| idSupplier | идентификатор поставщика, тип, внешний ключ для связи с таблицей Supplier | int |
| model | наименование модели авто | varchar2 |
| imgSource | Хранит путь до картинки авто | varchar2 |
| brand | марка автомобиля | varchar2 |
| color | цвет авто | varchar2 |
| cost | цена автомобиля | int |
| weight | вес авто | varchar2 |
| carRating | Рейтинг авто | int |

Таблица Equipment представляет выбранное оборудование для определенного автомобиля, которое идёт в комплекте с его покупкой при заказе, поэтому связана с таблицей Car и EquipmentList, и состоит из столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы Equipment

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор оснащения, прямой ключ | int |
| idEquipmentList | Идентификатор комплектации, внешний | int |

Продолжение таблицы 3.6

|  |  |  |
| --- | --- | --- |
|  | ключ для связи с таблицей EquipmentList |  |
| idCar | Идентификатор автомобиля, внешний ключ для связи с таблицей Car | int |

Таблица EquipmentList представляет оборудование автомобиля и состоит из столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы EquipmentList

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор оснащения, прямой ключ | int |
| Description | Описание комплектующей | varchar2 |

Данные таблицы оснащены всеми необходимыми переменными для хранения нужной информации как о клиентах и их заказах, так и автомобилях, и их поставщиках.

Листинг SQL-кода для создания таблиц находится в приложении А.

# **Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

Сперва было создано общее постоянное и временное табличное пространство для двух пользователей, представленное на рисунке 3.2.1.

|  |
| --- |
|  |

Рисунок 3.2.1 – табличные пространства для пользователей

Далее профиль безопасности для пользователей, представленный на рисунке 3.2.2.

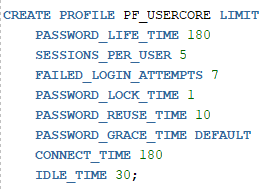


Рисунок 3.2.2. – профиль безопасности для пользователей

При проектировании базы данных было использовано 2 пользователя. Первый пользователь – сотрудник, который является системным администратором, имеет доступ для чтения и изменения таблиц, связанных с клиентами, машинами, заказами, поставщиками. Может добавить/удалить запрос на покупку/тест-драйв авто. Его роль и сам пользователь представлен на рисунке 3.2.3.

|  |
| --- |
|  |

Рисунок 3.2.3 – роль для администратора и пользователь бд для него

Вторая роль – пользователь, которой имеет возможность зарегистрироваться, авторизоваться, получить доступ к таблице с автомобилями, оборудованием и может оформить заказ автомобиля и тест-драйва. Его роль и сам пользователь представлен на рисунке 3.2.4.

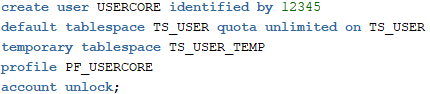


Рисунок 3.2.4 – роль для администратора и пользователь бд для него

# **Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. При создании хранимой процедуры можно определить необязательный список параметров. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Все хранимые процедуры, созданные в данном курсовом проекте, содержат обработку исключений, для того, чтобы ограничить пользователя от возможных ошибок.

При разработке курсового проекта были созданы следующие процедуры:

* getallOrderinfo - выборка данных из таблицы OrderTable, Car, Eqiupment, EqiupmentList, Client;
* AddNewOrder - добавление заказа;
* UpdateOrder – изменение заказа;
* DeleteOrder – удаление заказа;
* UpdateQuantity – изменение количества заказа (увеличение);
* UpdateQuantityDecrement – изменение количества заказа (уменьшение);
* UpdateIsTestDrive – подача заявки на тест-драйв;
* UpdateIsTestDriveToZero – отмена подачи заявки на тест-драйв;
* getClients – выборка данных о клиентах и их аккаунтов;
* insertuser – добавление нового клиента;
* updateclient – изменение информации о пользователе;
* deleteclient – удаление клиента;
* getAllCarInfo – выборка данных об авто, их комплектующих и поставщиков;
* insertcar – добавление нового клиента;
* updatecar – изменение информации о пользователе;
* deletecar – удаление клиента;
* getallsupplierinfo - выборка данных о поставщиках авто;
* XMLImportClient – импорт данных о клиентах из xml-файла;
* XMLExportClient – экспорт данных о клиентах в xml-файл.

# **Индексы**

Индекс является объектом базы данных, который создаётся с целью повышения скорости поиска данных. При работе с большим количеством строк в таблицах базы данных, которые хранятся в произвольном порядке, поиск данных по заданному критерию может занимать значительное количество времени, если происходит последовательный просмотр таблицы строка за строкой. Для ускорения поиска были созданы индексы столбцов idClient, idSupplier, idEquipmentList, idCar. Пример кода можно увидеть на рисунке 3.4.1.

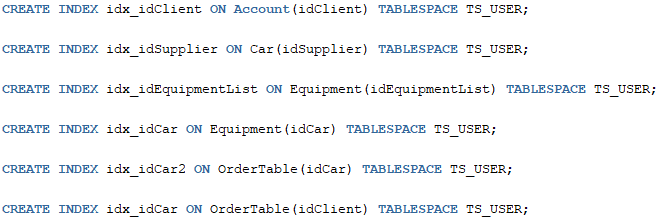


Рисунок 3.4.1 – Код индексов для ускорения поиска в бд;

Таким образом, процесс разработки индексов прошел успешно, а время поиска для таблиц Account, Car, Equipment, OrderTable было ускорено.

# **Описание процедур импорта и экспорта данных**

В данном курсовом проекте реализованы процедуры экспорта и импорта данных из XML файла в базу данных таблицы Client и наоборот.

# **Описание процедуры импорта данных**

PL/SQL инструкция Xmltype позволяет импортировать данные из файла непосредственно в таблицу базы данных или несекционированное представление. Для передачи исходного текста XML в конструктор XMLType можно использовать значения типов CLOB, BLOB, VARCHAR2, BFILE. Стоит отметить, что при использовании BFILE можно загрузить файл только с файловой системы сервера, а не клиента. Если XML-файл находится на стороне клиента и является достаточно большим для передачи в виде строки в запросе, необходимо рассмотреть возможность доставки файла с содержимым XML на файловую систему сервера. Код процедуры импорта данных в формат XML представлен на рисунке 4.1.



