МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ



Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных интернет-магазина компьютерных комплектующих с применением технологии резервного копирования и восстановления»

Выполнил студент Авдей Алексей Юрьевич

(Ф.И.О.)

Руководитель работы

(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 2024

Содержание



[Введение 3](#_Toc166899442)

[1.1 Обзор аналогичных решений 5](#_Toc166899443)

[1.2 Техническое задание 7](#_Toc166899444)

[1.3 Выводы 8](#_Toc166899445)

[2 Проектирование базы данных 8](#_Toc166899446)

[2.1 Определение вариантов использования 8](#_Toc166899447)

[2.2 Диаграммы UML, взаимодействие всех компонентов 9](#_Toc166899448)

[2.3 Вывод 10](#_Toc166899449)

[3 Разработка объектов базы данных 10](#_Toc166899450)

[3.1 Разработка таблиц базы данных 10](#_Toc166899451)

[3.2 Разработка представлений базы данных 12](#_Toc166899452)

[3.3 Разработка индексов базы данных 14](#_Toc166899453)

[3.4 Разработка процедур базы данных 14](#_Toc166899454)

[3.5 Разработка функций базы данных 15](#_Toc166899455)

[3.5 Разработка триггеров базы данных 16](#_Toc166899456)

[3.6 Разработка пользователей 16](#_Toc166899457)

[3.7 Вывод 18](#_Toc166899458)

[4 Описание процедур импорта и экспорта 19](#_Toc166899459)

[5 Тестирование производительности 20](#_Toc166899460)

[6 Описание технологии и ее применение в базе данных 21](#_Toc166899461)

[7 Руководство пользователя 23](#_Toc166899462)

[Заключение 24](#_Toc166899463)

[Список используемых источников 25](#_Toc166899464)

[Приложение А. Листинг создания таблиц 26](#_Toc166899465)

[Приложение Б. Листинг процедур базы данных 27](#_Toc166899466)

[Приложение В. Листинг функций базы данных 30](#_Toc166899467)

[Приложение Г. Листинг представлений базы данных 30](#_Toc166899468)

[Приложение Д. Листинг кода создания пользователей 31](#_Toc166899469)

[Приложение Е. Листинг процедур для импорта и экспорта в xml 33](#_Toc166899470)

# Введение

Целью работы является разработка базы данных для управления информацией о недвижимости, клиентах, сделках и контрактах. Создание такой базы данных позволит эффективно организовать работу агентства недвижимости, обеспечив удобный доступ к информации о доступных объектах, а также точный и надежный учет сделок. Кроме того, целью работы может быть создание онлайн-платформы для поиска и предложения недвижимости, которая автоматизирует процесс поиска объектов и заключения сделок, а также обеспечивает мониторинг и контроль за выполнением контрактов. Данная база данных обеспечит эффективное хранение данных, доступ к ним и их фильтрацию.

База данных (БД) – это система, предназначенная для хранения, организации и управления данными в структурированном виде, которая обеспечивает удобный доступ к этим данным и позволяет эффективно обрабатывать и анализировать большие объемы информации. База данных может содержать информацию о различных объектах недвижимости (например, квартирах, домах, земельных участках и т.д.), клиентах, сделках и контрактах, а также связи между ними и правила для доступа и использования этой информации. База данных используется в различных областях, включая бизнес, науку и другие.

Для обеспечения функциональности приложения используются технологии резервного копирования базы данных, чтобы защитить информацию от потери. В пояснительной записке содержится информация о необходимых моментах реализации, структуре и реализации проекта, а также инструкции по использованию приложения.

**1 Постановка задачи**

## 1.1 Обзор аналогичных решений

Для того, чтобы понять какие пункты необходимы для реализации при создании базы данных Агенства Недвижимости необходимо изучить его аналоги, чтобы знать каким образом лучше хранить данные.

В качестве аналогов были выбраны сайты интернет-магазинов и онлайн каталогов, а именно: onliner.by, hata.by, realt.by. Данные сайты имеют необходимый функционал.

Первым был выбран сайт onliner.by, фотография его страницы представлена на рисунке 1.1.

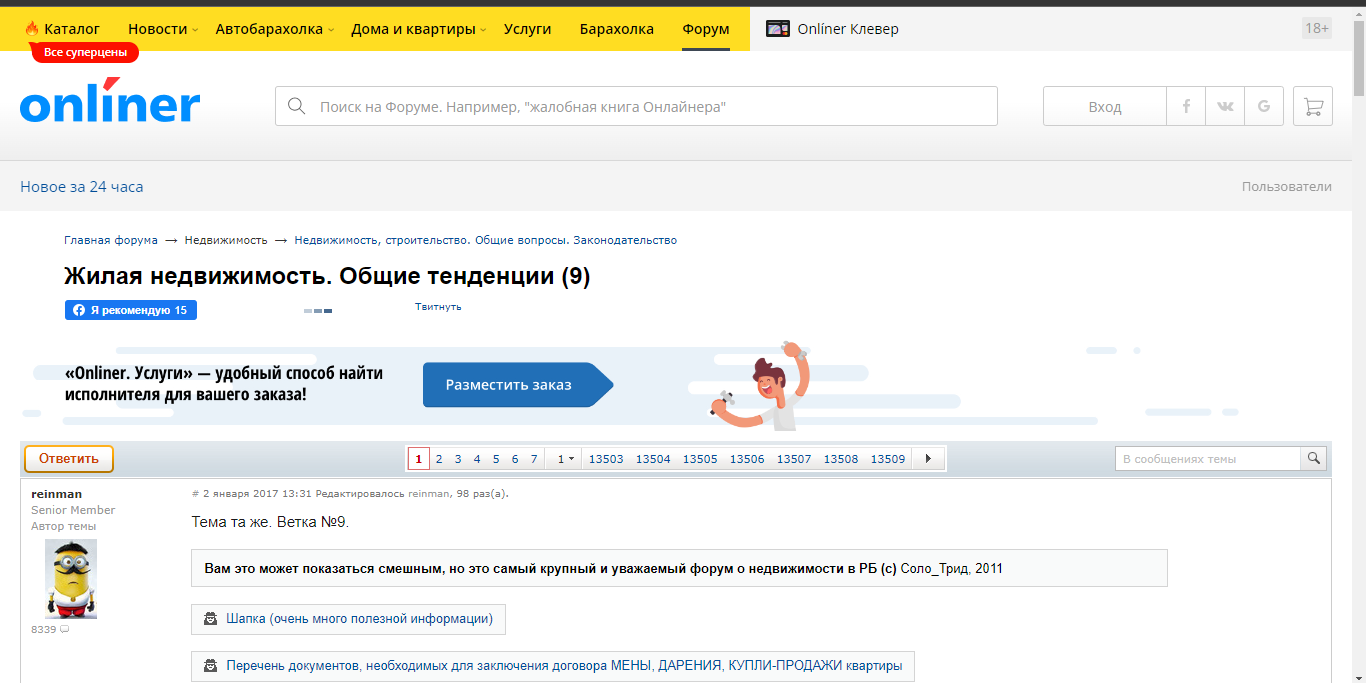


Рисунок 1.1 – Оформление сайта «onliner.by»

У сайта имеется распределение объектов по категориям и подкатегориям.

