МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных «Магазин автомобилей» с применением резервного копирования и восстановления данных»

Выполнил студент Денисюк И. С.

(Ф.И.О.)

Руководитель проекта ассистент Колмаков М.В.

(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 2021

**Оглавление**

[Введение 3](#_Toc71921725)

[1 Постановка задачи 4](#_Toc71921726)

[2 Разработка модели базы данных 5](#_Toc71921727)

[3 Разработка необходимых объектов 6](#_Toc71921728)

[3.1 Таблицы 6](#_Toc71921729)

[3.2 Пользователи 9](#_Toc71921730)

[3.3 Хранимые процедуры 9](#_Toc71921731)

[4 Описание процедур импорта и экспорта 10](#_Toc71921732)

[5 Тестирование производительности 11](#_Toc71921733)

[6 Описание технологии и ее применения в базе данных 13](#_Toc71921734)

[7 Руководство пользователя 15](#_Toc71921735)

[Заключение 21](#_Toc71921736)

[Cписок источников 22](#_Toc71921737)

[Приложение А 23](#_Toc71921738)

[Приложение Б 25](#_Toc71921739)

[Приложение В 27](#_Toc71921740)

[Приложение Г 35](#_Toc71921741)

## **Введение**

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, в структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

За последние несколько лет наблюдается тенденция к усложнению структур данных. Простые виды информации, представимой в форме чисел и текстовых строк, не утратив своей значимости, дополняются сегодня многочисленными мультимедийными документами, графическими образами, хронологическими рядами, процедурными, или активными, данными и мириадами прочих сложных информационных форм.

На сегодняшний день на рынке представлено множество технологий доступа к данным и серверов баз данных, каждое, из которых имеет свои отличительные черты. Современные приложения обработки данных ориентированы на работу с большим количеством пользователей, на их удаленность от места расположения основного сервера БД.

Темой данного курсового проекта является разработка программы «Магазин автомобилей».

Приложение «Магазин автомобилей» имеет доступ к таблицам в базе данных и предназначено для взаимодействия с источником данных. Взаимодействие подразумевает получение данных, их представление в определенном формате для просмотра пользователем, редактирование в соответствии с реализованными в программе бизнес-алгоритмами и возврат обработанных данных обратно в базу данных.

База данных «Магазин автомобилей» существенно упрощает работу сотрудников магазина автомобилей, а также предоставляет возможность администратору своевременно вносить необходимые изменения.

# **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:

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

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 12c.

# **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, спроектированной в ходе разработки показана на рисунке 2.1.

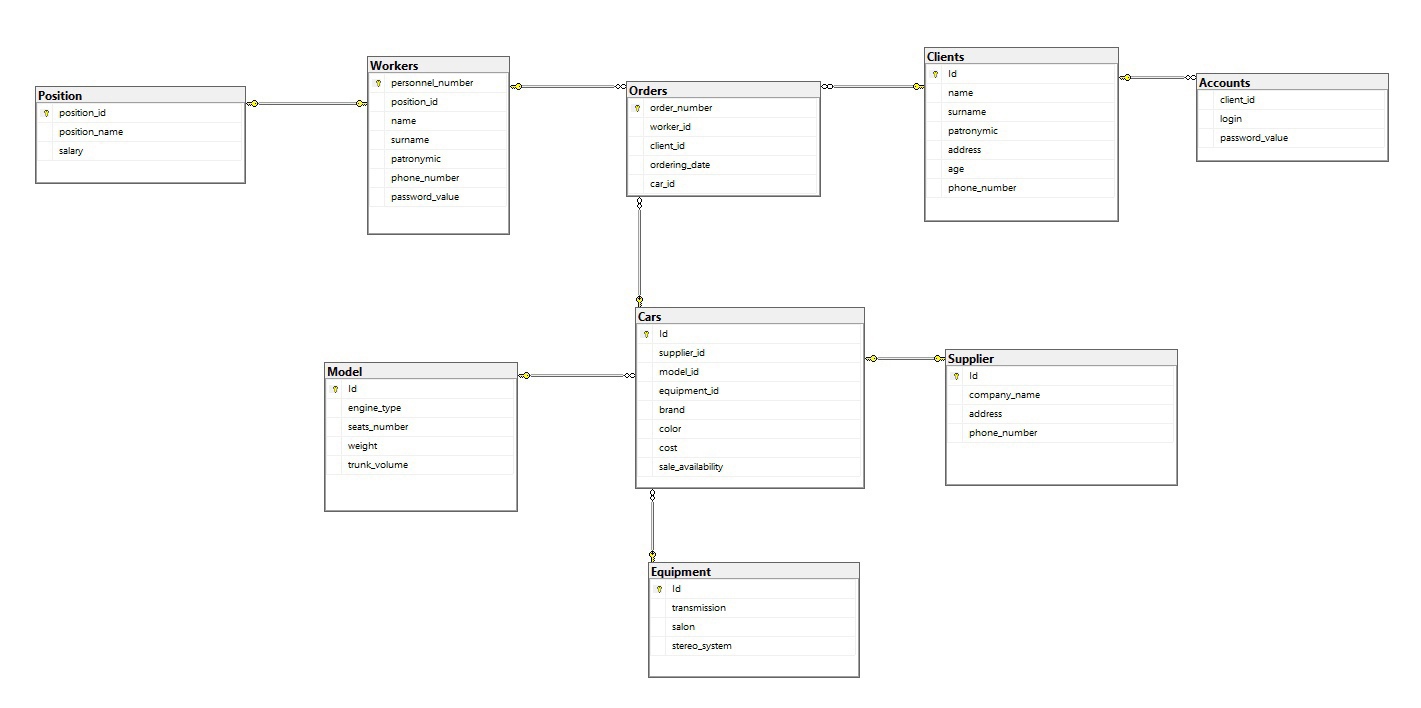


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 9 таблиц, взаимосвязанных между собой внешними ключами:

* Cars;
* Equipment;
* Supplier;
* Model;
* Orders;
* Clients;
* Workers;
* Accounts;
* Position.

# **Разработка необходимых объектов**

# **Таблицы**

Для реализации базы данных «Магазин автомобилей» было разработано 9 таблиц: Cars, Equipment, Supplier, Model, Orders, Clients, Workers, Accounts, Position.