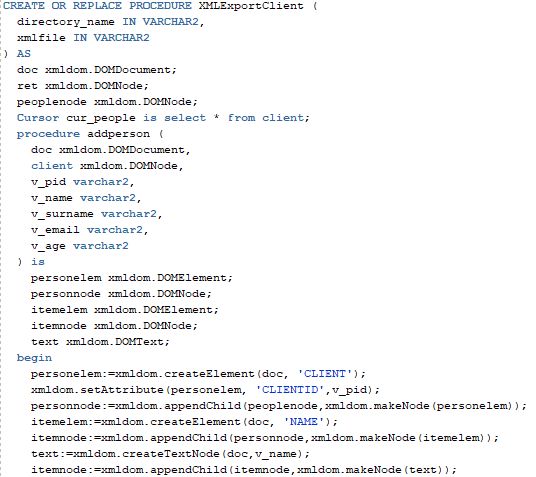
Рисунок 4.1 – Процедура импорта данных из xml в таблицу Driver

# **Описание процедуры экспорта данных**

Для осуществления преобразования применяется инструкция xmldom.

Объектная модель документа (DOM) — это API для документов HTML и XML, которая определяет логическую структуру документов и предоставляет доступ к ним и возможность их манипуляции.

В спецификации DOM термин «документ» используется в широком смысле. XML применяется для представления различных видов информации, которая может храниться в различных системах. Эта информация обычно относится к «данным», однако XML представляет эти данные в виде документов, и пакет DBMS\_XMLDOM позволяет получать доступ как к документам, основанным на схеме, так и к документам, не основанным на схеме. Код процедуры экспорта представлен на рисунке 4.2.



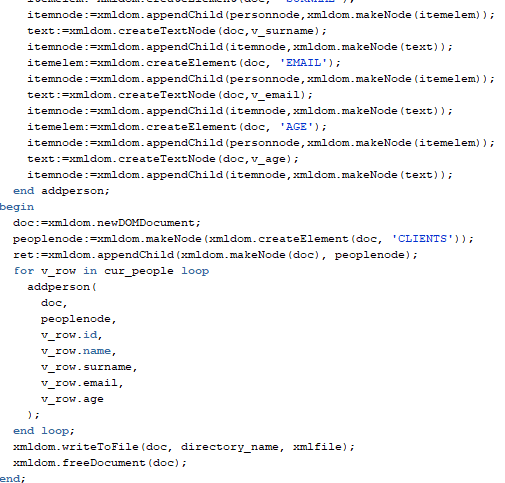


Рисунок 4.2 – Процедура экспорта

При вызове вышеуказанных процедур, данные из таблицы Client успешно импортируются в xml-файл и экспортируются обратно.

# **Описание технологии и ее применения в базе данных**

В процессе разработки проекта была использована технология: резервное копирование и восстановление данных.

Традиционный пользовательский метод резервного копирования состоит в применении команд операционной системы для копирования необходимых файлов в другое место и/или на ленточное устройство.

В процессе разработки проекта резервное копирование и восстановление данных происходило при помощи утилиты RMAN. Резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных. RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE и фрагментов резервных копий RMAN.

Возможности RMAN включают следующее:

* выполнение полного резервирования и резервирования изменений;
* выполнение холодного/горячего резервирования;
* обнаружение поврежденных блоков;
* параллельное выполнения операций ввода/вывода;
* автоматическое протоколирование операций копирования и восстановления.

С помощью RMAN можно выполнять инкрементное резервное копирование. Размер резервных копий в таком случае зависит не от размера базы данных, а скорее от уровня активности внутри нее, поскольку во время инкрементного резервного копирования не измененные блоки пропускаются.

Существует ряд сущностей, которые позволяют утилите RMAN выполнять ее функции в области резервного копирования и восстановления. В данном случае была использована целевая база данных (target database). Так называется база данных, в отношении которой RMAN выполняет резервное копирование. Все операции по резервному копированию и восстановлению осуществляются при помощи запускаемых в целевой базе данных серверных сеансов RMAN.

Подключаться к RMAN можно путем ввода в приглашении операционной системы команды rman. После этого будет появляться приглашение RMAN>, позволяющее вводить различные команды RMAN.

Резервирование файлов базы данных: горячее полное резервирование БД и холодное резервирование БД.

Горячее резервирование:

* может выполняться в состоянии СУБД OPEN;
* может выполняться только при включенном режиме архивирования журналов.

В курсовом проекте был использован метод холодного резервирования. Режим архивирования журналов выключен, СУБД выполняется в режиме NOMOUNT. Очевидный недостаток данного метода: из «холодной» резервной копии можно восстановить только то состояние базы данных, которое было в момент останова; транзакции, сделанные после рестарта базы, в «холодную» резервную копию не попадут;

Для резервирования файлов данных используется команда: BACKUP DATABASE TAG "FULL\_DATABASE\_DATAFILES". Результат можно посмотреть на рисунке 5.1, рисунке 5.2, рисунке 5.3.