При анализе функционала были выявлены следующие преимущества:

* Возможность сортировки объектов.
* Возможность фильтрации объектов по различным параметрам.

Для учета пользователей используются электронная почта и пароль.

Следующий аналог – hata.by – онлайн каталог различных товаров. Ниже, на рисунке 1.2, представлена одна из его страниц.

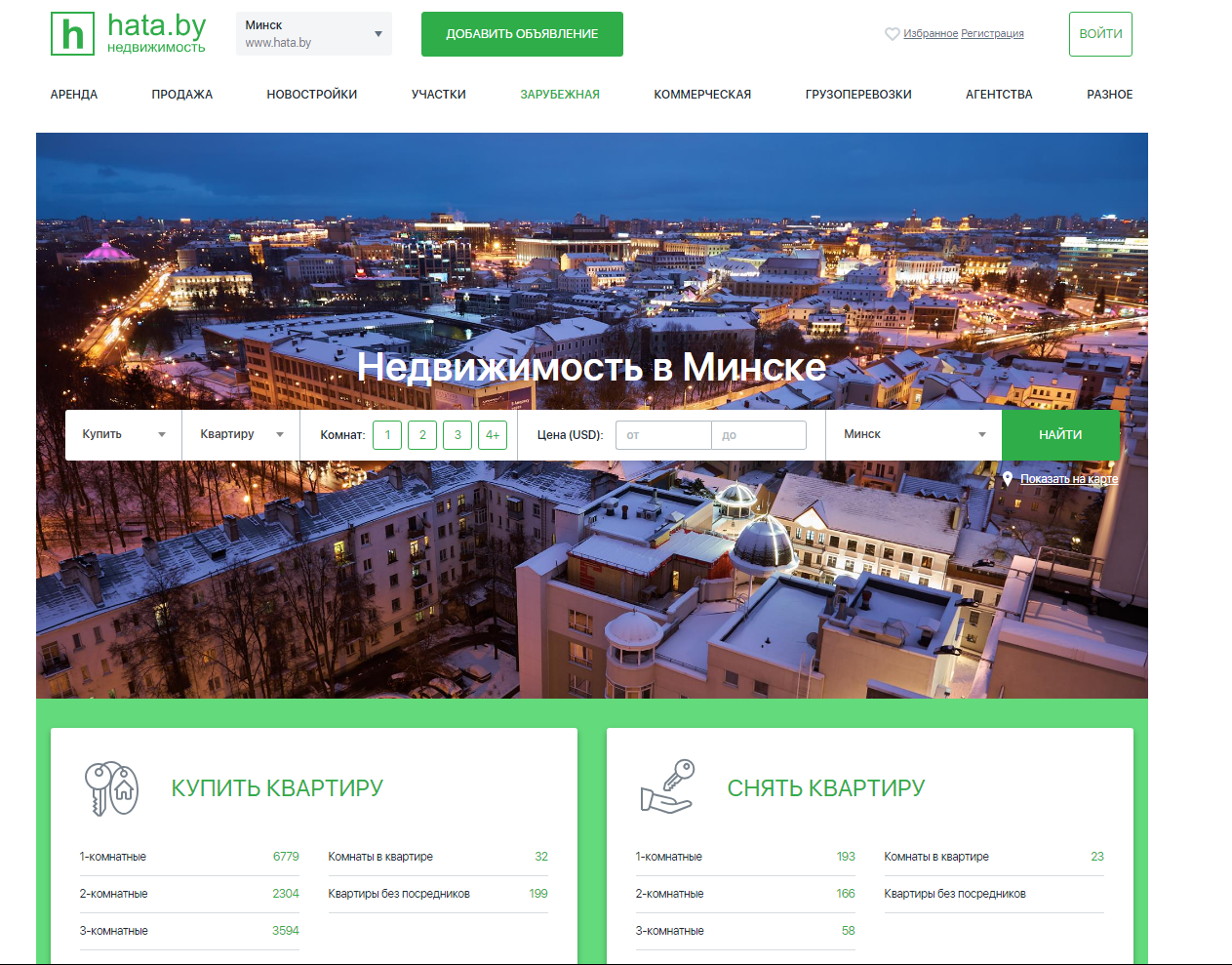


Рисунок 1.2 – Оформление сайта «hata.by»

Онлайн-каталог содержит в себе описания объектов, их описание, информацию о продавцах.

При анализе функционала были выявлены следующие преимущества:

* Указание количества найденных объектов.
* Возможна фильтрация по продавцу и городу.
* Большой набор параметров для фильтрации.
* Имеются теги для акционных объектов и спецпредложений.