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

Таблица 3.1 – Столбцы таблицы Clients

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор клиента, первичный ключ | int |
| surname | фамилия клиента | nvarchar2 |
| name | имя клиента | nvarchar2 |
| patronymic | отчество клиента | nvarchar2 |
| address | адрес | nvarchar2 |
| age | возраст | int |
| phone number | номер телефона | nvarchar2 |

Таблица Workers представляет список сотрудников состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы Workers

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| personnel number | идентификатор сотрудника, первичный ключ | int |
| surname | фамилия сотрудника, тип | nvarchar2 |
| name | имя сотрудника, тип | nvarchar2 |
| patronymic | отчество сотрудника, тип | nvarchar2 |
| phone number | номер телефона, тип | nvarchar2 |
| position id | идентификатор должности, внешний ключ для связи с таблицей position | int |
| password value | пароль сотрудника | nvarchar2 |

Таблица Accounts представляет список учётных записей клиентов в системе, таблица состоит из столбцов (таблица 3.3):

Таблица 3.3 – Столбцы таблицы Accounts

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| login | логин пользователя | nvarchar2 |
| password | пароль пользователя | nvarchar2 |
| client id | идентификатор клиента, внешний ключ | int |

Таблица Position представляет список должностей. В данной таблице отражаются две возможные должности, которые и будут проверяться в программе. Таблица состоит из столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы Position

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| position id | идентификатор должности, первичный ключ | int |
| salary | зарплата сотрудника | decimal |
| position name | название должности | nvarchar2 |

Таблица Supplier представляет список поставщиков автомобилей, состоит из столбцов (таблица 3.5):

Таблица 3.5 – Столбцы таблицы Supplier

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор поставщика, первичный ключ | int |
| company name | название компании | nvarchar2 |
| address | адрес поставщика | number |
| phone number | номер телефона поставщика | nvarchar2 |

Таблица Model представляет список моделей автомобилей, состоит из столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы Model

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор модели, первичный ключ | int |
| engine type | тип двигателя | nvarchar2 |
| seats number | количество мест | int |
| trunk volume | вместимость багажника | int |
| weight | вес автомобиля | number |

Таблица Orders представляет заказы клиента, состоит из столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы Orders

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| order number | номер заказа, тип, первичный ключ | int |
| worker id | номер сотрудника, внешний ключ для связи с таблицей workers | int |

Продолжение таблицы 3.7

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| client id | идентификатор клиента, тип, внешний ключ для связи с таблицей clients | int |
| ordering date | дата заказа | date |
| car id | идентификатор автомобиля, внешний ключ для связи с таблицей cars | int |
| client id | идентификатор клиента, тип, внешний ключ для связи с таблицей clients | int |
| ordering date | дата заказа | date |

Таблица Cars представляет список автомобилей, которые будет искать и заказывать пользователь через разработанное приложение. Данная таблица состоит из столбцов (таблица 3.8):

Таблица 3.8 – Столбцы таблицы Cars

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор автомобиля, тип, первичный ключ | int |
| supplier id | идентификатор поставщика, тип, внешний ключ для связи с таблицей Supplier | int |
| model id | идентификатор модели, тип, внешний ключ для связи с таблицей Models | int |
| equipment id | идентификатор оснащения, тип int, внешний ключ для связи с таблицей Equipment | int |
| brand | марка автомобиля | nvarchar2 |
| color | дата выселения | date |
| cost | цена автомобиля | int |
| sale availability | наличие для продажи | int |

Таблица Equipment представляет оборудование автомобиля, которое идёт в комплекте с его покупкой при заказе, поэтому связана с таблицей Car состоит из столбцов (таблица 3.9):

Таблица 3.9 – Столбцы таблицы Equipment

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор оснащения | int |

Продолжение таблицы 3.9

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| transmission | коробка передач | nvarchar2 |
| salon | тип автомобиля | nvarchar2 |
| stereo system | стереосистема | nvarchar2 |

# **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 было использовано 2 пользователя. Первый пользователь – сотрудник, который является системным администратором, имеет доступ для чтения и изменения таблиц, связанных с клиентами.

Второй пользователь – клиент – имеет доступ к таблице с автомобилями и может оформить заказ автомобиля.

# **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 было создано 37 процедур для следующих целей:

* выборка данных для таблиц;
* добавление/удаление сотрудника;
* создание учетной записи, для увеличения безопасности учетной записи пользователя его пароль находится в зашифрованном виде.
* авторизация пользователей;
* регистрация пользователей;
* добавление клиента;
* добавления заказа;
* обновление информации о клиенте;
* добавления автомобиля;
* экспорт и импорт таблицы в формат xml.

# **Описание процедур импорта и экспорта**

В курсовом проекте был проведен импорт данных из XML файлов и экспорт данных в формат XML.

XML — это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle, в данной курсовой работе используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат. Для работы с файлами так же использовался пакет DBMS\_LOB. Данные пакеты были выбраны, потому что они обладают очень широким функционалом и гибкой настройкой xml документа. В данном курсовом проекте функции экспорта, импорта используются для таблицы EQUIPMENT, так как эта таблица является одной из самых важных в базе данных. Пример создания процедуры экспорта можно посмотреть на рисунке 4.1.



Рисунок 4.1 – Пример создания процедуры ExportEQUIPMENTtoXml

Для сохранения и управления XML-данными в реляционной таблице применяется специальный тип данных XMLType. Тип данных XMLType поставляется с набором специальных XML-методов, которые можно использовать для работы с объектами XMLType. Эти методы можно применять для выполнения как типичных операций в базе данных, наподобие проверки на предмет существования узла или извлечения узла, так и нескольких специальных операций, позволяющих получать доступ к XML-данными и манипулировать ими в виде части обычного SQL-оператора.

# **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 5.1).

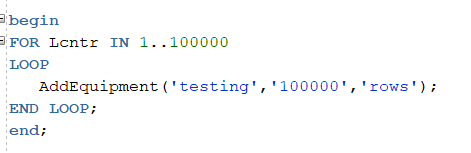


Рисунок 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Таким образом было добавлено 100000 строк в таблицу.

