МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация база данных для работы склада с применением технологии OLAP»

Выполнил студент Зыков Кирилл Андреевич

(Ф.И.О.)

Руководитель работы асс. Нистюк О.А.

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

И.о. зав. кафедрой ст. преп. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[Введение 3](#_Toc154062136)

[1 Постановка задачи 4](#_Toc154062137)

[1.1 Аналитический обзор аналогов и литературных решений 4](#_Toc154062138)

[1.2 Анализ и разработка функциональных требований 6](#_Toc154062139)

[2 Проектирование базы данных 8](#_Toc154062140)

[3 Разработка необходимых объектов 11](#_Toc154062141)

[3.1 Таблицы 11](#_Toc154062142)

[3.2 Пользователи 13](#_Toc154062143)

[4 Описание процедур импорта и экспорта 15](#_Toc154062144)

[4.1 Процедура импорта данных в БД 15](#_Toc154062145)

[4.2 Процедура экспорта данных из БД 15](#_Toc154062146)

[5 Тестирование производительности 16](#_Toc154062147)

[6 Описание технологии применяемой в БД 18](#_Toc154062148)

[7 Краткое описание приложения 19](#_Toc154062149)

[8 Руководство пользователя 20](#_Toc154062150)

[Заключение 23](#_Toc154062151)

[Список используемых источников 24](#_Toc154062152)

[Приложение А 25](#_Toc154062153)

[Приложение Б 28](#_Toc154062154)



# Введение

В современном мире эффективное управление предприятием невозможно без эффективных инструментов анализа и контроля. Одним из важных компонентов этого процесса является система учета и анализа данных, особенно в области управления складскими запасами. В рамках курсовой работы по предмету баз данных была разработана и реализована база данных, предназначенная для аналитики работы склада.

Складское управление является сложным и ответственным звеном в цепи поставок, требующим постоянного контроля, оптимизации процессов и оперативного принятия решений. Создание эффективной базы данных для аналитики позволяет автоматизировать процессы учета, управления и анализа данных, что, в свою очередь, облегчает принятие стратегических и тактических решений.

Объектом исследования стала база данных, ориентированная на хранение информации о поступлениях и отгрузках товаров, состоянии складских запасов, а также деятельности персонала, вовлеченного в управление складом. Разработка базы данных включает в себя создание структуры данных, определение связей между таблицами, а также реализацию хранимых процедур, обеспечивающих эффективное взаимодействие с данными.

Целью данной работы является создание интегрированной системы, способной обеспечить точный и оперативный анализ ключевых аспектов складской деятельности. В рамках работы также рассматриваются вопросы обеспечения безопасности данных, оптимизации запросов и разработки пользовательского интерфейса для удобного взаимодействия с информацией.

Также необходимо разработать приложение для демонстрации функциональности базы данных и взаимодействия с ней. Приложение было реализовано с использованием фреймворка WPF на языке C#.

Данная база данных не только предоставляет надежное хранилище данных применяя метод хэширования паролей перед их сохранением в базу данных., но и обеспечивает возможность проведения аналитических исследований, а также повышения эффективности управленческих решений в области складского управления.

Основные требования к приложению:

* Реализация ролей администратора, аналитика, заказчика и поставщика;
* добавление и удаление поставок и отгрузок;
* анализ продукции (количество привозов, отгрузок и др.);
* приемы заказов на привозы и отгрузки;
* вывод в файл данных о заказах и анализе;
* взаимодействие с БД при помощи хранимых процедур и функций.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

1. Постановка задачи

Задачи проекта: разработать архитектуру базы данных, создать процедуры, взаимодействие с которыми будет понятно любому пользователю.

Цели приложения состоит в демонстрации возможности базы данных и предоставления удобного инструмента для анализа работы склада.

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере демонстрировать возможности базы данных.

Для того, чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы из той же области.

* 1. Аналитический обзор аналогов и литературных решений

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день одни из основных аналогичных приложений или web–сервисов для контроля и анализа работы это CRM системы.

Веб–приложение «Monday.com» – представляет собой универсальное решение для управления проектами, командами и бизнес–процессами. В отличие от традиционных CRM–систем, Monday.com ориентирована на широкий спектр бизнес–задач и позволяет создавать настраиваемые доски, отслеживая прогресс и взаимодействие команды.

Интерфейс приведён на рисунке 1.1.

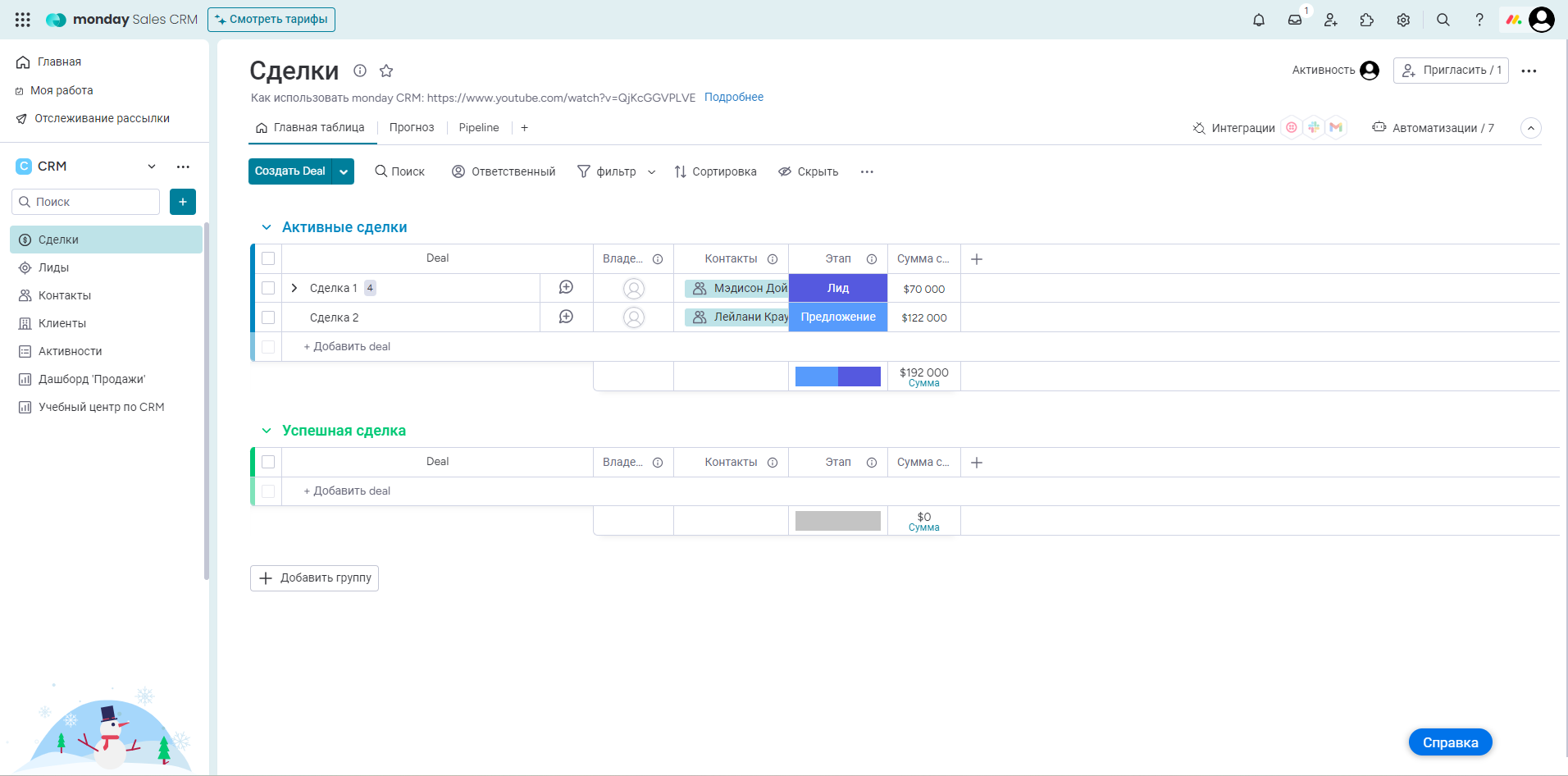


Рисунок 1.1 – Интерфейс веб–приложения «Monday.com»

Проанализировав «Monday.com», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* ­­удобный интерфейс;
* есть учебный центр;
* есть возможность управления таблицами для анализа;
* есть возможность добавления пользователей с правами.