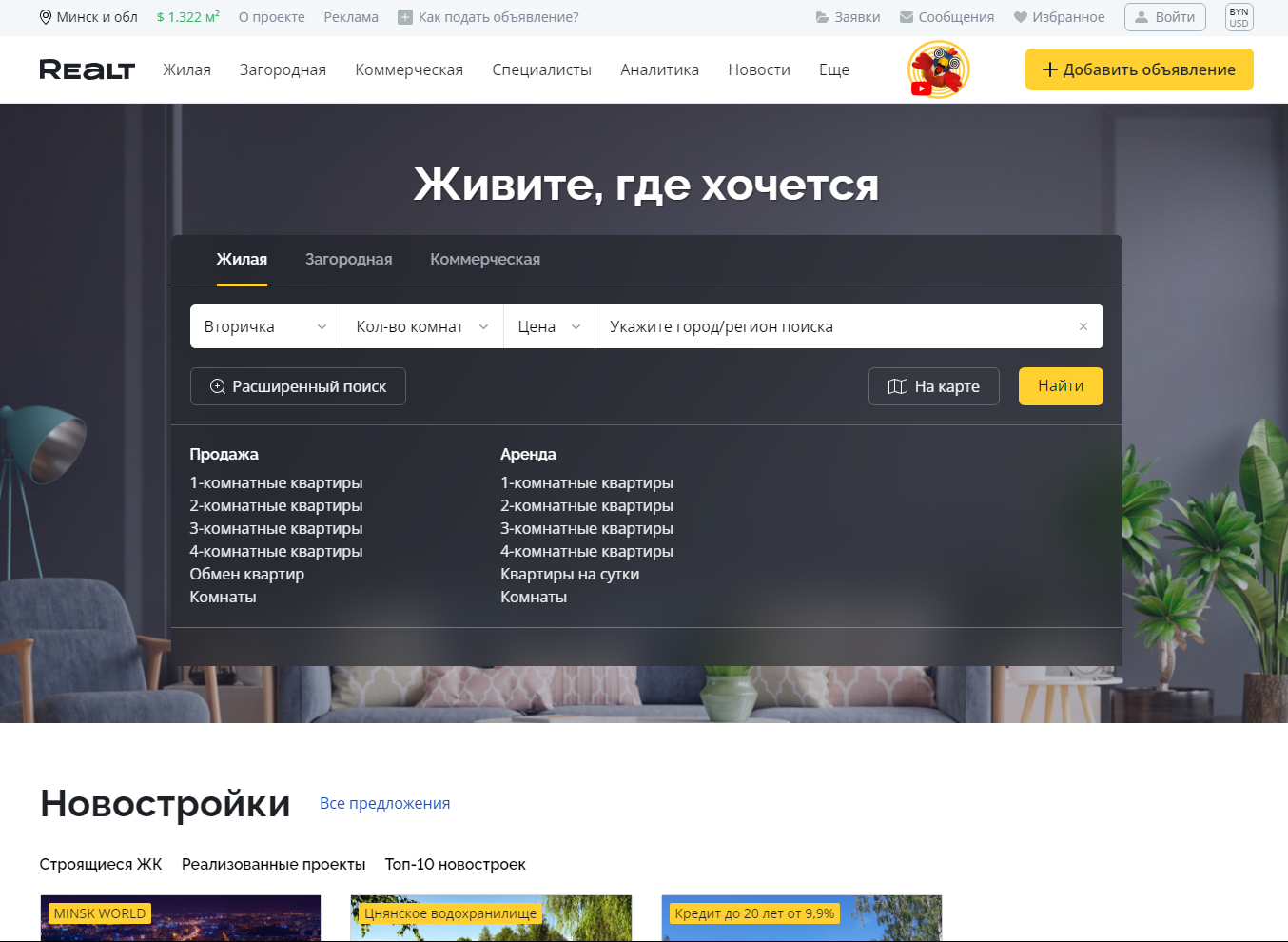


Рисунок 1.3 – Оформление сайта «realt.by»

Третий аналог сайт «realt.by». На рисунке 1.3 показана страница с объектами данного сайта.

На сайте объекты отсортированы по категориям, имеется бонусная программа, теги акций и пояснения для определенных параметров. Пользователи входят на сайт с помощью почты или номера телефона и пароля.

При анализе функционала были выявлены следующие преимущества:

* Указание скидок и первоначальной цены.
* Указание размера скидки.
* Указание числа бонусов и возможности их списания.

К недостаткам в функционале веб-сайта можно отнести малое количество параметров для фильтрации.

## 1.2 Техническое задание

Важным этапом разработки любого нового продукта является анализ существующих на рынке аналогов. Он позволяет определить, какие решения уже представлены потребителям, выявить их сильные и слабые стороны, а также определиться с направлениями для усовершенствований.

Агентство недвижимости – это компания, занимающаяся продажей, арендой и управлением объектами недвижимости. Такие агентства предлагают своим клиентам широкий спектр услуг, включая маркетинг объектов, управление арендаторами и поддержку сделок купли-продажи. На текущий момент рынок программного обеспечения для агентств недвижимости представляет собой множество платформ с различным уровнем функционала и безопасности. Однако, многие из них не могут предложить полноценного комплекса инструментов, сочетающего в себе удобство, гибкость и высокие стандарты безопасности. Это оставляет простор для новых разработок в этой сфере.

Задача проекта: Разработать архитектуру приложения для агентства недвижимости, взаимодействие с которой будет понятно любому пользователю. Построить базу данных и выполнить тестирование готового продукта. Должны быть выполнены следующие требования:

˗ база данных должна быть реализована в СУБД Oracle.

˗ 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 10000 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.

## 1.3 Выводы

Анализ существующих решений и разработка базы данных для агентства недвижимости показали, что использование встроенных возможностей Oracle, таких как RMAN для резервного копирования, процедур и функций для управления данными, а также изоляция таблиц от прямого доступа пользователей, позволяет создать надежную и безопасную систему. Это решение обеспечивает удобство использования, гибкость и высокие стандарты безопасности, что является ключевым фактором для успешной работы агентства недвижимости.

# 2 Проектирование базы данных

## 2.1 Определение вариантов использования

Главное начало функциональных требований базы данных заключается в определении способа обработки данных и обеспечении необходимой функциональности пользовательскому интерфейсу. Важно описать, как данные должны быть хранены и организованы, как осуществляется поиск и выборка данных, как обновляются данные и как обеспечивается безопасность информации. Кроме того, следует учесть взаимодействие базы данных с другими системами и программами. Например, для интернет-магазина функциональные требования могут включать в себя функции для хранения информации аукционах и их лотах.

Кроме функциональных требований, важно также определить роли пользователей и их варианты использования системы. Варианты использования описывают, как пользователи будут взаимодействовать с системой в зависимости от своих ролей. Это помогает определить, какие функции должны быть доступны для каждой роли, какие данные должны быть доступны для каждой роли, а также как должна быть организована навигация в системе. Варианты использования обычно представляются в виде Use Case диаграмм, которые позволяют наглядно отобразить взаимодействие между пользователями и системой. На рисунке 2.1 представлена UML диаграмма вариантов использования.

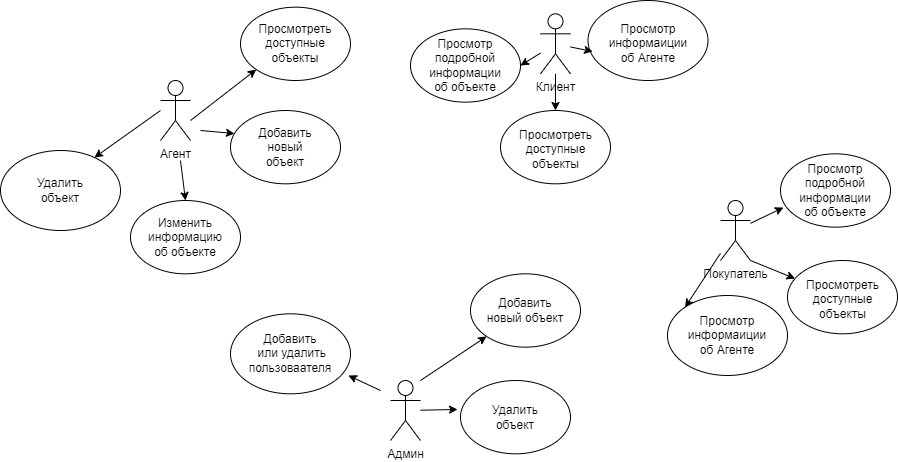


Рисунок 2.1 – UML диаграмма вариантов использования

## 2.2 Диаграммы UML, взаимодействие всех компонентов

Диаграмма базы данных таблиц (Database Table Diagram) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных представлена на рисунке 2.2. 