После этого в другом анонимном блоке, который содержит в себе дополнительно переменную для отслеживания времени выполнения, был сделан запрос, который должен возвращать в результате 20000 строк. Данный блок показан на рисунке 5.2.

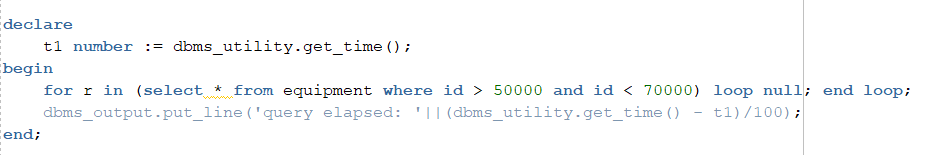


Рисунок 5.2 – Блок отбора данных

Далее после выполнения данного блока наша таблица будет содержать большое количество данных, и мы можем проанализировать время выполнения запроса. Также можно просмотреть план запроса используя стандартные средства Oracle, а именно кнопку на главной панели, предварительно выделив данный запрос. Покажем результат, в котором будет заметна разница во времени выполнениея запроса до создания индекса на рисунке 5.3.

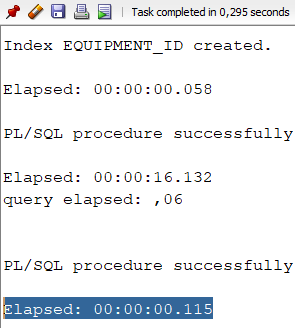


Рисунок 5.3 – Результат отбора данных

Теперь остаётся лишь создать индекс и протестировать время выполения аналогичного запроса. Покажем это на рисунке 5.4.

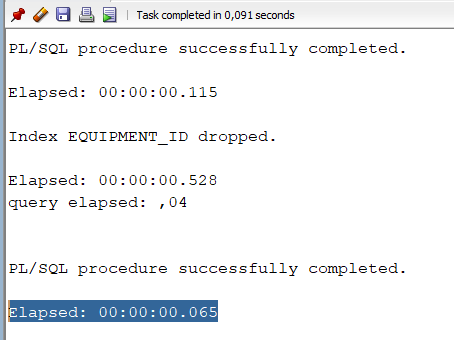


Рисунок 5.4 – Результат после создания индекса

Как видим, время запроса уменьшилось, но тем не менее даже без индекса наша база данных успешно прошла тест на производительность.

# **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 6.1.

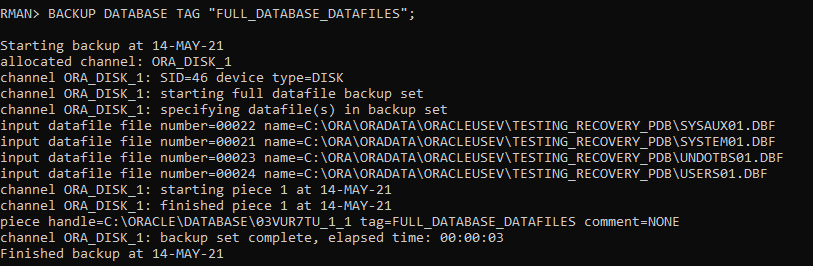


Рисунок 6.1 – Резервирование файлов данных

Для резервирования контрольных файлов используется команда: BACKUP CURRENT CONTROLFILE TAG "FULL\_DATABASE\_CONTROLFILE". Результат можно посмотреть на рисунке 6.2.

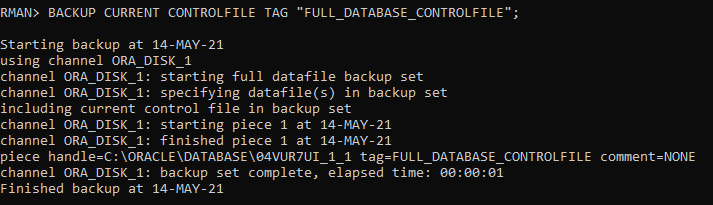


Рисунок 6.2 – Резервирование управляющих файлов

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).

# **Руководство пользователя**

Перед запуском приложения изначально его необходимо установить. Для это нужно запустить файл CarShopInstaller.exe и выбрать язык и пути для программы. После успешной установки можно переходить к запуску.

При запуске приложения открывается окно авторизации, которое является единым как для администратора, так и для клиента (рисунок.7.1).

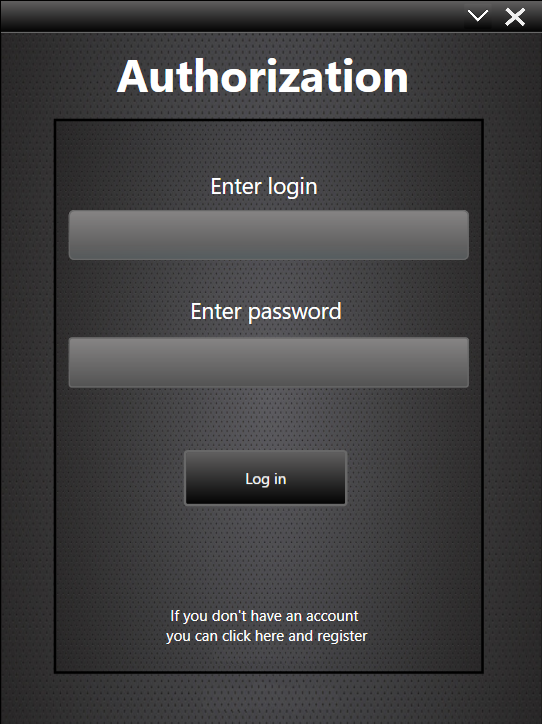


Рисунок 7.1 – Окно авторизации

Вход в систему можно выполнить от 2 типов пользователей: сотрудник, который является системным администратором и клиент.

Не все сотрудники могут входить в систему, это зависит от их должности в магазине. Клиенту для авторизации необходимо ввести свой логин и пароль, предварительно зарегистрировавшись в системе. А системному администратору нужно ввести свой номер работника. Клиент не может зарегистрироваться таким образом, чтобы его логин начинался с цифры либо состоял только их одних цифр.