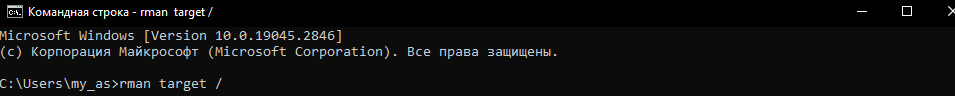


Рисунок 5.1 – Подключение к rman;



Рисунок 5.2 – Изменение пути сохранения файлов копирования;

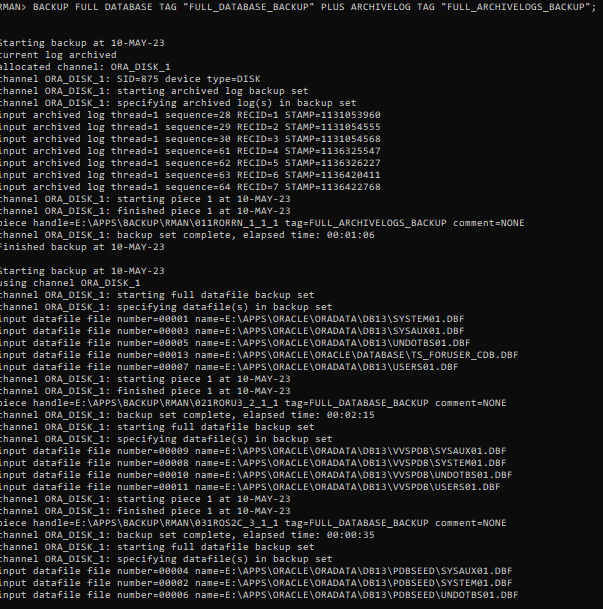


Рисунок 5.3 – Резервирование файлов данных

Для резервирования контрольных файлов используется команда: BACKUP CURRENT CONTROLFILE TAG "FULL\_DATABASE\_CONTROLFILE". Результат можно посмотреть на рисунке 5.4. А список файлов резервных копий на рисунке 5.5.

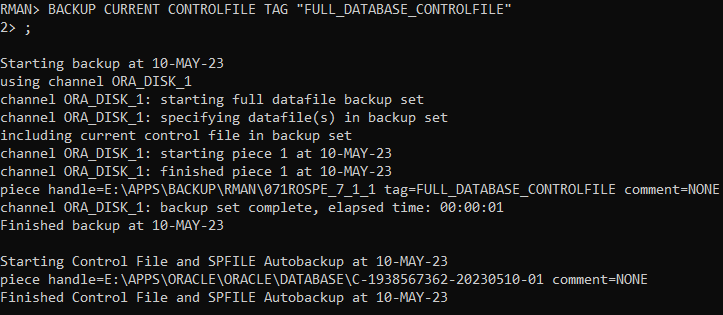


Рисунок 5.4 – Резервирование управляющих файлов

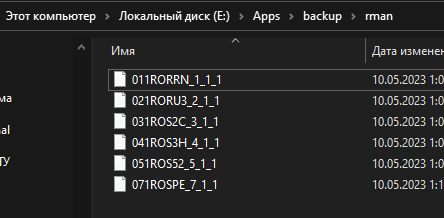


Рисунок 5.5 – Список файлов копии;

Чтобы получить информацию о созданных бэкапах нужно выполнить команду: RMAN> LIST BACKUP;

Восстановление данных. Для восстановления данных целевая БД должна находиться в состоянии NOMOUNT/ MOUNT/ OPEN в зависимости от характера восстановления, например:

* NOMOUNT: для восстановления контрольных файлов БД (фактически – СУБД)
* MOUNT: для восстановления БД целиком или табличного пространства SYSTEM
* OPEN: для восстановление табличных пространств, помимо SYSTEM (в этом случае перед процедурой восстановления само табличное пространство потребуется перевести в состояние OFFLINE).

# **Тестирование производительности**

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса.

Для данной задачи мы создали анонимный блок, и так как работаем именно с действующей базой данных, то можем использовать готовую процедуру для вставки данных в таблицу. Разработанный анонимный блок позволяет добавить большое количество строк за одно выполнение (рисунок 6.1).

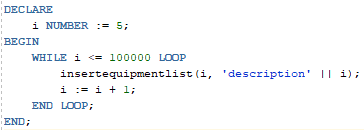




Рисунок 6.1 – Анонимный блок для заполнения таблицы большим количеством данных;

Таким образом было добавлено 100000 строк в таблицу EquipmentList.

Далее протестируем стоимость выполнения запроса к нескольким таблицам с использованием конструкции join. Запрос представлен на рисунке 6.2, его стоимость – 6.3.

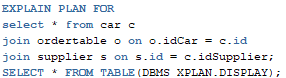


Рисунок 6.2 – запрос к таблицам OrderTable, Car, Supplier

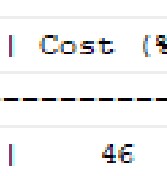


Рисунок 6.3 – стоимость запроса до создания индексов

Далее создадим индекс на внешние ключи, которые используются в данном запросе, и повторно проверим стоимость запроса (рисунок 6.4).



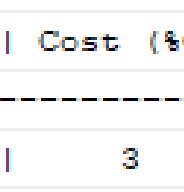


Рисунок 6.4 – стоимость запроса после создания индексов

Как видим, стоимость запроса уменьшилась. Однако, даже без индексов база данных успешно прошла тестирование на производительность.

# **Руководство пользователя**

Для работы с базой данных было разработано приложение на WPF. Сперва было создано два подключения к базе данных: от администратора и обычного пользователя. После чего, с помощью Entity Framework были сформированы edm модели для каждого подключения, которые и служили доступом к объектам базы данных.

По открытию приложения, пользователя ожидает форма авторизации (рисунок 7.1).

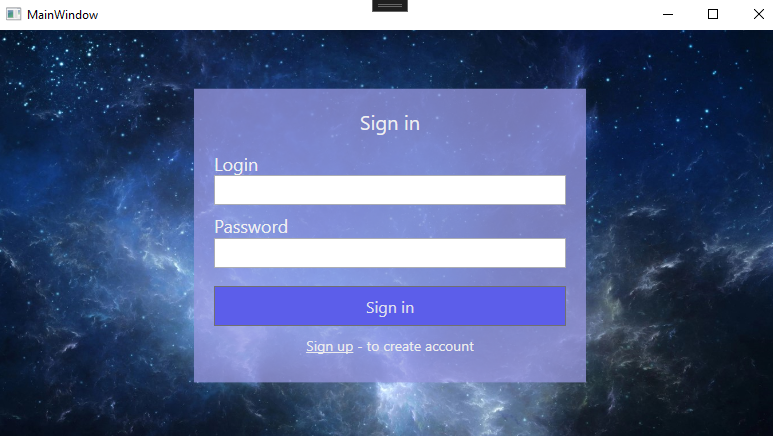


Рисунок 7.1 – форма авторизации;

При вводе логина, которого не существует в базе данных или неправильного пароля, выскакивает соответствующая ошибка (рисунок 7.2).