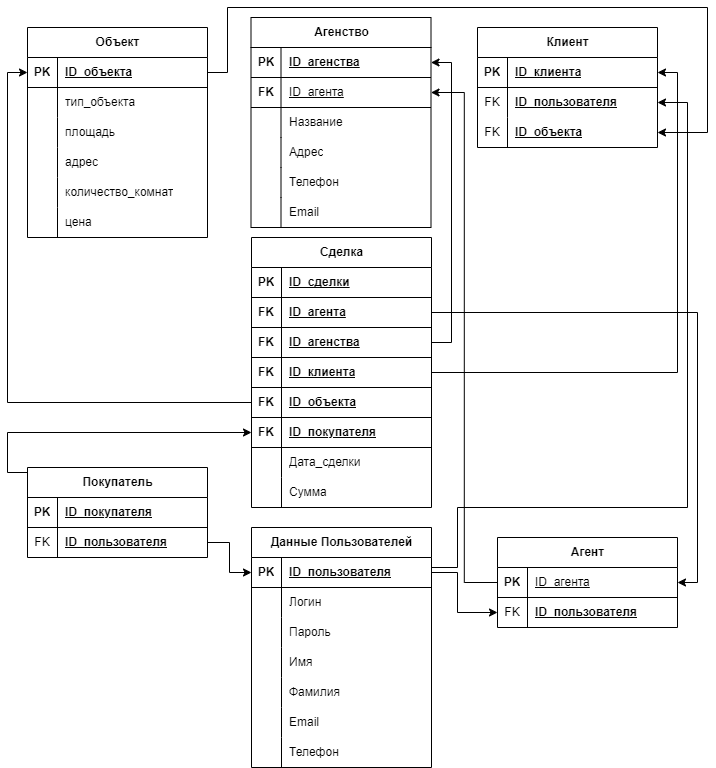




Рисунок 2.2 – Диаграмма базы данных

Видно, что диаграмма показывает связи между таблицами и полями, а также отношения между ними. Одна из основных связей – это связь "один ко многим" (one-to-many), которая определяет отношение одной записи в таблице к нескольким записям в другой таблице.

Например, таблица parts связана с таблицей order, через внешний ключ part\_id. Также видно, что таблицы customers и adress связаны друг с другом через внешний и первичный ключ email. Все связи представлены на рисунке 2.2, который располагается выше.

Для обеспечения безопасности данных необходимо реализовать обращение к данным через процедуры и реализовать резервное копирование на базе rman.

## 2.3 Вывод

Такая архитектура базы данных предоставляет гибкость в управлении данными, высокую производительность и безопасность. Структурированное разделение на авторизацию и доменную логику позволяет оптимально использовать ресурсы системы и обеспечивать эффективное взаимодействие с данными.

# 3 Разработка объектов базы данных

## 3.1 Разработка таблиц базы данных

При разработке приложения для курсового проекта была использована база данных Oracle.

Для реализации базы данных было разработано 8 таблиц. В структуру схемы базы данных для проекта входят следующие таблицы: Пользователи, Лоты, Аукционы и Ставки. Ниже будет описание про каждую из них более подробно.

Таблица Object содержит информацию об объекте. Состоит из следующих столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы Object

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ObjectID | ID объекта | INT |
| ObjectType | Пароль пользователя | Varchar(255) |
| Area | Площадь | DECIMAL(10,2) |
| NumRooms | Количество комнат | INT |
| Price | Цена | DECIMAL(12,2) |
| Address | Адрес объекта | VARCHAR(255) |

Таблица Agency содержит информацию об Агенстве. В таблице Address хранятся адреса объектов. Состоит из следующих столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы Agency

|  |  |  |
| --- | --- | --- |
| AgencyID | Уникальный идентификатор агенства, первичный ключ | INT |
| AgentID |  | INT |
| Name | Имя товара | VARCHAR(255) |
| Address | Адрес Агенства | VARCHAR(255) |
| Email |  | VARCHAR(255) |
| Phone |  | VARCHAR(20) |

Таблица Client хранит информацию о клиентах, содержит следующие столбцы (таблица 3.3):

Таблица 3.3 – Столбцы таблицы Client

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ClientID | Уникальный идентификатор клиента, первичный ключ | INT |
| ObjectID | ID объекта | INT |
| UserID | ID пользователя | INT |

Таблица Deal содержит данные о заказах. Содержит в себе такие столбцы, такие как (таблица 3.4):

Таблица 3.4 – Столбцы таблицы Deal

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| DealID | ID сделки | INT |
| AgencyID | ID агенства | INT |
| ClientID | ID клиента | INT |
| ObjectID | ID объекта | INT |
| BuyerID | ID покупателя | INT |
| DealDate | Дата сделки | DATE |
| Amount | Сумма | DECIMAL(12,2) |

Таблица Buyers содержит данные о покупателях. Содержит в себе такие столбцы, такие как (таблица 3.5):

Таблица 3.5 – Столбцы таблицы Buyers

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| BuyerID | Уникальный идентификатор элемента покупателя, первичный ключ | INT |
| User\_ID | ID пользователя | INT |

Таблица Agents содержит данные о агентах. Содержит в себе такие столбцы, такие как (таблица 3.6):

Таблица 3.6 – Столбцы таблицы Agents

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| AgentID | Уникальный идентификатор элемента покупателя, первичный ключ | INT |
| User\_ID | ID пользователя | INT |

Таблица DataUsers содержит данные о польователях. Содержит в себе такие столбцы, такие как (таблица 3.7):

Таблица 3.7 – Столбцы таблицы DataUsers

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| User\_ID | Уникальный идентификатор Пользователя с данными, первичный ключ | INT |
| Login | Логин | VARCHAR(10) |
| Password | Пароль | VARCHAR(10) |
| Name | Имя | VARCHAR(255) |
| LastName | Фамилия | VARCHAR(100) |
| Address | Адрес | VARCHAR(255) |
| Email | Почта | VARCHAR(255) |
| Phone | Телефон | VARCHAR(20) |

Перед созданием таблиц, для их структуризации и изоляции было принято решение использовать табличные пространства.

Табличное пространство – это механизм, который помогает связать объекты базы данных, такие как таблицы, индексы и представления, с файловой системой. Оно позволяет логически разделять объекты базы данных на разные физические устройства или диски, что может улучшить производительность работы с базой данных. Скрипт для создания табличных пространств представлен в листинге 3.1.

|  |
| --- |
| create tablespace agency\_ts  DATAFILE 'agency\_ts.dbf'  SIZE 10m  AUTOEXTEND ON NEXT 10m MAXSIZE UNLIMITED;  create temporary tablespace agency\_ts\_tmp  TempFILE 'agency\_ts\_temp.dbf'  SIZE 10m  AUTOEXTEND ON NEXT 10m MAXSIZE UNLIMITED; |

Листинг 3.1 – Cкрипт для создания табличных пространств

Листинг SQL-кода для создания таблиц находится в [приложении А](#_Приложение_А).