Основные минусы:

* недостаточно функционала;
* платный функционал.

Веб–приложение «Salesforce» – одна из наиболее распространенных и узнаваемых CRM–систем в мире. Она предоставляет широкий спектр инструментов для управления продажами, маркетингом и обслуживанием клиентов. Гибкая конфигурация и облачная архитектура делают Salesforce привлекательным выбором для компаний различного масштаба.

Интерфейс приведён на рисунке 1.2.

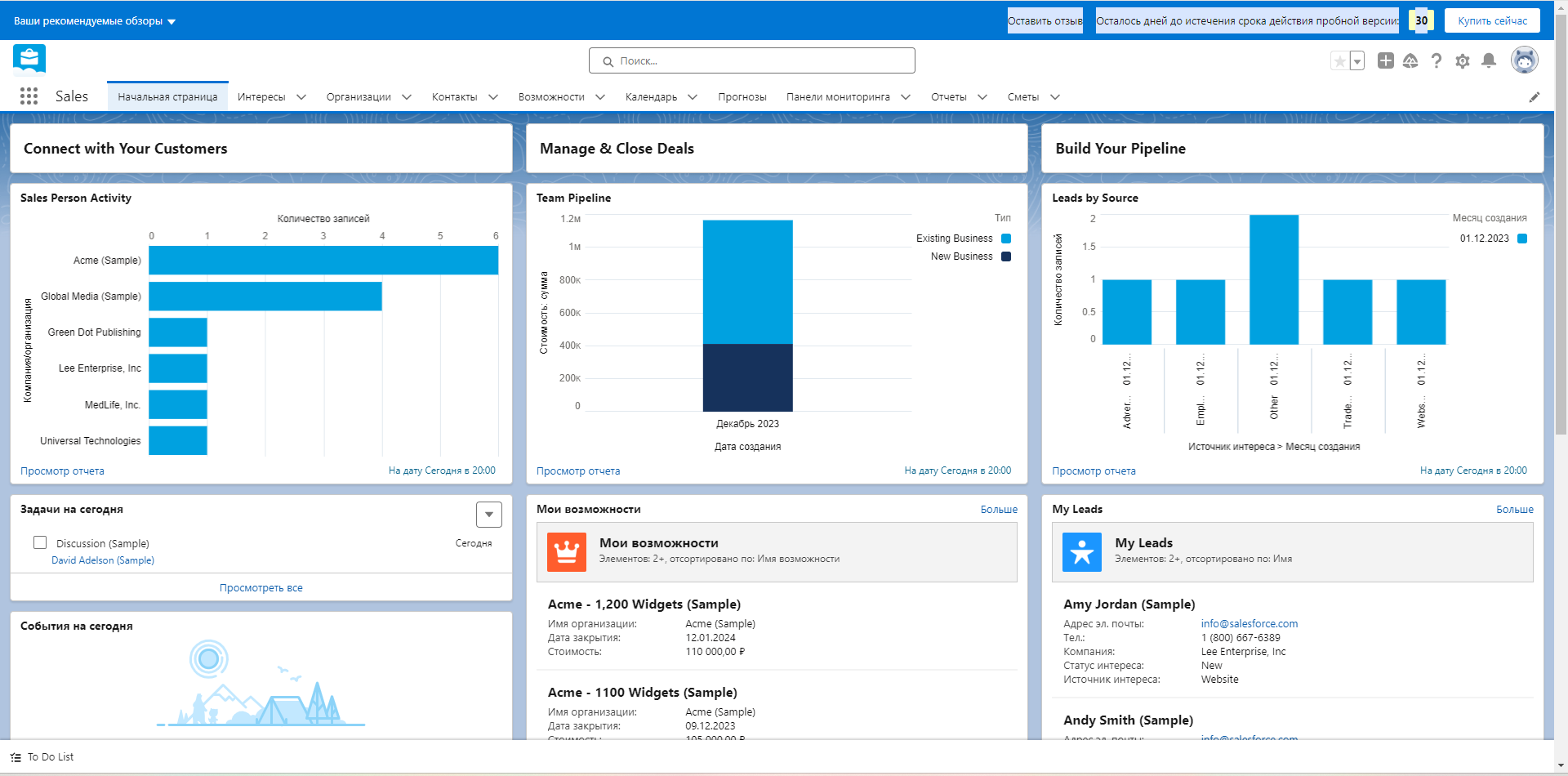


Рисунок 1.2 ­– интерфейс веб–приложения «Salesforce»

Проанализировав «Salesforce», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* есть учебный центр;
* есть возможность добавления пользователей с правами;
* большой объем функционала.

Основные минусы:

* неудобный интерфейс;
* платный функционал.

Таким образом был выполнен анализ необходимого функционала, предоставленный аналогами разрабатываемого приложения.

* 1. Анализ и разработка функциональных требований

Анализ требований – процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки ПО.

Цель анализа требований в проектах – получить максимум информации о заказчике и специфике его задач и уточнить рамки проекта.

На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи создаваемого проекта.

Определение и описание требований – шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес–требования;
* пользовательские требования;
* функциональные требования.

Бизнес–требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга.

Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес–требования, поэтому в качестве таких требований будут рассматривать общие требования к разрабатываемому курсовому проекту.

К их этим требованиям относятся:

* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);
* доступ к данным должен осуществляться только через соответствующие хранимые процедуры.

Следующими требованиями являются требования пользователей приложения, а именно администратора, аналитика, поставщика и заказчика.

Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Пользователь данного программного решения должен иметь возможности, соответствующие его роли.

Возможности администратора:

* авторизация в системе;
* регистрация поставщиков и заказчиков в системе;
* возможность нанимать сотрудников;
* возможность строительства склада;
* управление базой данных (удаление записей о поставках, отгрузок и соответствующих им товаров).

Возможности поставщика/заказчика:

* авторизация в системе;
* составление поставки/отгрузки.

Возможности аналитика:

* авторизация в системе;
* ввод запросов для анализа данных;
* получение данных;
* запись результатов в файл.

После проведения анализа были выявлены следующие функциональные требования программного средства:

* вся информация должна храниться в базе данных за исключением верификационной информации регистрируемых пользователей администратором, эта информация сохраняется в файл на компьютере администратора;
* приложение должно производить валидацию вводимых пользователем различных данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации, а именно отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять администратору возможность создания аккаунта для поставщиков или заказчиков в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем.
* должен производиться анализ данных по запросу аналитика.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

1. Проектирование базы данных

Проектирование баз данных представляет собой процесс создания структуры базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных из таблиц по всем необходимым запросам;
* обеспечение целостности базы данных.

Проектирование базы данных проходит через два основных этапа: концептуальное и логическое проектирование.

Концептуальное проектирование представляет собой создание семантической модели предметной области, то есть информационной модели на высоком уровне абстракции.

Концептуально на данном этапе было выделено 9 сущностей:

* пользователь;
* поставщик;
* заказчик;
* работник;
* склад;
* продукт;
* дата;
* факт поставки;
* факт отправки.

Так же были определены необходимые связи. Например, между сущностями «факт поставки» и «продукт» установлена связь один–к–одному. Так же были выделены атрибуты для сущностей. Например, для сущности «поставщик» были выделены атрибуты такие, как «id поставщика», «наименование организации», «адрес», «мобильный номер» а в относящийся к нему сущности «пользователь» были выделены атрибуты такие, как «id пользователя», «роль», «id поставщика», «логин» и «пароль».

Для взаимодействия с таблицами и приложением были определены специальные пользователи «администратор», «поставщик», «заказчик», «аналитик» и служебный пользователь для авторизации.

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных.

Для реляционной модели данных логическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы со структурой связей представлена на рисунке 2.1. В ней показано как будут взаимодействовать таблицы между собой.