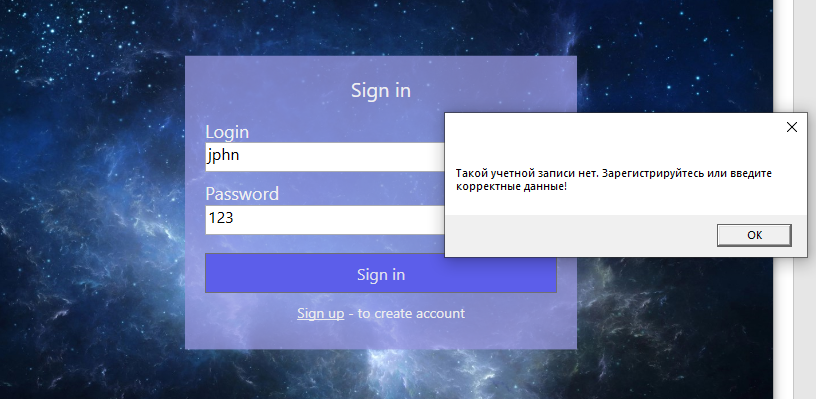


Рисунок 7.2 – форма авторизации;

Пользователю доступна возможность зарегистрироваться, если до этого у него не было аккаунта. Для этого ему достаточно кликнуть по надписи Sign up снизу кнопки Sign in. После чего откроется окно, где пользователю будет необходимо заполнить нужную информацию (рисунок 7.3).

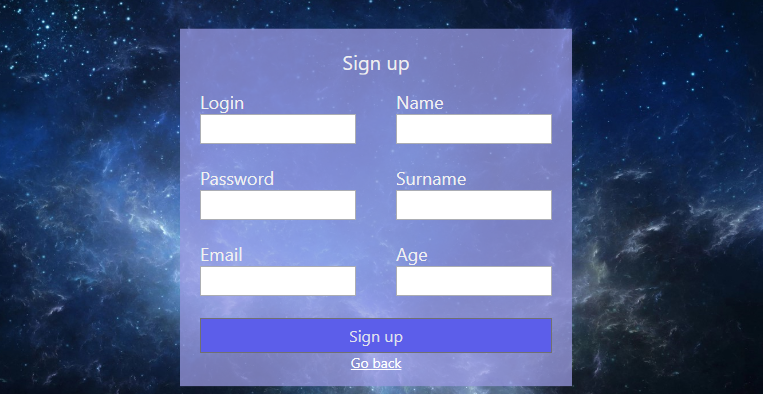


Рисунок 7.3 – форма регистрации;

Если же пользователь введет корректные данные, то его перенесет на окно каталога всех автомобилей (рисунок 7.4), которые он может заказать (или отменить заказ) и заказать тест-драйв (или отменить заявку). В таком случае, ему выведется соответствующее сообщение об успешной отправке заявки (рисунок 7.5).

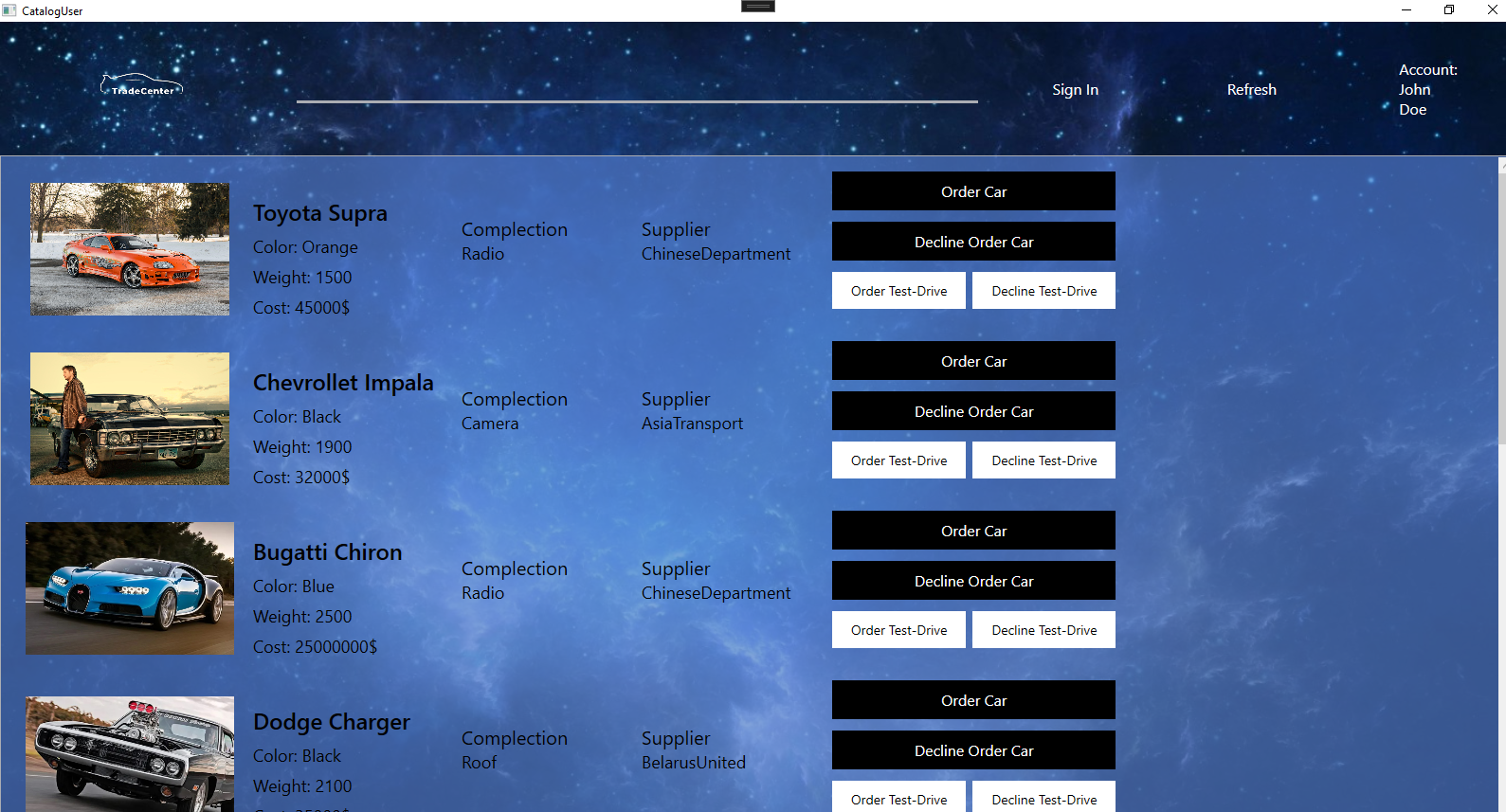


Рисунок 7.4 – каталог всех авто;