Для эффективного использования базы данных в проекте, необходимо создать индексы на столбцах, используемых в запросах с поиском данных. Для таблицы parts был создан индекс на основании столбцов type, manufacturer и model, так как они редко изменяются и часто используются для поиска.

Для более изоляции пользователя от таблиц и облегчения некоторого функционала создано несколько процедур и представлений.

## 3.2 Разработка представлений базы данных

Представление в базе данных – это подмножество базы данных, основанное на запросе, который запущен для одной или нескольких таблиц базы данных. Представления базы данных хранятся в базе данных как именованные запросы и могут использоваться для сохранения часто используемых сложных запросов. Представления позволяют объединять данные из нескольких таблиц в одну, не изменяя исходных таблиц и их структуры. Представления используются для обеспечения удобства доступа и управления данными в базе данных, а также для изоляции пользователя от оригинальных таблиц. В данном проекте были созданы представления:

Object\_View – содержит все объекты и владельцев

Buyers\_View – содержит всех покупателей

All\_Objects\_View – содержит все объекты

Client\_View – содержит всех клиентов

Deal\_View – содержит всю информацию по сделкам

|  |
| --- |
| --Client\_View  CREATE OR REPLACE VIEW Client\_View AS  SELECT Client.ClientID, Client.UserID, DataUsers.Name, DataUsers.LastName, DataUsers.Email, DataUsers.Phone  FROM Client  INNER JOIN DataUsers ON Client.UserID = DataUsers.UserID;  --- сделки Deal  CREATE OR REPLACE VIEW Deal\_View AS  SELECT D.DealID, O.ObjectID , O.ObjectType AS "Объект", D.DealDate AS "Дата Сделки", D.Amount AS "Сумма", DU.Name AS "Имя Покупателя", DU.LastName AS "Фамилия Покупателя", DU.Phone AS "Номер Телефона"  FROM Deal D  INNER JOIN Buyers B ON B.BuyerID = D.BuyerID  INNER JOIN DataUsers DU ON DU.UserID = B.UserID  INNER JOIN Object O ON O.ObjectID = D.ObjectID;  --все объекты и клиенты  CREATE OR REPLACE VIEW Object\_View AS  select ObjectType as Type, Addres, Area, NumRooms, Price, Name, LastName, Email, Phone  from Object  inner join Client on Object.ObjectID = Client.ObjectID  inner join DataUsers on Client.UserID = DataUsers.UserID;  --- объекты  CREATE OR REPLACE VIEW All\_Objects\_View AS  select \* from Object;  --- все покупатели  CREATE OR REPLACE VIEW Buyers\_View AS  SELECT Buyers.BuyerID, Buyers.UserID, DataUsers.Name, DataUsers.LastName, DataUsers.Email, DataUsers.Phone  FROM Buyers  INNER JOIN DataUsers ON Buyers.UserID = DataUsers.UserID; |

Листинг 3.2 – Представления Object\_View, All\_Objects\_View, Buyers\_View и Client\_View

Представление Object\_View содержит все объекты и клиенты.

Представление All\_Objects\_View содержит данные о объектах.

Представление Buyers\_View содержит данные о покупателях.

Представление Client\_View содержит данные о клиентах.

## 3.3 Разработка индексов базы данных

Индекс в базе данных представляет собой объект, который используется для ускорения поиска данных. Если таблица содержит большое количество строк, то последовательный поиск данных может занимать много времени. Индекс создается на основе значений одного или нескольких столбцов таблицы и указывает на соответствующие строки таблицы. Использование индексов помогает улучшить производительность базы данных, поскольку они имеют оптимизированную структуру для поиска, например, сбалансированное дерево. Один из наиболее распространенных алгоритмов индексации – это B-дерево (B-tree). B-дерево – это сбалансированное дерево поиска, которое обеспечивает эффективный поиск элементов, используя ключи для сортировки данных в индексе.

|  |
| --- |
| CREATE INDEX object\_Price ON Object(Price); |

Листинг 3.3 – Индексы таблицы Object

Применение индексов в базе данных значительно повышает скорость выполнения операций поиска, сортировки и фильтрации данных, особенно в случае больших объемов информации. Однако создание индексов может занять дополнительное время при добавлении или изменении данных в таблицах. Поэтому необходимо балансировать количество и типы индексов, чтобы обеспечить оптимальную производительность базы данных. А также использовать индексы на значениях, которые изменяются редко или вообще не меняются.

## 3.4 Разработка процедур базы данных

Подпрограммы PL / SQL называются блоками PL / SQL, которые могут быть вызваны с помощью набора параметров. PL / SQL предоставляет два вида подпрограмм: функции и процедуры. Процедуры не могут возвращать значений (но могут иметь выходные параметры) и имеют возможность в теле использовать DML-операторы. Всего было разработано мною 24 процедуры. На листинге 3.4 представлен пример реализации одной из процедур.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE create\_client (  p\_Login VARCHAR2,  p\_Password VARCHAR2,  p\_Name VARCHAR2,  p\_LastName VARCHAR2,  p\_Email VARCHAR2,  p\_Phone VARCHAR2  )  IS  v\_UserID INT;  BEGIN  DECLARE  v\_ClientID INT;  BEGIN  SELECT client\_id\_sequence.NEXTVAL INTO v\_ClientID FROM DUAL;  INSERT INTO DataUsers (UserID, Login, Password, Name, LastName, Email, Phone)  VALUES (v\_ClientID, p\_Login, p\_Password, p\_Name, p\_LastName, p\_Email, p\_Phone);  INSERT INTO Client (ClientID, ObjectID, UserID)  VALUES (v\_ClientID, v\_ClientID, v\_ClientID);  DBMS\_OUTPUT.PUT\_LINE('Данные успешно добавлены в таблицы DataUsers и Client.');  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Нарушение уникального ограничения. Дублирующиеся значения.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка ' || SQLCODE || ': ' || SQLERRM);  END;  END; |

Листинг 3.4 – Процедура создания клиента

Данная процедура позволяет создавать клиентов, в качестве параметров принимает поля таблицы DataUsers. Скрипты других процедур приведен в [приложении Б](#_Приложение_В).