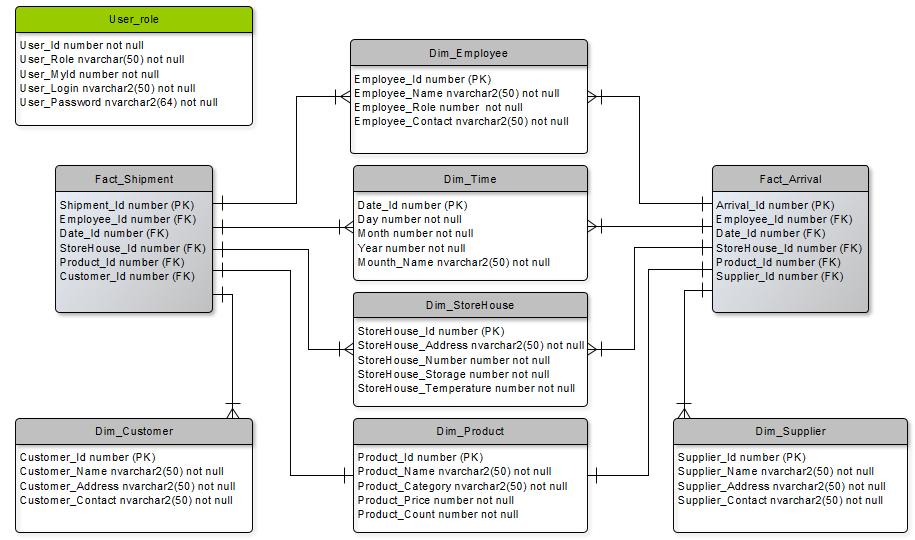


Рисунок 2.1 ­– Логическая модель базы данных

Всего в базе данных содержится 9 таблиц.

Таблица User\_role предназначена для хранения данных зарегистрированных пользователей в базе данный.

Таблица Dim\_Customer предназначена для хранения данных заказчиков.

Таблица Dim\_Supplier предназначена для хранения данных поставщиков.

Таблица Dim\_Time предназначена для хранения данных даты.

Таблица Dim\_StoreHouse предназначена для хранения данных складов.

Таблица Dim\_Product предназначена для хранения данных продукта.

Таблица Dim\_Employee предназначена для хранения данных рабочих складов.

Таблица Fact\_Shipment предназначена для создания связи между остальными таблицами базы данных за исключением User\_role и Dim\_Supplier.

Таблица Fact\_Arrival предназначена для создания связи между остальными таблицами базы данных за исключением User\_role и Dim\_Customer.

Так же была разработана use–case диаграмма, предоставленная на рисунке 2.2.

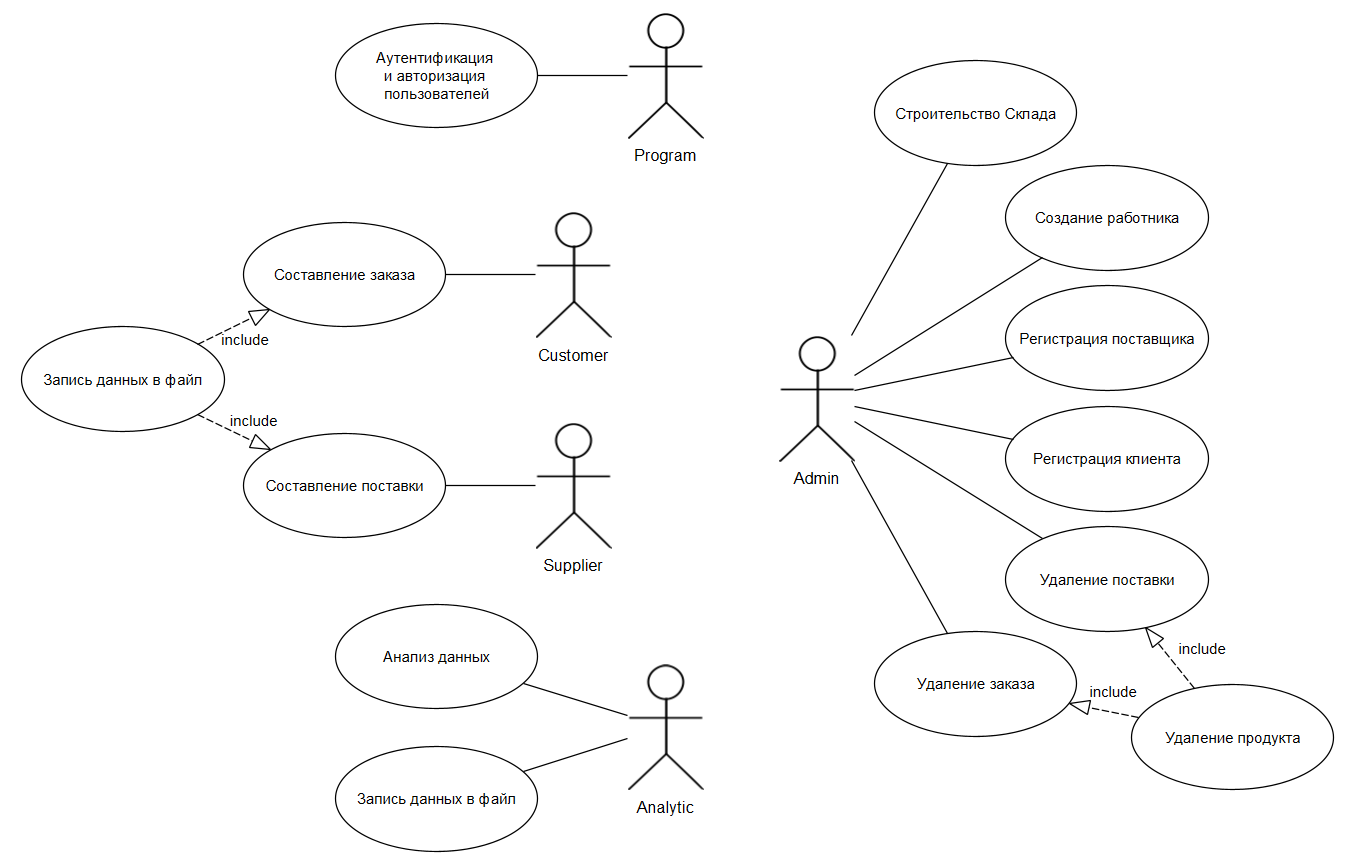


Рисунок 2.2 – Use–case диаграмма

Всего в базе данных содержится 5 пользователей:

Пользователь Program является служебным пользователем необходимым для аутентификации и авторизации.

Пользователь Admin создан для представления администратора в базе данных. Данный пользователь имеет возможность регистрировать поставщиков и заказчиков, добавлять рабочих и склады, а также удалять записи о поставках и заказах вместе с продуктами.

Пользователь Supplier создан для представления поставщика в базе данных. Данный пользователь имеет возможность составлять поставки вместе с поставляемыми продуктами.

Пользователь Customer создан для представления заказчика в базе данных. Данный пользователь имеет возможность составлять заказы вместе с поставляемыми продуктами.

Пользователь Analytic создан для представления аналитика в базе данных. Данный пользователь имеет возможность отправлять запросы в базу данных для получения информации о его работе, а также записывать полученные данные в файл.

1. Разработка необходимых объектов
   1. Таблицы

Таблица – это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк.

Столбцы таблицы называют полями; каждое поле характеризуется своим именем (названием соответствующего свойства) и типом данных, отражающих значения данного свойства. Каждое поле обладает определенным набором свойств (размер, формат и др.).

Поле базы данных – это столбец таблицы, включающий в себя значения определенного свойства.

В каждой таблице должно быть, по крайней мере, одно ключевое поле, содержимое которого уникально для любой записи в этой таблице.

Значения ключевого поля однозначно определяют каждую запись в таблице.

Для реализации базы данных было разработано 9 таблиц: User\_Role, Dim\_Customer, Dim\_Product, Dim\_Employee, Dim\_StoreHouse, Dim\_Time, Dim\_Supplier, Fact\_Shipment, Fact\_Arrival. Они будут реализованы в СУБД Oracle DataBase 19c.

Таблица User\_Role представляет список всех зарегистрированных пользователей в базе данных (Таблица 3.1).

Таблица 3.1 – Столбцы таблицы User\_Role

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| User\_Id | Идентификатор пользователя | integer |
| User\_Role | Роль пользователя | integer |
| User\_MyId | Идентификатор пользователя в соответствующей ему таблице | integer |
| User\_Login | Логин пользователя | nvarchar2(50) |
| User\_Password | Пароль пользователя | nvarchar2(64) |

Таблица Dim\_Customer представляет список всех заказчиков (таблица 3.2).