Само приложение предусматривает обработку некорректного ввода для абсолютно любого поля. К примеру, если ввести неправильный логин или пароль, то пользователь либо системный администратор будет оповещён соответствующим образом (рисунок 7.2).

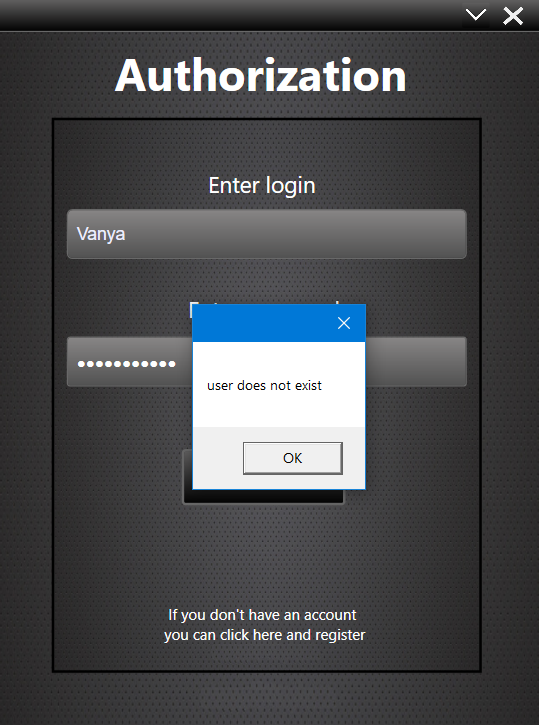


Рисунок 7.2 – Оповещение о некорректном логине или пароле

Аналогичным будет окно с оповещением о неправильном вводе и для администратора, разве что отличаться будет только сам текст оповещения.

После этого требуется повторно ввести данные пользователя, а системному администратору свои корректные значения для входа, ведь ему не обязательно проходить регистрацию, и администратор должен помнить только свой рабочий номер и соответствующий пароль.

Далее продемонстрируем работу системного администратора в нашем приложении. Для этого войдём в систему под одним из сотрудников магазина, у которого есть разрешение на пользование приложением, и проверим, что из этого выйдет. На рисунке 7.3 показан результат входа в систему в качестве администратора.



Рисунок 7.3 – Окно администратора

Администратор может просматривать список клиентов, а также некоторую информацию о них. В добавок у него предусмотрено обновление пользователей, поэтому администратор может выделить любого клиента из тех, что зарегистрировались в системе и затем нажать на кнопку «Update user» после которой откроется окно обновления пользователя. На рисунке 7.4 показано окно обновления.

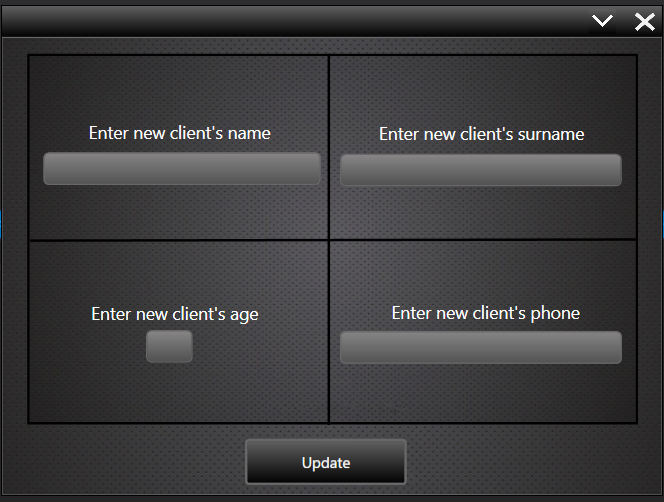


Рисунок 7.4 – Окно обновления пользователя

После ввода новых данных клиента системному администратору необходимо нажать на кнопку «Update» для успешного обновления пользователя.

Перейдём к работе непосредственно с клиентами. Если клиент до этого не существовал в нашей системе, ему необходимо зарегистрироваться. Нажав для этого в окне авторизации на соответствующую кнопку внизу. После этого откроется окно регистрации, показанное на рисунке 7.5.

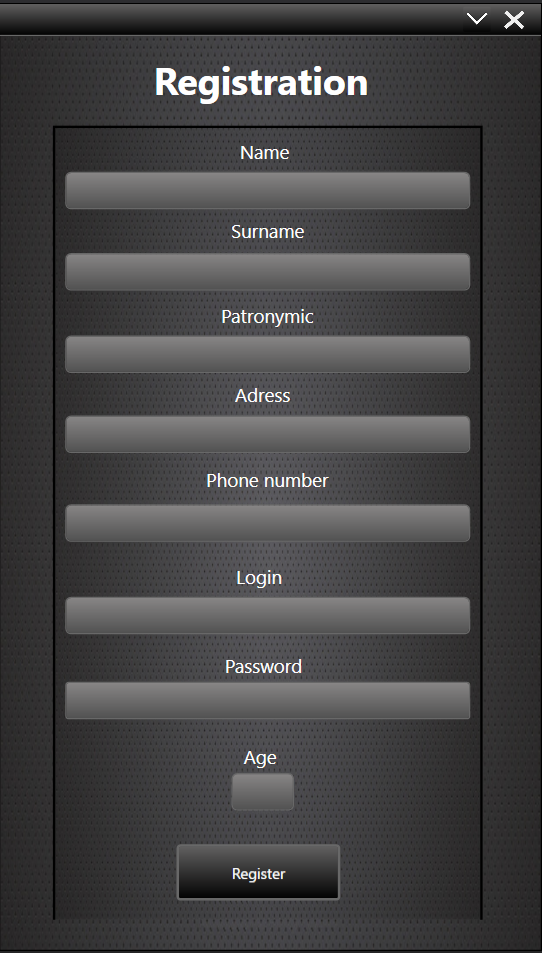


Рисунок 7.5 – Окно регистрации клиента

Далее клиенту необходимо корректно ввести такие обязательные поля как имя, фамилия, отчество, адрес, мобильный телефон, возраст и логин вместе с паролем для последующей авторизации. При ошибочном или пустом поле будет показано соответствующее сообщение.

После заполнения всех полей и нажатия на кнопку «Register» снова будет открыто окно авторизации.