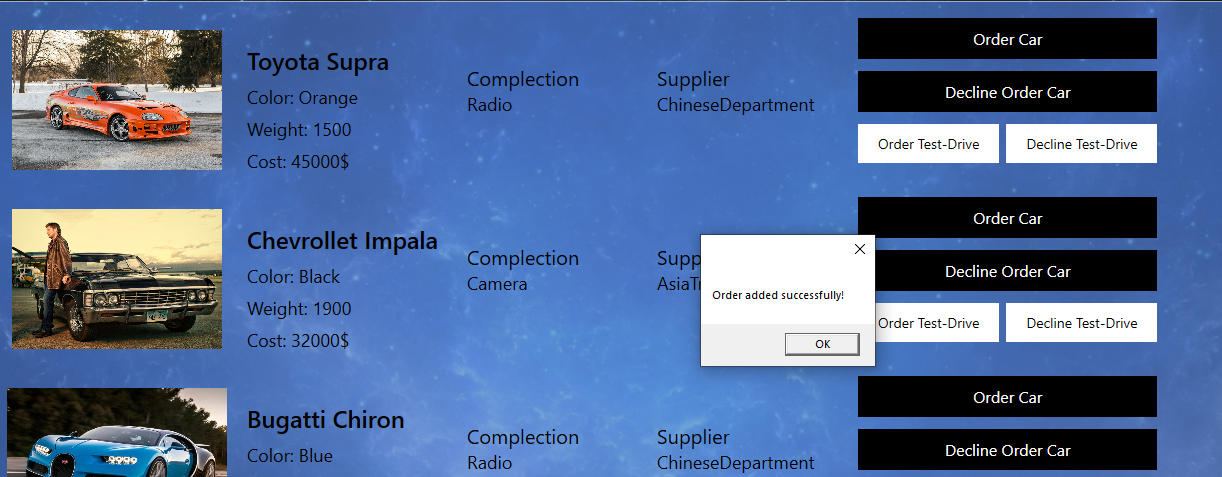


Рисунок 7.5 – успешная отправка заказа на автомобиль Toyota Supra;

Как видно на рисунке выше, пользователю доступен поиск для нахождения нужного авто, есть возможность перейти в другой аккаунт (кнопка Sign in), отображается текущая учетная запись (Account: John Doe), а в самом каталоге у каждого авто есть 4 кнопки, которые ему позволяют заказать авто (Order Car), отменить заказ (Decline Order Car), отправить заявку на тест-драйв (Order Test-Drive), отменить отправку заявки на тест-драйв (Decline Test-Drive).

Если перейти обратно на форму авторизации (Sign in) и ввести данные администратора (как показано на рисунке 7.6), то пользователю откроется панель управления приложением и базой данных.

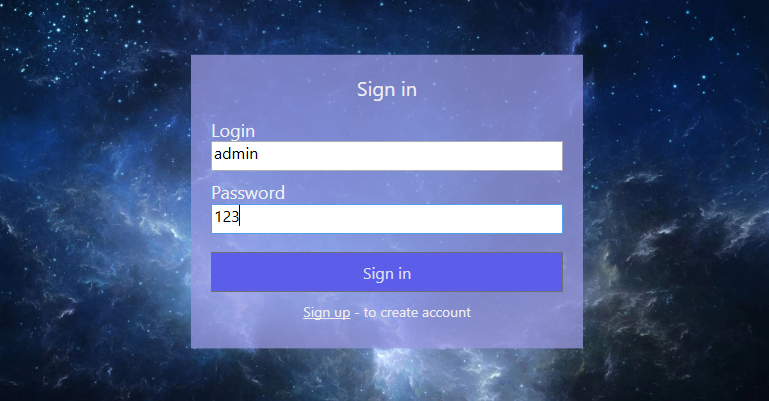


Рисунок 7.6 – вход в учетную запись администратора;

В каждой вкладке отображается полная информация о самых важных объектах базы данных: Заказов (рисунок 7.7), Клиентов (рисунок 7.9), Автомобилях (рисунок 7.10) и Поставщиках авто (рисунок 7.11). С помощью кнопок снизу можно добавить (Add, рисунок 7.8), удалить (Delete), обновить (Edit) строку, а также подгрузить новое состояние базы данных в приложение (Refresh). А с помощью кнопки Sign in можно вернуться обратно к форме авторизации.

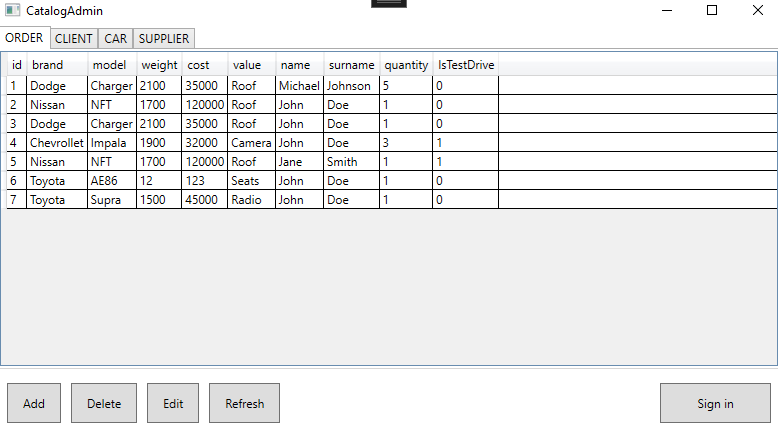


Рисунок 7.7 – панель управления базой данных: таблица ORDER;