Таблица 3.2 – Столбцы таблицы Dim\_Customer

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Customer\_Id | Идентификатор заказчика | integer |
| Customer\_Name | Наименование организации заказчика | nvarchar2(50) |
| Customer\_Addres | Адрес заказчика | nvarchar2(50) |
| Customer\_Contact | Контакты заказчика | nvarchar2(50) |

Таблица Dim\_Product представляет список всех продуктов (таблица 3.3).

Таблица 3.3 – Столбцы таблицы Dim\_Product

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Product\_id | Идентификатор продукта | integer |
| Product\_Name | Наименование продукта | nvarchar2(50) |
| Product\_Category | Наименование категории продукта | nvarchar2(50) |
| Product\_Price | Цена продукта | number |
| Product\_Count | Количество продукта | integer |

Таблица Dim\_Employee представляет список всех работников (таблица 3.4).

Таблица 3.4 – Столбцы таблицы Dim\_Employee

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Employee\_Id | Идентификатор рабочего | integer |
| Employee\_Name | Имя рабочего | nvarchar2(50) |
| Employee\_Role | Должность рабочего | nvarchar2(50) |
| Employee\_Contact | Контакты рабочего | nvarchar2(50) |

Таблица Dim\_StoreHouse представляет список всех складов (таблица 3.5).

Таблица 3.5 – Столбцы таблицы Dim\_StoreHouse

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| StoreHouse\_Id | Идентификатор склада | integer |
| StoreHouse\_Addres | Адрес склада | nvarchar2(50) |
| StoreHouse\_Number | Номер склада | integer |
| StoreHouse\_Storage | Вместимость склада | integer |
| StoreHouse\_Temperature | Температура на складе | integer |

Таблица Dim\_Time представляет список всех используемых дат (таблица 3.6).

Таблица 3.6 – Столбцы таблицы Dim\_Time

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Date\_Id | Идентификатор симптома | integer |
| Day | Название симптома | integer |
| Month | Описание симптома | integer |
| Year |  | integer |
| Month\_Name |  | nvarchar2(50) |

Таблица Dim\_Supplier представляет список всех поставщиков (Таблица 3.7).

Таблица 3.7 – Столбцы таблицы Dim\_Supplier

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Supplier\_Id | Идентификатор связи | integer |
| Supplier\_Name | Наименование организации поставщика | nvarchar2(50) |
| Supplier\_Addres | Адрес поставщика | nvarchar2(50) |
| Supplier\_Contact | Контакты поставщика | nvarchar2(50) |

Таблица Fact\_Shipment представляет список связей между таблицами для обобщения заказов (таблица 3.8).

Таблица 3.8 – Столбцы таблицы Fact\_Shipment

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Shipment\_Id | Идентификатор связи | integer |
| Date\_Id | Идентификатор даты | integer |
| Emplayee\_Id | Идентификатор рабочего | integer |
| StoreHouse\_Id | Идентификатор склада | integer |
| Product\_Id | Идентификатор продукта | integer |
| Customer\_Id | Идентификатор заказчика | integer |

Таблица Fact\_Arrival представляет список связей между таблицами для обобщения поставок (таблица 3.8).

Таблица 3.8 – Столбцы таблицы Fact\_Arrival

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Arrival\_Id | Идентификатор связи | integer |
| Date\_Id | Идентификатор даты | integer |
| Employee\_Id | Идентификатор рабочего | integer |
| StoreHouse\_Id | Идентификатор склада | integer |
| Product\_Id | Идентификатор продукта | integer |
| Supplier\_Id | Идентификатор поставщика | integer |

Скрипты создания таблиц и наложение ограничений целостности на них представлены в приложении А данной записки.

* 1. Пользователи

Пользователи в базе данных Oracle ­­­– это активные участники взаимодействия с системой, которые обеспечивают управление и безопасность базы данных. Они проходят процесс аутентификации, чтобы получить доступ к базе данных, и после этого им назначаются определенные полномочия. В базе данных Oracle пользователи выполняют действия, зависящие от предоставленных им полномочий. Полномочия могут быть выданы непосредственно пользователям или через роли.

База данных содержит 5 основных пользователей: Program, Admin, Analytic, Customer и Supplier. Созданием инфраструктуры и объектов БД занимается стандартный в Oracle пользователь SYS с привилегиями SYSDBA. Остальные пользователи могут взаимодействовать с базой данных благодаря полномочиям на вызов процедур и доступом к таблицам выданным им непосредственно пользователем SYS.

Для каждого пользователя был выделен определенный набор привилегий. Пример набора привилегий представлен в листинге 3.1 где выделяются привилегии для пользователя Analytic.

GRANT SELECT ON SYS.Dim\_Customer TO TLKAnalytic;

GRANT SELECT ON SYS.Dim\_Employee TO TLKAnalytic;

GRANT SELECT ON SYS.Dim\_Time TO TLKAnalytic;

GRANT SELECT ON SYS.Dim\_StoreHouse TO TLKAnalytic;

GRANT SELECT ON SYS.Dim\_Product TO TLKAnalytic;

GRANT SELECT ON SYS.Dim\_Supplier TO TLKAnalytic;

GRANT SELECT ON SYS.Fact\_Shipment TO TLKAnalytic;

GRANT SELECT ON SYS.Fact\_Arrival TO TLKAnalytic;

GRANT CREATE SESSION TO TLKAnalytic;

GRANT RESTRICTED SESSION TO TLKAnalytic;

GRANT CREATE PROCEDURE TO TLKAnalytic;

Листинг 3.1 – Выделение привилегий для пользователя Analytic

Пример набора привилегий представлен в листинге 3.2 где выделяются привилегии для пользователя Admin.

GRANT EXECUTE ON DBMS\_CRYPTO TO TLKAdmin;

GRANT CREATE SESSION TO TLKAdmin;

GRANT RESTRICTED SESSION TO TLKAdmin;

GRANT CREATE PROCEDURE TO TLKAdmin;

GRANT INSERT, SELECT ON SYS.USER\_ROLE TO TLKAdmin;

GRANT INSERT, SELECT ON SYS.Dim\_Employee TO TLKAdmin;

GRANT INSERT, SELECT ON SYS.Dim\_StoreHouse TO TLKAdmin;

GRANT INSERT, SELECT ON SYS.Dim\_Customer TO TLKAdmin;

GRANT INSERT, SELECT ON SYS.Dim\_Supplier TO TLKAdmin;

GRANT SELECT, DELETE ON SYS.Fact\_Shipment TO TLKAdmin;

GRANT SELECT, DELETE ON SYS.Fact\_Arrival TO TLKAdmin;

GRANT SELECT, DELETE ON SYS.Dim\_Product TO TLKAdmin;

Листинг 3.2 – Выделение привилегий для пользователя Admin

Создание пользователя Admin приведено в листинге 3.3.

CREATE USER Analytic IDENTIFIED BY AnalyticPass

DEFAULT TABLESPACE TlkZykovDB\_tablespace

TEMPORARY TABLESPACE TlkZykovDB\_tablespace\_TEMP;

Листинг 3.3 – Создания пользователя Analytic

Остальные пользователи базы данных создаются по тому же принципу.

1. Описание процедур импорта и экспорта
   1. Процедура импорта данных в БД

Чтобы импортировать данные в БД был выбран формат JSON.

Для импорта данных создается процедура с названием LoadDataFromJsonEmployee, которая принимаем параметр с типом данных clob, Данная процедура принимает массив json объектов (рабочих) и записывает их в таблицу Dim\_Employee. Код создания таблицы json\_documents представлен в листинге 4.1.

CREATE OR REPLACE PROCEDURE LoadDataFromJsonEmployee(p\_JsonData CLOB) AS

BEGIN

INSERT INTO Sys.Dim\_Employee (Employee\_Name, Employee\_Role, Employee\_Contact)

SELECT \*

FROM JSON\_TABLE(

p\_JsonData,

'$[\*]'

COLUMNS (

Employee\_Name PATH '$.Employee\_Name',

Employee\_Role PATH '$.Employee\_Role',

Employee\_Contact PATH '$.Employee\_Contact'

)

);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLCODE || ' - ' || SQLERRM);

ROLLBACK;

END LoadDataFromJsonEmployee;

Листинг 4.1 – Создание таблицы JSON\_DOCUMENT

Затем необходимо создать директорию, в которой располагаются документы для импорта затем извлечь необходимые данные и отправить запрос на c# коде к этой процедуре. Другие процедуры представлены в приложении Б.