Если клиент войдёт в свой аккаунт с корректными данными, то перед ним откроется окно с автомобилями, показанное на рисунке 7.6.



Рисунок 7.6 – Окно с автомобилями

Из данного списка автомобилей клиент может выбрать любой, какой захочет, затем нажать на кнопку «Order» и заказать себе соответствующую машину. Заказ автомобиля продемонстрирован на рисунке 7.7.

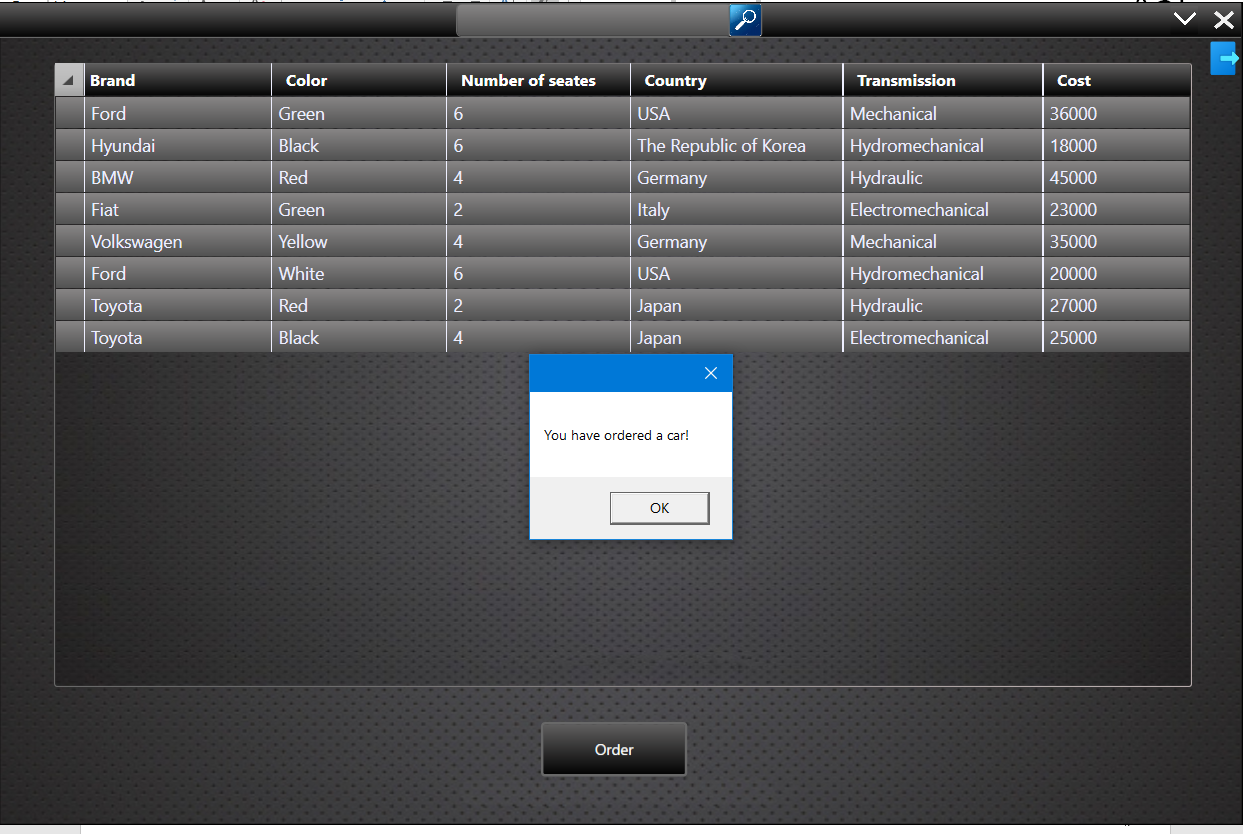


Рисунок 7.7 – Заказ автомобиля

Также на данном окне присутствует поиск в виде строки ввода и кнопки, который работает по любому полю из таблицы автомобилей, поэтому можно осуществить поиск по наименованию бренда, цвета, количеству сидений, поставщику и другим полям, входящим в таблицу. Покажем на рисунке 7.8 поиск автомобилей, в которых количество посадочных мест будет равно именно двум, чтобы продемонстрировать поиск

.

Рисунок 7.8 – Поиск автомобилей

Исходя из проделанной работы мы можем отметить, что данное разработанное приложение правильно выполняет свои функции такие как регистрация клиента, авторизация клиента, защита от некорректных данных, которые вводятся пользователем, шифрование нежелательных для утечки данных. 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 «Магазин автомобилей», которая в совокупности с приложением формирует полноценное desktop-приложение для управления магазином. В данной работе использовалось СУБД Oracle DataBase 12c, 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, функции, последовательности, триггеры.

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 ADO.NET. 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;
* шифрование данных зарегистрированного пользователя.

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». Шифрование данных было реализовано с помощью пакета DBMS\_CRYPTO.

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] – http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php. – Дата доступа: 09.04.2021.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – https://metanit.com/sharp/adonet/. – Дата доступа: 08.04.2021.
3. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm – Дата доступа: 09.04.2021.
4. Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html– Дата доступа: 11.04.2021.
5. Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678 – Дата доступа: 17.04.2021.

# **Приложение А**

//Создание таблиц и последовательностей

CREATE sequence CARS\_ID\_SEQ;

CREATE TABLE Cars (

Id INT DEFAULT CARS\_ID\_SEQ.NEXTVAL NOT NULL,

supplier\_id INT NOT NULL,

model\_id INT NOT NULL,

equipment\_id INT NOT NULL,

brand NVARCHAR2(50) NOT NULL,

color NVARCHAR2(50) NOT NULL,

cost DECIMAL(10) NOT NULL,

sale\_availability NUMBER(1) NOT NULL,

constraint CAR\_PK PRIMARY KEY (Id));

CREATE sequence EQUIPMENT\_ID\_SEQ;

CREATE TABLE Equipment (

Id INT DEFAULT EQUIPMENT\_ID\_SEQ.NEXTVAL NOT NULL,

transmission NVARCHAR2(60) NOT NULL,

salon NVARCHAR2(60) NOT NULL,

stereo\_system NVARCHAR2(60) NOT NULL,

constraint EQUIPMENT\_PK PRIMARY KEY (Id));