## 3.5 Разработка функций базы данных

Функции в Oracle - это именованные блоки программного кода, которые принимают входные значения, обрабатывают их и возвращают результат. Функции используются для выполнения различных операций, например, для вычислений, обработки данных. В отличие от процедур функции используются для получения значений или их получение в обработанном виде. В листинге 3.5 указан код функции

|  |
| --- |
| CREATE OR REPLACE FUNCTION calculate\_vat (  p\_amount DECIMAL  )  RETURN DECIMAL  IS  v\_vat\_rate DECIMAL := 1.2;  v\_vat DECIMAL;  BEGIN  v\_vat := p\_amount \* v\_vat\_rate;  RETURN v\_vat;  END calculate\_vat; |

Листинг 3.5 – Скрипт функции calculate\_vat

calculate\_vat. Данная функция выводит данные о сумме за сделку, на вход она принимает сумму за объект.

## 3.5 Разработка триггеров базы данных

Триггер в базе данных – это объект, который автоматически выполняет определенные действия при возникновении определенных событий в таблице или представлении базы данных. Триггер может быть настроен на срабатывание при вставке, обновлении или удалении строк в таблице.

Триггеры используются для поддержки целостности данных, контроля доступа к данным и автоматической обработки данных при выполнении определенных операций в таблице.

в листинге 3.6 приведен код триггера, уведомляющего о добавлении пользователя.

|  |
| --- |
| CREATE OR REPLACE TRIGGER trg\_deal  BEFORE UPDATE ON Deal  FOR EACH ROW  BEGIN  :NEW.AgentID := 1;  :NEW.AgencyID:= 1;  END; |

Листинг 3.6 – Скрипт триггеров

Триггер trg\_deal создается для Deal и устанавливает ID на 1.

## 3.6 Разработка пользователей

Пользователь базы данных — это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. На каждом этапе развития БД (проектирование, реализация, эксплуатация, модернизация и развитие, полная реорганизация) с ней связаны разные категории пользователей.

При проектировании базы данных было создано 3 типа пользователей для подключения к базе данных: покупатель, продавец и разработчик.

Создание пользователей представлено в листинге 3.7.

|  |
| --- |
| create role Agent\_role;  create role Buyers\_role;  create role Client\_role;  create user Agent  identified by Agent  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Agent  account unlock  password expire;  create user Buyers  identified by Buyers  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Buyers  account unlock  password expire;  create user Client  identified by Client  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Client  account unlock  password expire;  grant  create session,  create table,  create view  to Agent\_role;  grant execute on GetObjectData to Agent\_role;  grant execute on GetObjectDataByType to Agent\_role;  grant  create session,  create table,  create view  to Agent\_role;  grant select on Object\_View TO Agent\_role;  grant select on Buyers\_View TO Agent\_role;  grant select on All\_Objects\_View to Agent\_role;  grant select on Client\_View to Agent\_role;  grant select on Deal\_View to Agent\_role;  grant execute on create\_object to Agent\_role;  grant execute on create\_client to Agent\_role;  grant execute on create\_buyers to Agent\_role;  grant execute on test\_insert\_performance to Agent\_role;  grant execute on create\_deal to Agent\_role;  grant  create session  to Buyers\_role;  grant  create session  to Client\_role; |

Листинг 3.6 – Создание пользователей

Далее для созданного пользователя базы данных были выданы необходимые привилегии необходимые для функционала клиента.

## 3.7 Вывод

В данном разделе была рассмотрена разработка объектов базы данных для Агенства Недвижимости. Все рассмотренные объекты вместе создают сложную, но гибкую и эффективную структуру, которая обеспечивает правильное функционирование и управление данными в системе. Благодаря хорошо спроектированным объектам базы данных система способна обеспечивать надежность, своевременное быстродействие и целостность данных.

# 4 Описание процедур импорта и экспорта

Для таблицы adress в базе данных доступна функция экспорта и импорта данных в формате xml. Это может пригодиться, если необходимо переместить данные на другой сервер или создать резервную копию.

Для реализации экспорта данных в xml, была разработана функция, результатом которой стало создание xml файла, с записанными строками из определенной таблицы в xml формат. Скрипт процедуры представлен в листинге 4.1.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GenerateAgencyXML IS  xml\_data CLOB;  BEGIN  SELECT XMLELEMENT("Agency",  XMLFOREST(AgencyID, AgentID, Name, Address, Email, Phone)  ).getClobVal()  INTO xml\_data  FROM Agency;  DBMS\_OUTPUT.PUT\_LINE(xml\_data);  END; |

Листинг 4.1 – процедура GenerateAgencyXML

Для обратного экспорта данные в xml формат из файла, реализован запросом, который обращается к xml, и выбирает данные в обычные строки, это можно реализовать с помощью функции, либо по мере надобности записать данные во временную таблицу, либо таблицу в другой базе данных. Скрипт запроса представлен в [приложении Г](#_Приложение_Г).

Таким образом, пользователи базы данных могут легко экспортировать и импортировать данные в форматах xml, что делает управление базой данных более удобным и эффективным.

# 5 Тестирование производительности

Одной из ключевых задач в процессе разработки является тестирование производительности базы данных. Такое тестирование позволяет оценить способность базы данных обрабатывать запросы и возвращать результаты с высокой скоростью. Оценка производительности важна для определения эффективности базы данных и выявления возможных узких мест, которые могут вызывать задержки или проблемы в работе приложения.

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 (листинг 5.1).

|  |
| --- |
| --- тестирование 10000 строк  CREATE SEQUENCE object\_id\_seq  START WITH 1  INCREMENT BY 1  NOCACHE;  CREATE OR REPLACE PROCEDURE test\_insert\_performance AS  v\_object\_id INT;  BEGIN  FOR i IN 1..10000 LOOP  ---Получаем следующее значение последовательности  ObjectID  SELECT object\_id\_seq.NEXTVAL INTO v\_object\_id FROM DUAL;  INSERT INTO Object (ObjectID, ObjectType, Addres, Area, NumRooms, Price)  VALUES (v\_object\_id, 'Тип объекта ' || i, 'Адрес объекта ' || i, i \* 10, i, i \* 1000);  END LOOP;  END test\_insert\_performance;  BEGIN  test\_insert\_performance;  END; |

Листинг 5.1 – Заполнение таблицы 10000 строк

Для получения выборки данных использовался запрос, который представлен на листинге 5.1.

|  |
| --- |
| select Area from Object where Price like '3%'; |

Листинг 5.2 – Запрос к таблице Object

Результаты выполнения запроса к таблице указывают на значительные затраты времени и ресурсов, особенно при сканировании всей таблицы и применении фильтра. Время планирования – 2,061 seconds. Результаты запроса будут представлены на рисунке 5.1.