* 1. Процедура экспорта данных из БД

Для экспорта данных из БД был выбран формат JSON. Для того что бы экспортировать данные из БД в JSON были использованы стандартные инструменты среды разработки IntelliJ DataGrip.

1. Тестирование производительности

Для тестирования производительности БД в таблицу USERS было добавлено 100000 строк при помощи импорта, описанного ранее. Для того что бы провести тесты цены запроса и времени его выполнения были разработаны три select запроса затрагивающие выборку с ключевыми полями и без них. Тестирование производиться при помощи инструментов среды разработки IntelliJ DataGrip.

Первый запрос, подремонтированный в листинге 5.1, необходим для теста стоимости и времени запроса при поиске строк с указанным идентификатором.

select \* from Sys.Dim\_Employee where Employee\_Id = 2;

Листинг 5.1 – Первый тестовый запрос

Результат первого тестового запроса показал, что время запроса составляет около 25ms, а цена 2 что является не плохим результатом для поиска с ключевым или же индексированным полем в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.1.

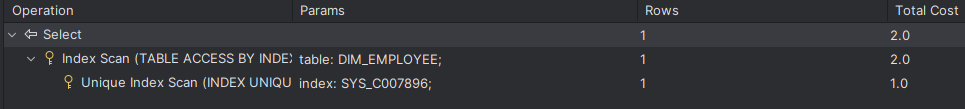


Рисунок 5.1 – Результат выполнения первого тестового запроса

Второй запрос, подремонтированный в листинге 5.2, необходим для теста стоимости и времени запроса при поиске строки с указанным номером телефона.

select \* from Sys.Dim\_Employee

where Employee\_Contact = '+375441046566';

Листинг 5.2 – Второй тестовый запрос

Результат второго тестового запроса показал, что время запроса составляет около 45ms, а цена 445 что является хорошим результатом для поиска без ключевых или же индексированных поле в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.2.

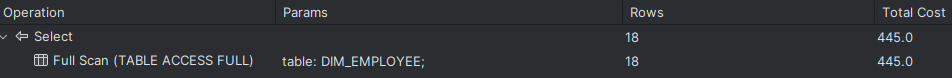


Рисунок 5.2 – Результат выполнения второго тестового запроса

Третий запрос необходим для теста стоимости и времени запроса при поиске строки с указанным именем. Данный тест является важным ведь в приложении часто используется подобный поиск, запрос, подремонтирован в листинге 5.3,

select \* from Sys.Dim\_Employee

where Employee\_Name = 'Работник-46566';

Листинг 5.3 – Третий тестовый запрос

Результат третьего тестового запроса показал, что время запроса составляет около 53ms, а цена 443 что является не плохим результатом для поиска без ключевых или же индексированных полей в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.3.

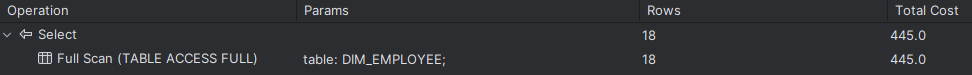


Рисунок 5.3 – Результат выполнения третьего тестового запроса

Однако было принято решение оптимизировать именно этот тип запросов так как он является самым частым по отношению к данной таблице. Был разработан простой индекс код, которого предоставлен в листинге 5.4.

CREATE INDEX idx\_employee\_name ON Sys.Dim\_Employee (Employee\_Name);

Листинг 5.4 – Индекс для таблицы Dim\_Employee

Повторной результат третьего тестового запроса показал, что время запроса снизилось до 15ms, а цена составила 4 что является хорошим результатом для поиска с ключевым или же индексированным полем в БД с 100000 строк. Результат выполнения предоставлен на рисунке 5.4.

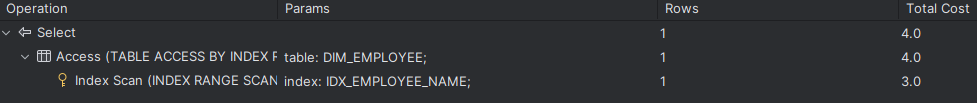


Рисунок 5.4 – Повторный результат выполнения третьего тестового запроса

По результатам проведённых в рамках тестирования производительности базы данных были проведены некоторые работы, направленные оптимизировать производительность. После проведения повторных тестов можно заявлять, что база данных является достаточно производительной что бы не вызывать проблем у пользователя, а также что база данных выдерживает 100000 строк без сильных падений в производительности.

1. Описание технологии применяемой в БД

В рамках данной курсовой работы для обеспечения эффективного анализа данных была применена реляционная технология OLAP (Online Analytical Processing). OLAP представляет собой мощный инструмент, предназначенный для обработки и анализа больших объемов данных, что делает ее идеальным выбором для проекта, связанного с управлением и анализом данных складской системы. Данная технология работает напрямую с реляционной базой данных, используя реляционные таблицы для хранения фактов и измерений.

В данной работе технология OLAP использовалась для организации данных в многомерные кубы, позволяя проводить анализ по временным периодам, продуктам, поставщикам и другим ключевым аспектам работы склада. В реальной бизнес–модели это обеспечило бы более эффективное управление складскими процессами и принятие обоснованных решений на основе данных.

Применение OLAP также существенно сокращает время, необходимое для формирования сложных запросов к БД, что повышает эффективность принятия стратегических решений. Система спроектирована с учетом потребностей аналитиков и менеджеров, предоставляя интуитивно понятные средства взаимодействия с данными. Это подчеркивает не только техническую значимость технологии OLAP, но и ее стратегическую важность в обеспечении конкурентоспособности и успешного развития складского бизнеса.

Таким образом, применение технологии OLAP в контексте разработки базы данных для склада обеспечивает высокую производительность, удобство в анализе и глубокое понимание данных, что является важным фактором для успешного функционирования и оптимизации складской системы.

1. Краткое описание приложения

Для демонстрации работы БД было разработано приложение, написанное на языке С# при помощи фреймворка WPF. В приложении используются разные NuGet пакеты такие как: Oracle.ManagedDataAccess, Newtonsoft.Json и т.д. Для работы с БД были разработаны специальные классы и окна.

Классы ProcedureHelper и Jsoner созданы для работы с данными.

ProcedureHelper создаёт сессию БД под необходимым пользователем, этими пользователями могут быть все ранее описанные пользователи: Admin, Analytic, Supplier и Customer. Для авторизации данных пользователей используется пользователь «Program». Для сопоставления данных из БД в приложение были созданны специальные блоки кода. Блок кода, сопоставляющий данные С# и БД предоставлен в листинге 7.1.

using (OracleCommand command = new OracleCommand("CheckUserLoginPassword", connection))

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add("p\_Login", OracleDbType.NVarchar2).Value = login;

command.Parameters.Add("p\_Password", OracleDbType.NVarchar2).Value = password;

OracleParameter userRoleParam = new OracleParameter("p\_UserRole", OracleDbType.Int32);

userRoleParam.Direction = ParameterDirection.Output;

command.Parameters.Add(userRoleParam);

OracleParameter userMyIdParam = new OracleParameter("p\_UserMyId", OracleDbType.Int32);

userMyIdParam.Direction = ParameterDirection.Output;

command.Parameters.Add(userMyIdParam);

command.ExecuteNonQuery();

int userRole = Convert.ToInt32(userRoleParam.Value);

int userMyId = Convert.ToInt32(userMyIdParam.Value);

}

Листинг 7.1 – Блок кода, сопоставляющий данные С# и БД

Jsoner – это класс отвечающий за работу с json–файлами, с его помощью приложение создает, записывает и читает json–файлы, генерирует json–объекты рабочих для тестирования и др. Этот класс содержит функции такие как: CreateLogAndPassJsonFile, GeneratyEmployeesJsonFile, InputEmployeesToTable.

1. Руководство пользователя

Разработанное в рамках курсового проекта приложение поддерживает четыре основные роли: администратор, аналитик, поставщик и заказчик. Заходя на главную страницу приложения пользователю представляется форма для авторизации, а также кнопка отправки формы и перехода к следующей странице. Скриншот страницы авторизации предоставлен на рисунке 8.1.