CREATE sequence SUPPLIER\_ID\_SEQ;

CREATE TABLE Supplier (

Id INT DEFAULT SUPPLIER\_ID\_SEQ.NEXTVAL NOT NULL,

сompany\_name NVARCHAR2(200) NOT NULL,

adress NVARCHAR2(200) NOT NULL,

phone\_number NVARCHAR2(50) NOT NULL,

constraint SUPPLIER\_PK PRIMARY KEY (Id));

CREATE sequence MODEL\_ID\_SEQ;

CREATE TABLE Model (

Id INT DEFAULT MODEL\_ID\_SEQ.NEXTVAL NOT NULL,

engine\_type NVARCHAR2(50) NOT NULL,

seats\_number INT NOT NULL,

weight NUMBER(6) NOT NULL,

trunk\_volume INT NOT NULL,

constraint MODEL\_PK PRIMARY KEY (Id));

CREATE sequence ORDERS\_ORDERS\_NUMBER\_SEQ;

CREATE TABLE Orders (

order\_number INT DEFAULT ORDERS\_ORDERS\_NUMBER\_SEQ.NEXTVAL NOT NULL,

worker\_id INT NOT NULL,

client\_id INT NOT NULL,

ordering\_date DATE NOT NULL,

car\_id INT UNIQUE NOT NULL,

constraint ORDER\_PK PRIMARY KEY (order\_number));

CREATE sequence CLIENTS\_ID\_SEQ;

CREATE TABLE Clients (

Id INT DEFAULT CLIENTS\_ID\_SEQ.NEXTVAL NOT NULL,

name NVARCHAR2(50) NOT NULL,

surname NVARCHAR2(50) NOT NULL,

patronymic NVARCHAR2(50) NOT NULL,

adress NVARCHAR2(100) NOT NULL,

age INT NOT NULL,

phone\_number NVARCHAR2(100) UNIQUE NOT NULL,

constraint CLIENT\_PK PRIMARY KEY (id));

CREATE TABLE Workers (

personnel\_number INT NOT NULL,

position\_id INT NOT NULL,

name NVARCHAR2(50) NOT NULL,

surname NVARCHAR2(50) NOT NULL,

patronymic NVARCHAR2(50) NOT NULL,

phone\_number NVARCHAR2(50) UNIQUE NOT NULL,

password\_value NVARCHAR2(50) NOT NULL,

constraint WORKER\_PK PRIMARY KEY (personnel\_number));

CREATE sequence POSITION\_POSITION\_ID\_SEQ;

CREATE TABLE Position (

position\_id INT DEFAULT POSITION\_POSITION\_ID\_SEQ.NEXTVAL NOT NULL,

position\_name NVARCHAR2(50) UNIQUE NOT NULL,

salary DECIMAL(10) NOT NULL,

constraint POSITION\_PK PRIMARY KEY (position\_id));

CREATE TABLE Accounts (

client\_id INT NOT NULL,

login NVARCHAR2(100) UNIQUE NOT NULL,

password\_value NVARCHAR2(100) NOT NULL);

ALTER TABLE Cars ADD CONSTRAINT Cars\_fk1 FOREIGN KEY (supplier\_id) REFERENCES Supplier(Id);

ALTER TABLE Cars ADD CONSTRAINT Cars\_fk2 FOREIGN KEY (model\_id) REFERENCES Model(Id);

ALTER TABLE Cars ADD CONSTRAINT Cars\_fk3 FOREIGN KEY (equipment\_id) REFERENCES Equipment(Id);

ALTER TABLE Orders ADD CONSTRAINT Orders\_fk1 FOREIGN KEY (worker\_id) REFERENCES Workers(personnel\_number);

ALTER TABLE Orders ADD CONSTRAINT Orders\_fk2 FOREIGN KEY (client\_id) REFERENCES Clients(id);

ALTER TABLE Orders ADD CONSTRAINT Orders\_fk3 FOREIGN KEY (car\_id) REFERENCES Cars(Id);

ALTER TABLE Workers ADD CONSTRAINT Workers\_fk FOREIGN KEY (position\_id) REFERENCES Position(position\_id);

ALTER TABLE Accounts ADD CONSTRAINT Accounts\_fk FOREIGN KEY (client\_id) REFERENCES Clients(id);

# **Приложение Б**

//XML EXPORT/IMPORT

create or replace procedure ExportEQUIPMENTtoXml

is

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("EQUIPMENTS",

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

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

XMLAGG(XMLELEMENT("EQUIPMENT",

xmlelement("id", EQUIPMENT.id),

xmlelement("transmission", EQUIPMENT.transmission),

xmlelement("salon", EQUIPMENT.salon),

xmlelement("stereo\_system", EQUIPMENT.stereo\_system)

))) from EQUIPMENT;

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/EQUIPMENTS.xml');

END;

create or replace procedure ImportEQUIPMENTFromXml

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('UTLDATA', 'EQUIPMENTS.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;

et EQUIPMENT%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;

P := Dbms\_Xmlparser.Newparser;

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

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

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

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

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

LOOP

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

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

'id/text()',et.id);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,

'transmission/text()',et.transmission);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,

'salon/text()',et.salon);

dbms\_xslprocessor.valueof(v\_current\_node,

'stereo\_system/text()',et.stereo\_system);

insert into EQUIPMENT(transmission, salon, stereo\_system)

values(et.transmission, et.salon, et.stereo\_system);

end loop;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

commit;

END;

# **Приложение В**

//Процедуры

create or replace procedure AddCar(p\_supplier\_id cars.supplier\_id%type, p\_model\_id cars.model\_id%type, p\_equipment\_id cars.equipment\_id%type, p\_brand cars.brand%type, p\_color cars.color%type, p\_cost cars.cost%type, p\_sale\_availability cars.sale\_availability%type)

is

begin

insert into cars(supplier\_id, model\_id, equipment\_id, brand, color, cost ,sale\_availability)