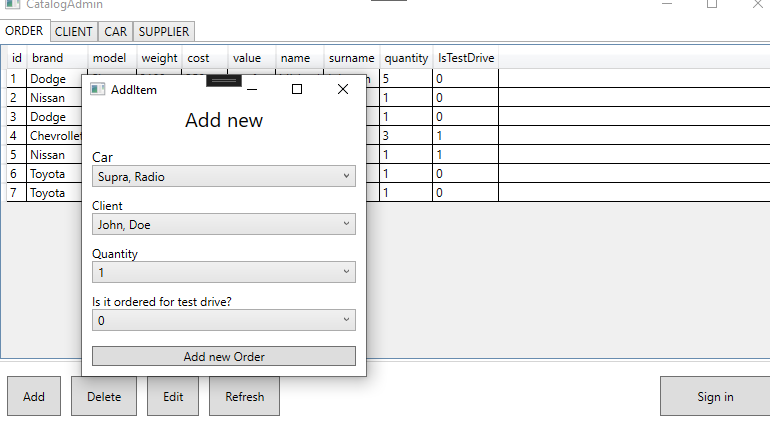


Рисунок 7.8 – добавление нового заказа;

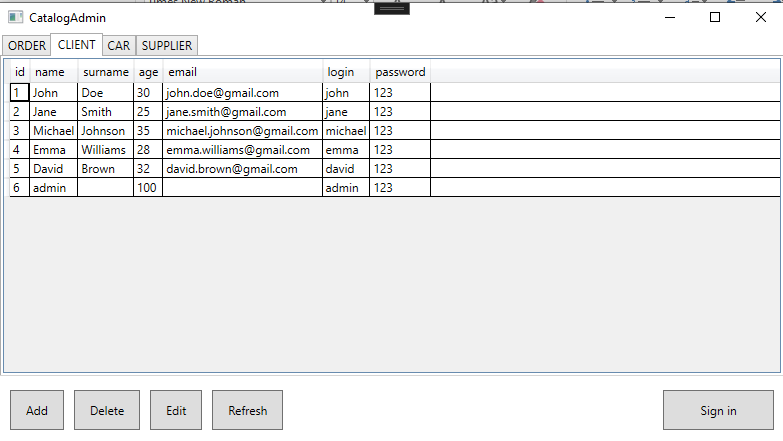


Рисунок 7.9 – панель управления базой данных: таблица CLIENT;

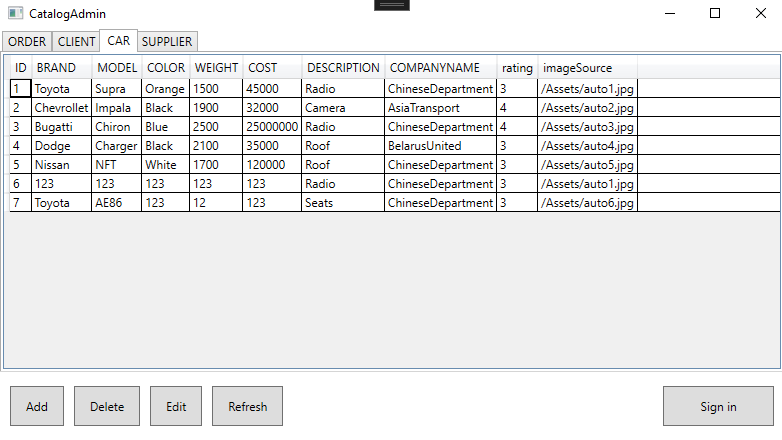


Рисунок 7.10 – панель управления базой данных: таблица CAR;

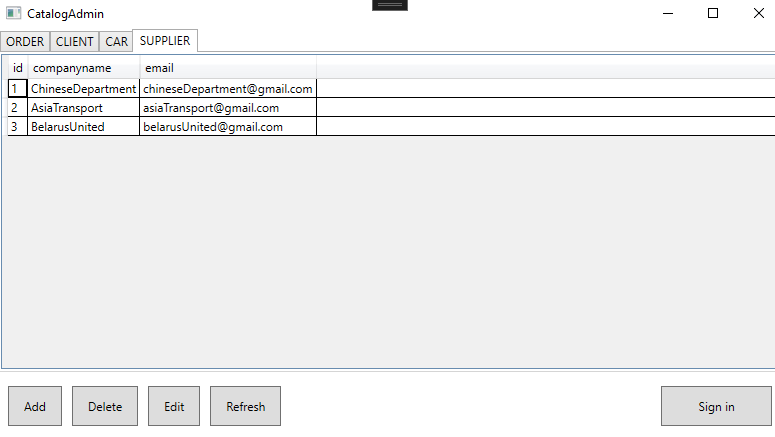


Рисунок 7.11 – панель управления базой данных: таблица SUPPLIER;

Если после добавление заказа со стороны пользователя, со стороны администратора обновить состояние приложения (кнопка Refresh), то ему отобразиться новый заказ, который он сможет подредактировать или удалить. Стоит также отметить, что если пользователь будет заказывать одни и те же модели, то будет меняться счетчик, который будет записывать свое значение в поле quantity. При достижении в 5 одинаковых заказов, пользователю будет недоступно последующий заказ того же авто. А при отмене заказа, счетчик quantity будет уменьшаться и при достижении в 0, заказ будет автоматически удаляться из базы данных.

Исходя из проделанной работы, мы можем отметить, что разработанное приложение правильно выполняет свои функции: как регистрация и авторизация клиента, так и защита от некорректных данных, которые вводятся пользователем. При этом приложение чётко разграничивает работу системного администратора, который проходит авторизацию для просмотра и редактирования данных о клиенте, и клиента, который проходит авторизацию для выполнения своих основных операций.