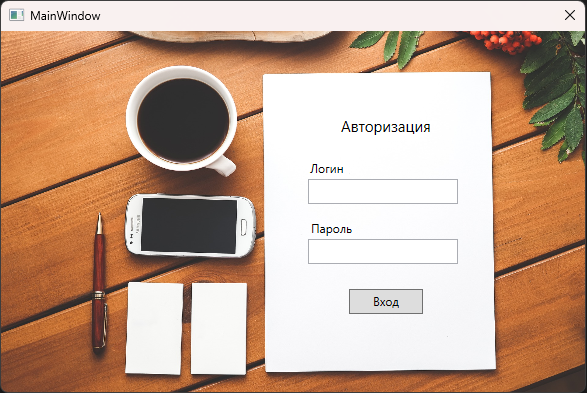


Рисунок 8.1 – Страница авторизации

После авторизации пользователь попадёт на основную страницу, где будет расположены элементы необходимые для работы данного пользователя. В зависимости от роли у пользователя будет разный набор возможный действий.

Администратору доступны кнопки для перехода на страницы добавления рабочего, поставщика, заказчика, склада и удаления записей. Скриншот основной страницы админа приведена на рисунке 8.2

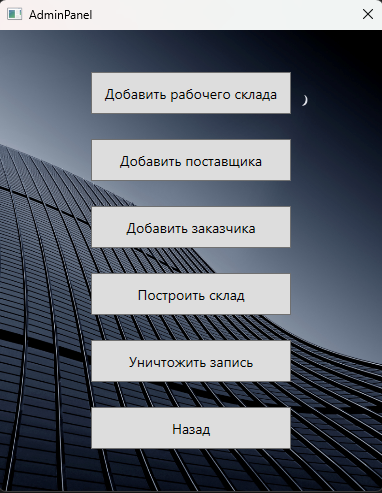


Рисунок 8.2 – Главная страница админа

В каждом окне добавления или удаления объектов есть возможность вводить или указывать данные из предоставленных, есть кнопки для добавления или удаления объектов по предоставленным данным и кнопка возврата на предыдущую страницу. В пример приведено окно добавление поставщика на рисунке 8.3.



Рисунок 8.3 – Обозреватель лекарств

В окне следующего после авторизации поставщика или заказчика есть поля ввода для определения продукта, который они поставляют или заказывают, также присутствуют кнопки для отправки формы в БД и кнопка возврата на предыдущую страницу. В качестве примера приведено окно поставщика для заполнения формы поставки на рисунке 8.4.



Рисунок 8.4 – Окно поставщика

В этом окне требуется ввод следующих данных: наименование, категория, цена и количество товара. Все поля проходят валидацию на наличие данных внутри. В поле цены проверяется ввод отрицательных значений.

В окне следующего после авторизации аналитика есть поле ввода select запроса аналитиком, кнопка выполнения данного запроса и кнопки для вывода определенных данных в таблицу вывода. Скриншот главной страницы аналитика предоставлен на рисунке 8.5.

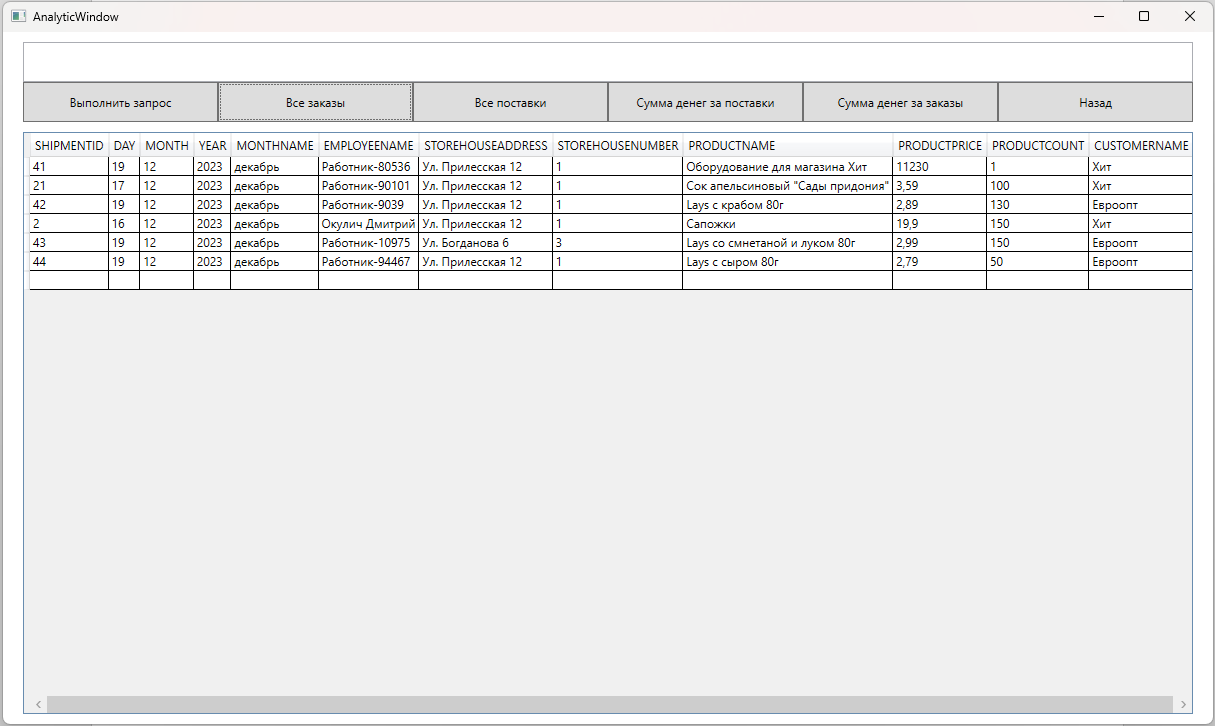


Рисунок 8.5 – Главная страница врача

Поле ввода select запроса проходит проверку на пустое значение, некорректный ввод select запроса и отсутствие ключевых слов для выполнения иных запрещенных запросов.

Таблица вывода подстраивается под различного рода запросы вывода данных и меняет количество и имена колонок под выводимые данные, а также увеличивается и уменьшается в зависимости от количества выводимых данных.

Заключение

Результатом выполнения курсового проекта стало приложение с использованием NuGet пакета Oracle.ManagedDataAccess для взаимодействия с базой данных. В ходе курсового проекта была использована база данных Oracle, которая спроектирована для анализа работы склада.

Были реализованы все основные требования, указанные в листе задания

вместе со следующими пунктами:

* создание роли аналитика и поставщика;
* добавление рабочих и складов;
* прием данных по поставкам на склад;
* использование системы управления базами данных (СУБД).

Возможности администратора:

* управление базой данных работников;
* управление базой данных складов;
* управление базой данных поставок;
* управление базой данных заказов;
* регистрация поставщиков и заказчиков.  
   Возможности аналитика:
* получение данных БД;
* вывод данных в файл;
* ввод select запроса.

Возможности поставщика:

* авторизация;
* запись данных о поставке;
* запись данных о поставляемом продукте.

Возможности поставщика:

* авторизация;
* запись данных о заказе;
* запись данных о заказываемом продукте.

Функциональные требования программного средства:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем различных данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем.

Данный курсовой проект является примером разработки базы данных и приложения для работы с ней. Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] Режим доступа: https://metanit.com – Дата доступа: 14.11.2023.
2. Oracle–dba.ru [Электронный ресурс] / Режим доступа: [https://oracle–dba.ru](https://oracle-dba.ru) – Дата доступа: 15.11.2023.
3. OLAP–Технологии [Электронный ресурс] / Режим доступа: https://eor.dgu.ru/lectures\_f\_8.htm – Дата доступа: 15.11.2023.
4. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm – Дата доступа: 18.11.2023.
5. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 18.11.2023.