VALUES(p\_supplier\_id, p\_model\_id, p\_equipment\_id, p\_brand, p\_color, p\_cost, p\_sale\_availability);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteCar(p\_id cars.id%type)

is

begin

delete from cars where cars.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateCar(p\_id cars.id%type, p\_supplier\_id cars.supplier\_id%type, p\_model\_id cars.model\_id%type, p\_equipment\_id cars.equipment\_id%type, p\_brand cars.brand%type, p\_color cars.color%type, p\_cost cars.cost%type, p\_sale\_availability cars.sale\_availability%type)

is

begin

update cars set id = p\_id, supplier\_id = p\_supplier\_id, model\_id = p\_model\_id, equipment\_id = p\_equipment\_id, brand = p\_brand, color = p\_color, cost = p\_cost, sale\_availability = p\_sale\_availability where cars.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddEquipment(p\_transmission equipment.transmission%type, p\_salon equipment.salon%type, p\_stereo\_system equipment.stereo\_system%type)

is

begin

insert into equipment(transmission, salon, stereo\_system)

VALUES(p\_transmission, p\_salon, p\_stereo\_system);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteEquipment(p\_id equipment.id%type)

is

begin

delete from equipment where equipment.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateEquipment(p\_id equipment.id%type, p\_transmission equipment.transmission%type, p\_salon equipment.salon%type, p\_stereo\_system equipment.stereo\_system%type)

is

begin

update equipment set id = p\_id, transmission = p\_transmission, salon = p\_salon, stereo\_system = p\_stereo\_system where equipment.id = p\_id ;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

create or replace procedure AddSupplier(p\_сompany\_name supplier.сompany\_name%type, p\_adress supplier.adress%type, p\_phone\_number supplier.phone\_number%type)

is

begin

insert into supplier(сompany\_name, adress, phone\_number)

VALUES(p\_сompany\_name, p\_adress, p\_phone\_number);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteSupplier(p\_id supplier.id%type)

is

begin

delete from supplier where supplier.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateSupplier(p\_id supplier.id%type, p\_сompany\_name supplier.сompany\_name%type, p\_adress supplier.adress%type, p\_phone\_number supplier.phone\_number%type)

is

begin

update supplier set id = p\_id, сompany\_name = p\_сompany\_name, adress = p\_adress, phone\_number = p\_phone\_number where supplier.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddModel(p\_engine\_type model.engine\_type%type, p\_seats\_number model.seats\_number%type, p\_weight model.weight%type, p\_trunk\_volume model.trunk\_volume%type)

is

begin

insert into model(engine\_type, seats\_number, weight, trunk\_volume)

VALUES(p\_engine\_type, p\_seats\_number, p\_weight, p\_trunk\_volume);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteModel(p\_id model.id%type)

is

begin

delete from Model where Model.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateModel(p\_id model.id%type, p\_engine\_type model.engine\_type%type, p\_seats\_number model.seats\_number%type, p\_weight model.weight%type, p\_trunk\_volume model.trunk\_volume%type)

is

begin

update model set id = p\_id, engine\_type = p\_engine\_type, seats\_number = p\_seats\_number, weight = p\_weight, trunk\_volume = p\_trunk\_volume where model.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddOrder(p\_worker\_id orders.worker\_id%type, p\_client\_id orders.client\_id%type, p\_ordering\_date orders.ordering\_date%type,p\_car\_id orders.car\_id%type)

is

begin

insert into orders(worker\_id, client\_id, ordering\_date, car\_id)

VALUES(p\_worker\_id, p\_client\_id, p\_ordering\_date, p\_car\_id);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteOrder(p\_id orders.order\_number%type)

is

begin

delete from orders where orders.order\_number = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateOrder(p\_id orders.order\_number%type, p\_worker\_id orders.worker\_id%type, p\_client\_id orders.client\_id%type, p\_ordering\_date orders.ordering\_date%type)

is

begin

update orders set order\_number = p\_id, worker\_id = p\_worker\_id, client\_id = p\_client\_id, ordering\_date = p\_ordering\_date where orders.order\_number = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddClient(p\_name clients.name%type, p\_surname clients.surname%type, p\_patronymic clients.patronymic%type,p\_adress clients.adress%type,p\_age clients.age%type,p\_phone\_number clients.phone\_number%type, new\_id out clients.id%type)

is

p\_adress\_encrypted clients.adress%type;

p\_phone\_number\_encrypted clients.phone\_number%type;

begin

p\_adress\_encrypted := encode(p\_adress, 'superkey12345678');

p\_phone\_number\_encrypted := encode(p\_phone\_number, 'superkey12345678');

insert into clients(name, surname ,patronymic, adress, age, phone\_number)

VALUES(p\_name, p\_surname, p\_patronymic, p\_adress\_encrypted, p\_age, p\_phone\_number\_encrypted);

new\_id := CLIENTS\_ID\_SEQ.CURRVAL;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteClient(p\_id clients.id%type)

is

begin

delete from clients where clients.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateClient(p\_id clients.id%type, p\_name clients.name%type, p\_surname clients.surname%type, p\_age clients.age%type, p\_phone\_number clients.phone\_number%type)

is

p\_phone\_number\_encrypted clients.phone\_number%type;

begin

p\_phone\_number\_encrypted := encode(p\_phone\_number, 'superkey12345678');

update clients set name = p\_name, surname = p\_surname, age= p\_age, phone\_number= p\_phone\_number\_encrypted where clients.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddWorker(p\_personnel\_number workers.personnel\_number%type, p\_position\_id workers.position\_id%type, p\_name workers.name%type,p\_surname workers.surname%type,p\_patronymic workers.patronymic%type,p\_phone\_number workers.phone\_number%type, p\_password\_value workers.password\_value%type)

is

begin

insert into workers(personnel\_number, position\_id ,name, surname, patronymic, phone\_number, password\_value)

VALUES(p\_personnel\_number, p\_position\_id, p\_name, p\_surname, p\_patronymic, p\_phone\_number, p\_password\_value);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteWorker(p\_id workers.personnel\_number%type)

is

begin

delete from workers where workers.personnel\_number = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateWorker(p\_personnel\_number workers.personnel\_number%type, p\_position\_id workers.position\_id%type, p\_name workers.name%type,p\_surname workers.surname%type,p\_patronymic workers.patronymic%type,p\_phone\_number workers.phone\_number%type, p\_password\_value workers.password\_value%type)

is

begin

update workers set personnel\_number = p\_personnel\_number, position\_id = p\_position\_id, name = p\_name, surname = p\_surname, patronymic= p\_patronymic, phone\_number= p\_phone\_number, password\_value= p\_password\_value where workers.personnel\_number = p\_personnel\_number;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddPosition(p\_position\_name position.position\_name%type, p\_salary position.salary%type)

is

begin

insert into position(position\_name, salary)