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных для программного средства «Салон автомобилей», которая в совокупности с приложением формирует полноценное desktop-приложение для управления магазином. В данной работе использовалось СУБД Oracle DataBase 21c, а формирование desktop-приложения осуществлялось на языке C# с помощью системы для построения клиентских приложений Windows – WPF. При разработке курсового проекта использовались объекты: таблицы, хранимые процедуры, индексы.

Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло облегчить взаимодействие с базой данных посредством программного интерфейса. Взаимодействие между приложением и базой данных было реализовано с помощью технологии Entity Framework. При разработке выполнены следующие пункты:

* авторизация и регистрация пользователей;
* составление интерфейса для взаимодействия с БД;
* возможность выбора и заказа понравившегося автомобиля;
* изменение данных пользователя через учётную запись администратора;

Приложение прошло тестирование при использовании в БД большого количество данных. Также были реализованы процедуры для импорта, экспорта данных в формат XML.

Была реализована технология «Резервное копирование и восстановление». В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

# **Cписок источников**

1. Язык C# и .NET Framework [Электронный ресурс] / Режим доступа – URL: http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php – Дата доступа: 18.03.2023.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] / Режим доступа – URL: – https://metanit.com/sharp/adonet/. – Дата доступа: 22.03.2023.
3. Документация Oracle [Электронный ресурс] / Режим доступа – URL: https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm – Дата доступа: 16.03.2023.
4. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL: <https://www.oracle.com/database/database-vault/index.html> – Дата доступа: 17.03.2023.

5 Продукты Oracle [Электронный ресурс] / Режим доступа – URL: http://www.interface.ru/home.asp?artId=24678 – Дата доступа: 17.03.2023.

ПРИЛОЖЕНИЕ А

Создание базы данных

|  |
| --- |
| CREATE TABLE Client (  id INT PRIMARY KEY,  name VARCHAR(40),  surname VARCHAR(40),  age INT,  email VARCHAR(50)  ) TABLESPACE TS\_USER;  CREATE TABLE Account (  id INT PRIMARY KEY,  idClient INT,  login VARCHAR(40),  password VARCHAR(40),  FOREIGN KEY (idClient) REFERENCES Client(id)  ) TABLESPACE TS\_USER;  CREATE TABLE EquipmentList (  id INT PRIMARY KEY,  description VARCHAR2(40)  ) TABLESPACE TS\_USER;  CREATE TABLE Supplier (  id INT PRIMARY KEY,  company\_name VARCHAR(40),  email VARCHAR(40)  ) TABLESPACE TS\_USER;  CREATE TABLE Car (  id INT PRIMARY KEY,  idSupplier INT,  imgSource varchar(40),  brand VARCHAR(40),  model VARCHAR(40),  color VARCHAR(40),  weight FLOAT,  cost FLOAT,  carRating int,  FOREIGN KEY (idSupplier) REFERENCES Supplier(id),  CONSTRAINT CHK\_Rating CHECK (carRating IN (1, 2, 3, 4, 5))  ) TABLESPACE TS\_USER;  CREATE TABLE Equipment (  id INT PRIMARY KEY,  idEquipmentList INT,  idCar int,  FOREIGN KEY (idEquipmentList) REFERENCES EquipmentList(id),  FOREIGN KEY (idCar) REFERENCES Car(id)  ) TABLESPACE TS\_USER;  CREATE TABLE OrderTable (  id INT PRIMARY KEY,  idClient INT,  idCar INT,  quantity INT,  IsTestDrive number(1),  FOREIGN KEY (idClient) REFERENCES ADMINCORE.Client(id),  FOREIGN KEY (idCar) REFERENCES ADMINCORE.Car(id),  CONSTRAINT CHK\_IsTestDrive CHECK (IsTestDrive IN (0, 1))  ) TABLESPACE TS\_USER; |