Приложение А

CREATE TABLE Dim\_Customer (

Customer\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Customer\_Name NVARCHAR2(50) NOT NULL,

Customer\_Address NVARCHAR2(50) NOT NULL,

Customer\_Contact NVARCHAR2(50) NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Dim\_Employee (

Employee\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Employee\_Name NVARCHAR2(50) NOT NULL,

Employee\_Role NVARCHAR2(50) NOT NULL,

Employee\_Contact NVARCHAR2(50) NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Dim\_Time (

Date\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Day INTEGER NOT NULL,

Month INTEGER NOT NULL,

Year INTEGER NOT NULL,

Month\_Name NVARCHAR2(50) NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Dim\_StoreHouse (

StoreHouse\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

StoreHouse\_Address NVARCHAR2(50) NOT NULL,

StoreHouse\_Number INTEGER NOT NULL,

StoreHouse\_Storage INTEGER NOT NULL,

StoreHouse\_Temperature INTEGER NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Dim\_Product (

Product\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Product\_Name NVARCHAR2(50) NOT NULL,

Product\_Category NVARCHAR2(50) NOT NULL,

Product\_Price NUMBER NOT NULL,

Product\_Count INTEGER NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Dim\_Supplier (

Supplier\_Id NUMBER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Supplier\_Name NVARCHAR2(50) NOT NULL,

Supplier\_Address NVARCHAR2(50) NOT NULL,

Supplier\_Contact NVARCHAR2(50) NOT NULL

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Fact\_Shipment (

Shipment\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Date\_Id INTEGER NOT NULL,

Employee\_Id INTEGER NOT NULL,

StoreHouse\_Id INTEGER NOT NULL,

Product\_Id INTEGER NOT NULL,

Customer\_Id INTEGER NOT NULL,

CONSTRAINT fk\_date FOREIGN KEY (Date\_Id) REFERENCES Dim\_Time (Date\_Id),

CONSTRAINT fk\_employee FOREIGN KEY (Employee\_Id) REFERENCES Dim\_Employee (Employee\_Id),

CONSTRAINT fk\_customer FOREIGN KEY (Customer\_Id) REFERENCES Dim\_Customer (Customer\_Id),

CONSTRAINT fk\_storehouse FOREIGN KEY (StoreHouse\_Id) REFERENCES Dim\_StoreHouse (StoreHouse\_Id),

CONSTRAINT fk\_product FOREIGN KEY (Product\_Id) REFERENCES Dim\_Product (Product\_Id)

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE Fact\_Arrival (

Arrival\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

Date\_Id INTEGER NOT NULL,

Employee\_Id INTEGER NOT NULL,

StoreHouse\_Id INTEGER NOT NULL,

Product\_Id INTEGER NOT NULL,

Supplier\_Id INTEGER NOT NULL,

CONSTRAINT fk\_arrival\_employee FOREIGN KEY (Employee\_Id) REFERENCES Dim\_Employee (Employee\_Id),

CONSTRAINT fk\_arrival\_date FOREIGN KEY (Date\_Id) REFERENCES Dim\_Time (Date\_Id),

CONSTRAINT fk\_arrival\_supplier FOREIGN KEY (Supplier\_Id) REFERENCES Dim\_Supplier (Supplier\_Id),

CONSTRAINT fk\_arrival\_product FOREIGN KEY (Product\_Id) REFERENCES Dim\_Product (Product\_Id),

CONSTRAINT fk\_arrival\_storehouse FOREIGN KEY (StoreHouse\_Id) REFERENCES Dim\_StoreHouse (StoreHouse\_Id)

) tablespace TlkZykovDB\_tablespace;

CREATE TABLE User\_Role (

User\_Id INTEGER generated always as identity (start with 1 increment by 1) PRIMARY KEY,

User\_Role INTEGER NOT NULL,

User\_MyId INTEGER NOT NULL,

User\_Login NVARCHAR2(50) NOT NULL,

User\_Password NVARCHAR2(64) NOT NULL

) tablespace TlkZykovDB\_tablespace;

Приложение Б

Процедуры пользователя Admin

CREATE OR REPLACE PROCEDURE AddCustomer(

p\_Customer\_Name IN NVARCHAR2,

p\_Customer\_Address IN NVARCHAR2,

p\_Customer\_Contact IN NVARCHAR2,

p\_Customer\_Id OUT NUMBER

) AS

BEGIN

BEGIN

INSERT INTO SYS.Dim\_Customer (Customer\_Name, Customer\_Address, Customer\_Contact)

VALUES (p\_Customer\_Name, p\_Customer\_Address, p\_Customer\_Contact)

RETURNING Customer\_Id INTO p\_Customer\_Id;

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

ROLLBACK;

–– Прокидываем исключение дальше

RAISE;

END;

END AddCustomer;

CREATE OR REPLACE PROCEDURE AddCustomerToUserRole(

p\_User\_Role INTEGER,

p\_User\_MyId INTEGER,

p\_User\_Login NVARCHAR2,

p\_User\_Password NVARCHAR2

)

AS

v\_Hash RAW(256); –– Размер хеша (256 бит)

BEGIN

BEGIN

–– Хеширование пароля с использованием SHA–256

v\_Hash := DBMS\_CRYPTO.HASH(Utl\_I18N.STRING\_TO\_RAW(p\_User\_Password, 'AL32UTF8'), DBMS\_CRYPTO.HASH\_SH256);

–– Вставка хешированного пароля в базу данных

INSERT INTO SYS.User\_Role (User\_Role, User\_MyId, User\_Login, User\_Password)

VALUES (p\_User\_Role, p\_User\_MyId, p\_User\_Login, v\_Hash);

COMMIT; –– Фиксация изменений

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

ROLLBACK;

–– Прокидываем исключение дальше

RAISE;

END;

END AddCustomerToUserRole;

CREATE OR REPLACE PROCEDURE CheckUserLoginExists(

p\_User\_Login IN NVARCHAR2,

p\_UserExists OUT INTEGER

)

AS

BEGIN

BEGIN

SELECT COUNT(\*) INTO p\_UserExists

FROM Sys.User\_Role

WHERE User\_Login = p\_User\_Login;

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

ROLLBACK;

–– Прокидываем исключение дальше

RAISE;

END;

END CheckUserLoginExists;

CREATE OR REPLACE PROCEDURE AddSupplier(

p\_Supplier\_Name IN NVARCHAR2,

p\_Supplier\_Address IN NVARCHAR2,

p\_Supplier\_Contact IN NVARCHAR2,

p\_Supplier\_Id OUT NUMBER

)

AS

BEGIN

BEGIN

INSERT INTO Sys.Dim\_Supplier (Supplier\_Name, Supplier\_Address, Supplier\_Contact)

VALUES (p\_Supplier\_Name, p\_Supplier\_Address, p\_Supplier\_Contact)

RETURNING Supplier\_Id INTO p\_Supplier\_Id;

COMMIT; –– Фиксация изменений

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

ROLLBACK;

–– Прокидываем исключение дальше

RAISE;

END;

END AddSupplier;

Процедуры пользователя Analytic

CREATE OR REPLACE PROCEDURE GetArrivalDetails(

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

BEGIN

OPEN p\_Cursor FOR

SELECT fa.Arrival\_Id, t.Day, t.Month, t.Year, t.Month\_Name, e.Employee\_Name,

s.StoreHouse\_Address, s.StoreHouse\_Number, p.Product\_Name, p.Product\_Price,

p.Product\_Count, su.Supplier\_Name

FROM Sys.Fact\_Arrival fa

JOIN Sys.Dim\_Time t ON fa.Date\_Id = t.Date\_Id

JOIN Sys.Dim\_Employee e ON fa.Employee\_Id = e.Employee\_Id

JOIN Sys.Dim\_StoreHouse s ON fa.StoreHouse\_Id = s.StoreHouse\_Id

JOIN Sys.Dim\_Product p ON fa.Product\_Id = p.Product\_Id

JOIN Sys.Dim\_Supplier su ON fa.Supplier\_Id = su.Supplier\_Id;

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

RAISE;

END;

END GetArrivalDetails;

CREATE OR REPLACE PROCEDURE GetArrivalProductsDetails(

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

BEGIN

OPEN p\_Cursor FOR

select fa.Arrival\_ID, p.Product\_Name, p.Product\_Count, p.Product\_Price, (p.Product\_Price \* p.Product\_Count) as "Sum\_Price"

from Sys.Fact\_Arrival fa join Sys.Dim\_Product p

on fa.Product\_ID = p.Product\_ID;

EXCEPTION

WHEN OTHERS THEN

–– Обработка ошибок, например, запись в лог или выброс исключения

DBMS\_OUTPUT.PUT\_LINE('Неприятная ситуация произошла: ' || ' – ' || SQLERRM);

RAISE;

END;

END GetArrivalProductsDetails;

Процедуры пользователя Supplier

CREATE OR REPLACE PROCEDURE ADDDIMTIMESUPPLIER (

p\_Day INTEGER,

p\_Month INTEGER,

p\_Year INTEGER,

p\_Month\_Name NVARCHAR2,

p\_Result OUT INTEGER

) AS

BEGIN

–– Пытаемся найти запись с такими значениями

BEGIN

SELECT Date\_Id INTO p\_Result

FROM Sys.Dim\_Time

WHERE Day = p\_Day AND Month = p\_Month AND Year = p\_Year AND Month\_Name = p\_Month\_Name;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