VALUES(p\_position\_name, p\_salary);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeletePosition(p\_id position.position\_id%type)

is

begin

delete from position where position.position\_id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdatePosition(p\_id position.position\_id%type, p\_position\_name position.position\_name%type, p\_salary position.salary%type)

is

begin

update position set position\_id = p\_id, position\_name = p\_position\_name, salary = p\_salary where position.position\_id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure AddAccout(p\_client\_id Accounts.client\_id%type, p\_login Accounts.login%type, p\_password\_value Accounts.password\_value%type)

is

p\_password\_value\_encrypted Accounts.password\_value%type;

begin

p\_password\_value\_encrypted := encode(p\_password\_value, 'superkey12345678');

insert into Accounts(client\_id, login, password\_value)

VALUES(p\_client\_id, p\_login,p\_password\_value\_encrypted);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure DeleteAccount(p\_client\_id Accounts.client\_id%type)

is

begin

delete from Accounts where Accounts.client\_id = p\_client\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure UpdateAccount(p\_client\_id Accounts.client\_id%type, p\_login Accounts.login%type, p\_password\_value Accounts.password\_value%type)

is

begin

update Accounts set client\_id = p\_client\_id, login = p\_login, password\_value = p\_password\_value where Accounts.client\_id = p\_client\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure GetCurrentUser(p\_login Accounts.login%type, p\_password\_value Accounts.password\_value%type, cur out sys\_refcursor)

is

p\_password\_value\_encrypted Accounts.password\_value%type;

begin

p\_password\_value\_encrypted := encode(p\_password\_value, 'superkey12345678');

open cur for select \* from Accounts where login = p\_login and password\_value = p\_password\_value\_encrypted;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

select \* from workers;

create or replace procedure GetCurrentWorker(p\_personnel\_number Workers.personnel\_number%type, p\_password\_value Workers.password\_value%type, cur out sys\_refcursor)

is

begin

open cur for select \* from Workers where personnel\_number = p\_personnel\_number and password\_value = p\_password\_value and position\_id = 1 ;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure GetCars(cur out sys\_refcursor)

is

begin

open cur for select cars.id, cars.brand, cars.color, model.seats\_number, Equipment.transmission, supplier.adress, cars.cost from model join cars on model.id = cars.model\_id join equipment on cars.equipment\_id = equipment.id join supplier on cars.supplier\_id =supplier.id where cars.sale\_availability = 1;

dbms\_output.enable();

dbms\_sql.return\_result(cur);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure GetClients(cur out sys\_refcursor)

is

begin

open cur for select accounts.login, accounts.client\_id, clients.name, clients.surname, clients.age, clients.phone\_number from accounts join clients on accounts.client\_id = clients.id;

dbms\_output.enable();

dbms\_sql.return\_result(cur);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure GetCarsSearch( super\_param cars.brand%type,cur out sys\_refcursor)

is

begin

open cur for select cars.id, cars.brand, cars.color, model.seats\_number, Equipment.transmission, supplier.adress, cars.cost from model join cars on model.id = cars.model\_id join equipment on cars.equipment\_id = equipment.id join supplier on cars.supplier\_id = supplier.id

where cars.sale\_availability = 1 and cars.brand = super\_param or

cars.sale\_availability = 1 and cars.color = super\_param or

cars.sale\_availability = 1 and TO\_CHAR(model.seats\_number) = super\_param or

cars.sale\_availability = 1 and Equipment.transmission =super\_param or

cars.sale\_availability = 1 and supplier.adress = super\_param or

cars.sale\_availability = 1 and TO\_CHAR(cars.cost) = super\_param;

dbms\_output.enable();

dbms\_sql.return\_result(cur);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure GetSalesmen(cur out sys\_refcursor)

is

begin

open cur for select workers.personnel\_number, position.position\_name from workers join position on workers.position\_id = position.position\_id where position.position\_name ='Salesman' ;

dbms\_output.enable();

dbms\_sql.return\_result(cur);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

/

create or replace procedure SetCarSaleAvailability(p\_car\_id cars.id%type, p\_sale\_availability cars.sale\_availability%type)

is

begin

update cars set cars.sale\_availability = p\_sale\_availability where cars.id = p\_car\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

# **Приложение Г**

//Функции

create or replace FUNCTION encode (p\_in\_val IN VARCHAR2, p\_key IN VARCHAR2)

RETURN VARCHAR2

IS

l\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes128

+ DBMS\_CRYPTO.chain\_cbc

+ DBMS\_CRYPTO.pad\_pkcs5;

l\_enc RAW (2000);

BEGIN

l\_enc :=

DBMS\_CRYPTO.encrypt (utl\_i18n.string\_to\_raw (p\_in\_val, 'AL32UTF8'),

l\_mod,

utl\_i18n.string\_to\_raw (p\_key, 'AL32UTF8')

);

return RAWTOHEX (l\_enc);

END;

/

create or replace FUNCTION decode\_val (p\_in\_val IN VARCHAR2, p\_key IN VARCHAR2)

RETURN VARCHAR2

IS

l\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes128

+ DBMS\_CRYPTO.chain\_cbc

+ DBMS\_CRYPTO.pad\_pkcs5;

l\_dec RAW (2000);

BEGIN

l\_dec :=

DBMS\_CRYPTO.decrypt (HEXTORAW (p\_in\_val),

l\_mod,

utl\_i18n.string\_to\_raw (p\_key, 'AL32UTF8')

);

return utl\_i18n.raw\_to\_char (l\_dec);

END;.Valueof(V\_Current\_Node,

'transmission/text()',et.transmission);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,

'salon/text()',et.salon);

dbms\_xslprocessor.valueof(v\_current\_node,

'stereo\_system/text()',et.stereo\_system);

insert into EQUIPMENT(transmission, salon, stereo\_system)

values(et.transmission, et.salon, et.stereo\_system);

end loop;