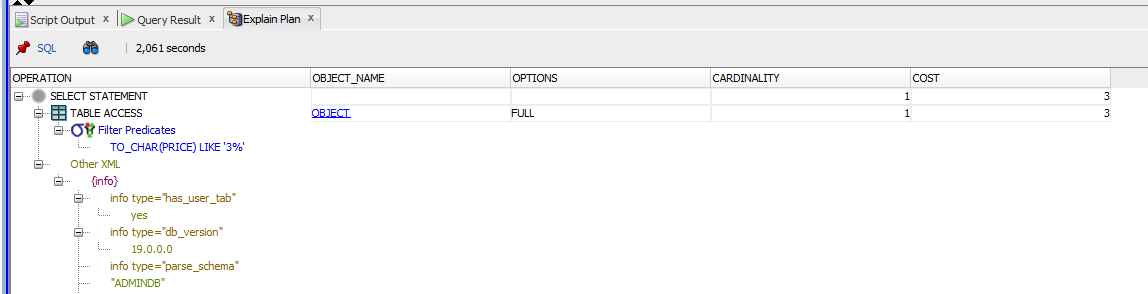


Рисунок 5.1 – Результат выполнения запроса

Для ускорения данного процесса создадим индекс на поле Наименование так как именно по этому полю выполняется фильтрация. После создания индекса, можно повторить запрос и сравнить стоимость с предыдущим запросом. Результат будет представлен на рисунке 5.2.

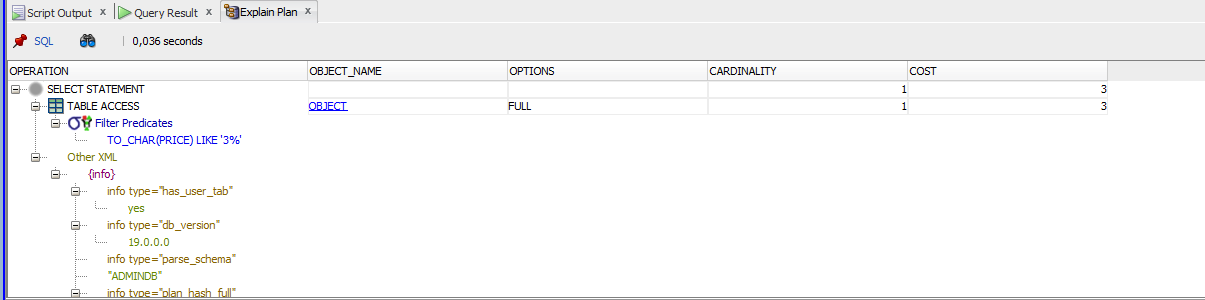


Рисунок 5.2 – Результат выполнения запроса

После создания индексов видно, что лоты стали генерировать намного быстрее. Время планирования заняло 0.036 seconds.

Результаты тестирования говорят о том, что создание индексов на полях, по которым выполняются частые запросы, может значительно повысить производительность базы данных.

# 6 Описание технологии и ее применение в базе данных

Резервное копирование и восстановление в базах данных представляет собой систему методов и инструментов, предназначенных для сохранения и восстановления данных в случае их потери или повреждения. В данной базе данных используется технология резервного копирования для обеспечения надежности и безопасности хранения информации.

На рисунке 6.1 иллюстрируется процесс резервного копирования данных в базе.

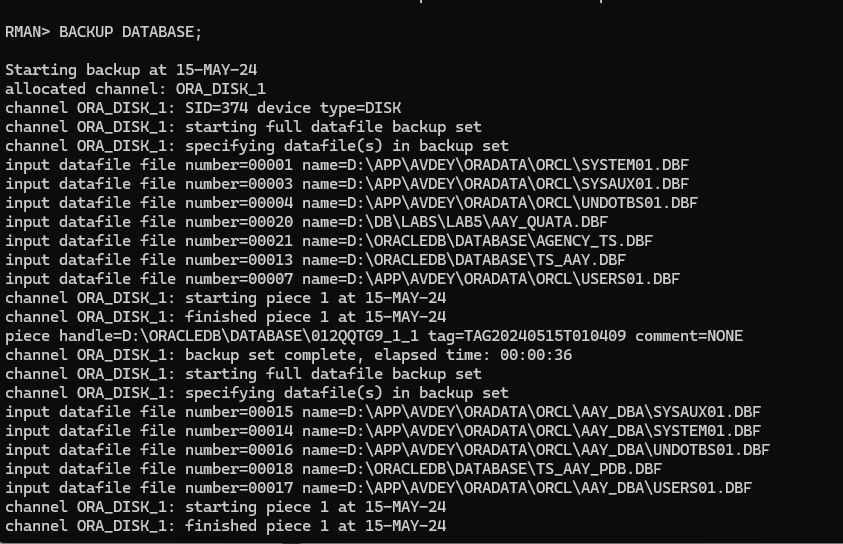


Рисунок 6.1 – Процесс резервного копирования данных

Применение резервного копирования и восстановления обладает несколькими ключевыми преимуществами:

* 1. Сохранение целостности данных. Регулярные резервные копии помогают предотвратить потерю информации в случае аварийных ситуаций, таких как сбои оборудования или программного обеспечения.
  2. Обеспечение безопасности. Резервные копии данных могут служить важным инструментом в защите от угроз безопасности, таких как вирусы или злонамеренные атаки, позволяя быстро восстановить информацию после инцидента.
  3. Восстановление в случае необходимости. Благодаря наличию резервных копий базы данных возможно оперативное восстановление данных до момента их утраты или повреждения.
  4. Эффективное управление данными. Возможность регулярного создания резервных копий и их последующего использования дает возможность контролировать и управлять данными с большей уверенностью.

Технология резервного копирования позволяет обеспечить сохранность и доступность данных, играя ключевую роль в обеспечении непрерывности работы базы данных и предоставлении средств для восстановления информации в случае необходимости.

# 7 Руководство пользователя

В данном проекте была поставлена задача разработки базы данных для Агенства Недвижимости с использованием технологии резервного копирования и восстановления в СУБД oracle 19c.

В ходе выполнения проекта были использованы различные объекты, такие как таблицы, представления, процедуры, индексы триггеры, специальные функции, чтобы обеспечить структурированное хранение данных, изолированность данных таблиц от прямого вмешательства и удобный доступ к ним. В итоге, поставленная цель была успешно достигнута, и на выходе получена готовая база данных.

В рамках тестирования базы данных был использован поток данных, и результаты тестирования оказались положительными. Также были разработаны процедуры импорта и экспорта данных в формате xml, что значительно облегчило управление данными и повысило удобство использования базы данных.

Разработанная база данных для интернет-магазина компьютерных комплектующих имеет ряд важных особенностей, которые позволяют эффективно управлять ей. Одной из главных особенностей является технология резервного копирования и восстановления, которая обеспечивает сохранность данных в случае любых сбоев или ошибок. Это означает, что все данные интернет-магазина, могут быть быстро восстановлены, минимизируя потерю информации и обеспечивая непрерывность работы площадки.

Кроме того, база данных обладает гибкой системой привилегий, которая позволяет разграничивать доступ к различным функциям и данным в зависимости от роли пользователя. Это обеспечивает безопасность данных и предотвращает несанкционированный доступ.