ПРИЛОЖЕНИЕ Б

Создание процедур

|  |
| --- |
| CREATE OR REPLACE PROCEDURE insertuser(  insertlogin IN account.login%TYPE,  insertpassword IN account.password%TYPE,  insertname IN client.name%TYPE,  insertsurname IN client.surname%TYPE,  insertage IN client.age%TYPE,  insertemail IN client.email%TYPE  )  IS  clientid NUMBER;  accountid NUMBER;  BEGIN  SELECT COALESCE(MAX(id), 0) + 1 INTO clientid FROM client;  SELECT COALESCE(MAX(id), 0) + 1 INTO accountid FROM account;  INSERT INTO client(id, name, surname, age, email)  VALUES (clientid, insertname, insertsurname, insertage, insertemail);  INSERT INTO account(id, login, password)  VALUES (accountid, insertlogin, insertpassword);  COMMIT;  END;  create or replace procedure getallcarinfo (  cars out sys\_refcursor  )  IS  BEGIN  OPEN cars FOR  SELECT c.id, c.idsupplier, c.imgsource, c.brand, c.model, c.color, c.weight, c.cost, c.carrating,  e.idCar, e.idEquipmentList,  l.description,  s.company\_name, s.email  FROM ADMINCORE.Car c  JOIN ADMINCORE.Equipment e ON c.id = e.idCar  Join ADMINCORE.EquipmentList l on e.idEquipmentList = l.id  Join ADMINCORE.Supplier s on c.idSupplier = s.id  order by id;  END;  CREATE OR REPLACE PROCEDURE AddNewOrder(  p\_id IN INT,  p\_idClient IN INT,  p\_idCar IN INT,  p\_quantity IN INT,  p\_IsTestDrive IN NUMBER  )  IS  BEGIN  INSERT INTO admincore.OrderTable (id, idClient, idCar, quantity, IsTestDrive)  VALUES (p\_id, p\_idClient, p\_idCar, p\_quantity, p\_IsTestDrive);    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('New order has been added successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE UpdateIsTestDrive(orderId IN INT)  IS  BEGIN  UPDATE admincore.OrderTable SET IsTestDrive = 1 WHERE id = orderId;  COMMIT;  END;  CREATE OR REPLACE PROCEDURE UpdateIsTestDriveToZero(orderId IN INT)  IS  BEGIN  UPDATE OrderTable SET IsTestDrive = 0 WHERE id = orderId;  COMMIT;  END;  CREATE OR REPLACE PROCEDURE getclients(  clients OUT SYS\_REFCURSOR  )  IS  BEGIN  OPEN clients FOR  SELECT c.name, c.surname, c.age, c.email, a.login, a.password  FROM client c  JOIN account a ON c.id = a.idClient;  END;  create or replace procedure getallOrderinfo (  orders out sys\_refcursor  )  IS  BEGIN  OPEN orders FOR  SELECT o.id, c.brand, c.model, c.weight, c.cost, el.description, cl.name, cl.surname, o.quantity, o.IsTestDrive, o.idCar  FROM OrderTable o  Join Car c on o.idCar = c.id  JOIN Equipment e ON c.id = e.idCar  JOIN EquipmentList el ON e.idEquipmentList = el.id  Join Client cl on o.idClient = cl.id  order by o.id;  END;  CREATE OR REPLACE PROCEDURE UpdateOrder(  p\_id IN INT,  p\_idClient IN INT,  p\_idCar IN INT,  p\_quantity IN INT,  p\_IsTestDrive IN NUMBER  )  IS  BEGIN  UPDATE OrderTable SET idClient = p\_idClient where id = p\_id;  UPDATE OrderTable SET idCar = p\_idCar where id = p\_id;  UPDATE OrderTable SET quantity = p\_quantity where id = p\_id;  UPDATE OrderTable SET IsTestDrive = p\_IsTestDrive where id = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Order has been updated successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE DeleteOrder(  p\_id IN INT  )  IS  BEGIN  DELETE from OrderTable where id = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Order has been deleted successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  create or replace procedure getallsupplierinfo (  supplier out sys\_refcursor  )  IS  BEGIN  OPEN supplier FOR  SELECT id, company\_name, email  FROM Supplier;  END;  CREATE OR REPLACE PROCEDURE UpdateClient(  p\_id in client.id%TYPE,  insertlogin IN account.login%TYPE,  insertpassword IN account.password%TYPE,  insertname IN client.name%TYPE,  insertsurname IN client.surname%TYPE,  insertage IN client.age%TYPE,  insertemail IN client.email%TYPE  )  IS  BEGIN  UPDATE Account SET login = insertlogin where id = p\_id;  UPDATE Account SET password = insertpassword where id = p\_id;  UPDATE Client SET name = insertname where id = p\_id;  UPDATE Client SET surname = insertsurname where id = p\_id;  UPDATE Client SET age = insertage where id = p\_id;  UPDATE Client SET email = insertemail where id = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Client has been updated successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE UpdateCar(  p\_id in car.id%TYPE,  insertbrand IN car.brand%TYPE,  insertmodel IN car.model%TYPE,  insertcolor IN car.color%TYPE,  insertweight IN car.weight%TYPE,  insertcost IN car.cost%TYPE,  insertdescription IN equipment.idEquipmentList%TYPE,  insertcompanyname IN car.idSupplier%TYPE,  insertratingCar IN car.carrating%TYPE  )  IS  BEGIN  UPDATE Car SET brand = insertbrand where id = p\_id;  UPDATE Car SET model = insertmodel where id = p\_id;  UPDATE Car SET color = insertcolor where id = p\_id;  UPDATE Car SET weight = insertweight where id = p\_id;  UPDATE Car SET cost = insertcost where id = p\_id;  UPDATE Equipment SET idEquipmentList = insertdescription where idCar = p\_id;  UPDATE Car SET idSupplier = insertcompanyname where id = p\_id;  UPDATE Car SET carrating = insertratingCar where id = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Car has been updated successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE InsertCar(  insertbrand IN car.brand%TYPE,  insertmodel IN car.model%TYPE,  insertcolor IN car.color%TYPE,  insertweight IN car.weight%TYPE,  insertcost IN car.cost%TYPE,  insertimgsource IN car.imgsource%TYPE,  insertcompanyname IN car.idSupplier%TYPE,  insertratingCar IN car.carrating%TYPE  )  IS  caridNew number;  BEGIN  SELECT COALESCE(MAX(id), 0) + 1 INTO caridNew FROM car;  INSERT INTO car(id, idSupplier, imgSource, brand, model, color, weight, cost, carrating)  VALUES (caridNew, insertcompanyname, insertimgsource, insertbrand, insertmodel, insertcolor, insertweight, insertcost, insertratingCar);    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Car has been inserted successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE DeleteClient(  p\_id IN INT  )  IS  BEGIN  DELETE from Client where id = p\_id;  DELETE from Account where idClient = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('CLient has been deleted successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE DeleteCar(  p\_id IN INT  )  IS  BEGIN  DELETE from Car where id = p\_id;  DELETE from Equipment where idCar = p\_id;    COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Car has been deleted successfully.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error occured while adding a new order. ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE UpdateQuantity(orderId IN INT)  IS  BEGIN  UPDATE admincore.OrderTable SET Quantity = Quantity + 1 WHERE id = orderId;  COMMIT;  END;  CREATE OR REPLACE PROCEDURE UpdateQuantityDecrement(orderId IN INT)  IS  BEGIN  UPDATE admincore.OrderTable SET Quantity = Quantity - 1 WHERE id = orderId;  COMMIT;  END; |

|  |
| --- |
| > sqlplus / as sysdba  SQL> startup  RMAN> connect target /  RMAN> alter system set db\_recovery\_file\_dest\_size=4096M scope=both;  RMAN> alter system set db\_recovery\_file\_dest=' C:\BackupBB\' scope=both;  SQL> SELECT log\_mode FROM v$database;  SQL> SHUTDOWN IMMEDIATE; SQL> STARTUP MOUNT;  SQL> ALTER DATABASE ARCHIVELOG;  SQL> ALTER DATABASE OPEN;  SQL> ALTER SYSTEM SWITCH LOGFILE;  RMAN> backup as compressed backupset database plus archivelog;  Удаление файлов dbf  > rman  RMAN> connect target /  RMAN> shutdown abort  RMAN> startup mount  RMAN> restore database;  RMAN> recover database;  RMAN> alter database open; |