–– Если запись не найдена, устанавливаем p\_Result в NULL

p\_Result := NULL;

END;

–– Если запись не найдена, вставляем новую

IF p\_Result IS NULL THEN

INSERT INTO Sys.Dim\_Time (Day, Month, Year, Month\_Name)

VALUES (p\_Day, p\_Month, p\_Year, p\_Month\_Name)

RETURNING Date\_Id INTO p\_Result;

–– Фиксация изменений

COMMIT;

END IF;

END ADDDIMTIMESUPPLIER;

CREATE OR REPLACE PROCEDURE AddDimProductSupplier(

p\_Product\_Name IN NVARCHAR2,

p\_Product\_Category IN NVARCHAR2,

p\_Product\_Price IN NUMBER,

p\_Product\_Count IN INTEGER,

p\_Product\_Id OUT INTEGER

)

AS

BEGIN

–– Вставляем новый продукт

INSERT INTO Sys.Dim\_Product (Product\_Name, Product\_Category, Product\_Price, Product\_Count)

VALUES (p\_Product\_Name, p\_Product\_Category, p\_Product\_Price, p\_Product\_Count)

RETURNING Product\_Id INTO p\_Product\_Id;

–– Фиксация изменений

COMMIT;

END AddDimProductSupplier;

CREATE OR REPLACE PROCEDURE GetRandomStoreHouseId(

p\_RandomId OUT INTEGER

)

AS

BEGIN

SELECT StoreHouse\_Id

INTO p\_RandomId

FROM (

SELECT StoreHouse\_Id

FROM Sys.Dim\_StoreHouse

ORDER BY DBMS\_RANDOM.VALUE

)

WHERE ROWNUM = 1;

END GetRandomStoreHouseId;

CREATE OR REPLACE PROCEDURE GetRandomEmployeeId(

p\_RandomId OUT INTEGER

)

AS

BEGIN

SELECT Employee\_Id

INTO p\_RandomId

FROM (

SELECT Employee\_Id

FROM Sys.Dim\_Employee

ORDER BY DBMS\_RANDOM.VALUE

)

WHERE ROWNUM = 1;

END GetRandomEmployeeId;

CREATE OR REPLACE PROCEDURE AddArrival(

p\_Date\_Id INTEGER,

p\_Employee\_Id INTEGER,

p\_StoreHouse\_Id INTEGER,

p\_Product\_Id INTEGER,

p\_Supplier\_Id INTEGER

)

AS

BEGIN

INSERT INTO Sys.Fact\_Arrival (Date\_Id, Employee\_Id, StoreHouse\_Id, Product\_Id, Supplier\_Id)

VALUES (p\_Date\_Id, p\_Employee\_Id, p\_StoreHouse\_Id, p\_Product\_Id, p\_Supplier\_Id);

COMMIT; –– Фиксация изменений

END AddArrival;

Процедуры пользователя Customer

CREATE OR REPLACE PROCEDURE AddDimTimeCustomer(

p\_Day INTEGER,

p\_Month INTEGER,

p\_Year INTEGER,

p\_Month\_Name NVARCHAR2,

p\_Result OUT INTEGER

) AS

BEGIN

–– Пытаемся найти запись с такими значениями

BEGIN

SELECT Date\_Id INTO p\_Result

FROM Sys.Dim\_Time

WHERE Day = p\_Day AND Month = p\_Month AND Year = p\_Year AND Month\_Name = p\_Month\_Name;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

–– Если запись не найдена, устанавливаем p\_Result в NULL

p\_Result := NULL;

END;

–– Если запись не найдена, вставляем новую

IF p\_Result IS NULL THEN

INSERT INTO Sys.Dim\_Time (Day, Month, Year, Month\_Name)

VALUES (p\_Day, p\_Month, p\_Year, p\_Month\_Name)

RETURNING Date\_Id INTO p\_Result;

–– Фиксация изменений

COMMIT;

END IF;

END AddDimTimeCustomer;

CREATE OR REPLACE PROCEDURE AddDimProductCustomer(

p\_Product\_Name IN NVARCHAR2,

p\_Product\_Category IN NVARCHAR2,

p\_Product\_Price IN NUMBER,

p\_Product\_Count IN INTEGER,

p\_Product\_Id OUT INTEGER

)

AS

BEGIN

–– Вставляем новый продукт

INSERT INTO Sys.Dim\_Product (Product\_Name, Product\_Category, Product\_Price, Product\_Count)

VALUES (p\_Product\_Name, p\_Product\_Category, p\_Product\_Price, p\_Product\_Count)

RETURNING Product\_Id INTO p\_Product\_Id;

–– Фиксация изменений

COMMIT;

END AddDimProductCustomer;

CREATE OR REPLACE PROCEDURE GetRandomStoreHouseIdC(

p\_RandomId OUT INTEGER

)

AS

BEGIN

SELECT StoreHouse\_Id

INTO p\_RandomId

FROM (

SELECT StoreHouse\_Id

FROM Sys.Dim\_StoreHouse

ORDER BY DBMS\_RANDOM.VALUE

)

WHERE ROWNUM = 1;

END GetRandomStoreHouseIdC;

CREATE OR REPLACE PROCEDURE GetRandomEmployeeIdC(

p\_RandomId OUT INTEGER

)

AS

BEGIN

SELECT Employee\_Id

INTO p\_RandomId

FROM (

SELECT Employee\_Id

FROM Sys.Dim\_Employee

ORDER BY DBMS\_RANDOM.VALUE

)

WHERE ROWNUM = 1;

COMMIT; –– Фиксация изменений

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' SQLCODE ' – ' || SQLERRM);

ROLLBACK;

END GetRandomEmployeeIdC;

Процедуры пользователя Program

CREATE OR REPLACE PROCEDURE CHECKUSERLOGINPASSWORD (

p\_Login NVARCHAR2,

p\_Password NVARCHAR2,

p\_UserRole OUT NUMBER,

p\_UserMyId OUT NUMBER

) AS

v\_UserRole NUMBER;

v\_UserMyId NUMBER;

v\_StoredPassword NVARCHAR2(64); –– Предположим, что пароль хранится в виде хэша

v\_InputPasswordHash RAW(64); –– Предположим, что вы используете SHA–256

BEGIN

–– Получаем хэш пароля из базы данных

SELECT User\_Role, User\_MyId, User\_Password

INTO v\_UserRole, v\_UserMyId, v\_StoredPassword

FROM SYS.User\_Role

WHERE User\_Login = p\_Login;

–– Если пользователя не найдено, устанавливаем значения по умолчанию

IF SQL%NOTFOUND THEN

v\_UserRole := NULL;

v\_UserMyId := NULL;

ELSE

–– Получаем хэш введенного пароля

v\_InputPasswordHash := DBMS\_CRYPTO.HASH(Utl\_I18N.STRING\_TO\_RAW(p\_Password, 'AL32UTF8'), DBMS\_CRYPTO.HASH\_SH256);

–– Проверяем, совпадает ли хэш пароля

IF v\_InputPasswordHash = v\_StoredPassword THEN

–– Если пароль совпадает, возвращаем роли и MyId

p\_UserRole := v\_UserRole;

p\_UserMyId := v\_UserMyId;

ELSE

–– Если пароль не совпадает, устанавливаем значения по умолчанию

p\_UserRole := NULL;

p\_UserMyId := NULL;

END IF;

END IF;

END CHECKUSERLOGINPASSWORD;

DECLARE

v\_Login NVARCHAR2(50) := 'admin'; –– Введите нужное имя пользователя

v\_Password NVARCHAR2(50) := 'admin'; –– Введите нужный пароль

v\_UserRole NUMBER;

v\_UserMyId NUMBER;

BEGIN

CHECKUSERLOGINPASSWORD(

p\_Login => v\_Login,

p\_Password => v\_Password,

p\_UserRole => v\_UserRole,

p\_UserMyId => v\_UserMyId

);

–– Выводим результаты

DBMS\_OUTPUT.PUT\_LINE('User Role: ' || v\_UserRole);

DBMS\_OUTPUT.PUT\_LINE('User MyId: ' || v\_UserMyId);

END;