Делая вывод, можно сказать, что поставленные цели были выполнены. интернет-магазина компьютерных комплектующих выполняет необходимую функцию хранения и управления данными.

# Заключение

В данном проекте была поставлена задача разработки базы данных для Агенства Недвижимости с использованием технологии резервного копирования и восстановления в СУБД oracle 19c.

В ходе выполнения проекта были использованы различные объекты, такие как таблицы, представления, процедуры, индексы триггеры, специальные функции, чтобы обеспечить структурированное хранение данных, изолированность данных таблиц от прямого вмешательства и удобный доступ к ним. В итоге, поставленная цель была успешно достигнута, и на выходе получена готовая база данных.

В рамках тестирования базы данных был использован поток данных, и результаты тестирования оказались положительными. Также были разработаны процедуры импорта и экспорта данных в формате xml, что значительно облегчило управление данными и повысило удобство использования базы данных.

Разработанная база данных для интернет-магазина компьютерных комплектующих имеет ряд важных особенностей, которые позволяют эффективно управлять ей. Одной из главных особенностей является технология резервного копирования и восстановления, которая обеспечивает сохранность данных в случае любых сбоев или ошибок. Это означает, что все данные агенства, могут быть быстро восстановлены, минимизируя потерю информации и обеспечивая непрерывность работы площадки.

Кроме того, база данных обладает гибкой системой привилегий, которая позволяет разграничивать доступ к различным функциям и данным в зависимости от роли пользователя. Это обеспечивает безопасность данных и предотвращает несанкционированный доступ.

# Список используемых источников

1. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс] – https://habr.com/ru/articles/566218/ – Дата доступа 20.04.2024
2. IBM [Электронный ресурс] – https://www.ibm.com/docs/ru/mfci/7.6.2?topic=structure-views – Дата доступа 21.04.2024
3. oracleplsql [Электронный ресурс] – https://oracleplsql.ru/procedures.html – Дата доступа 03.05.2024
4. Администрирование баз данных Oracle [Электронный ресурс] – <https://oracle-dba.ru/database/backup-and-restore/rman/about-oracle-rman/> – Дата доступа 03.05.2024

# Приложение А. Листинг создания таблиц

|  |
| --- |
| create table Object (  ObjectID int primary key,  ObjectType varchar(255),  Addres VARCHAR(255),  Area decimal(10,2),  NumRooms int,  Price decimal(12,2)  );  create table Agency (  AgencyID INT PRIMARY KEY,  AgentID INT,  Name VARCHAR(255),  Address VARCHAR(255),  Email VARCHAR(255),  Phone VARCHAR(20),  foreign key (AgentID) references Agents(AgentID)  );  create table Client (  ClientID int primary key,  ObjectID int,  UserID int,  foreign key (UserID) references DataUsers(UserID),  foreign key (ObjectID) references Object(ObjectID)  );  create table Buyers (  BuyerID int primary key,  UserID int,  foreign key (UserID) references DataUsers(UserID)  );  create table Agents (  AgentID int primary key,  UserID int,  foreign key (UserID) references DataUsers(UserID)  );  create table Deal (  DealID int primary key,  AgentID int,  AgencyID int,  ClientID int,  ObjectID int,  BuyerID int,  DealDate date,  Amount decimal(12,2),  foreign key (AgentID) references Agents(AgentID),  foreign key (AgencyID) references Agency(AgencyID),  foreign key (ClientID) references Client(ClientID),  foreign key (ObjectID) references Object(ObjectID),  foreign key (BuyerID) references Buyers(BuyerID)  );  create table DataUsers(  UserID int primary key,  Login varchar(10),  Password varchar(10),  Name VARCHAR(255),  LastName VARCHAR(100),  Email VARCHAR(255),  Phone VARCHAR(20)  ); |

# Приложение Б. Листинг процедур базы данных

|  |
| --- |
| CREATE OR REPLACE PROCEDURE create\_client (  p\_Login VARCHAR2,  p\_Password VARCHAR2,  p\_Name VARCHAR2,  p\_LastName VARCHAR2,  p\_Email VARCHAR2,  p\_Phone VARCHAR2  )  IS  v\_UserID INT;  BEGIN  DECLARE  v\_ClientID INT;  BEGIN  SELECT client\_id\_sequence.NEXTVAL INTO v\_ClientID FROM DUAL;  INSERT INTO DataUsers (UserID, Login, Password, Name, LastName, Email, Phone)  VALUES (v\_ClientID, p\_Login, p\_Password, p\_Name, p\_LastName, p\_Email, p\_Phone);  INSERT INTO Client (ClientID, ObjectID, UserID)  VALUES (v\_ClientID, v\_ClientID, v\_ClientID);  DBMS\_OUTPUT.PUT\_LINE('Данные успешно добавлены в таблицы DataUsers и Client.');  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Нарушение уникального ограничения. Дублирующиеся значения.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка ' || SQLCODE || ': ' || SQLERRM);  END;  END;  CREATE OR REPLACE PROCEDURE GenerateAgencyXML IS  xml\_data CLOB;  BEGIN  SELECT XMLELEMENT("Agency",  XMLFOREST(AgencyID, AgentID, Name, Address, Email, Phone)  ).getClobVal()  INTO xml\_data  FROM Agency;  DBMS\_OUTPUT.PUT\_LINE(xml\_data);  END;  CREATE OR REPLACE PROCEDURE create\_deal (  p\_ClientID INT,  p\_ObjectID INT,  p\_BuyerID INT,  p\_DealDate DATE,  p\_Amount DECIMAL  ) IS  BEGIN  DECLARE  v\_DealID INT;  v\_CalculatedAmount DECIMAL;  BEGIN  SELECT seq\_deal\_id.NEXTVAL INTO v\_DealID FROM DUAL;    v\_CalculatedAmount := calculate\_vat(p\_Amount);    -- Вставка данных в таблицу Deal  INSERT INTO Deal (DealID, ClientID, ObjectID, BuyerID, DealDate, Amount)  VALUES (v\_DealID, p\_ClientID, p\_ObjectID, p\_BuyerID, p\_DealDate, v\_CalculatedAmount);    -- Output a success message  DBMS\_OUTPUT.PUT\_LINE('Данные успешно добавлены в таблицу Deal.');  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Нарушение уникального ограничения. Дублирующиеся значения.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка ' || SQLCODE || ': ' || SQLERRM);  END;  END;  --- заполнение таблицы объектов  CREATE OR REPLACE PROCEDURE create\_object (p\_ObjectType VARCHAR2, p\_Addres VARCHAR2, p\_Area DECIMAL, p\_NumRooms INT, p\_Price DECIMAL)  IS  v\_ObjectID INT;  v\_ClientID INT;  BEGIN  SELECT object\_id\_sequence.NEXTVAL INTO v\_ObjectID FROM DUAL;  SELECT object\_id\_sequence.NEXTVAL INTO v\_ClientID FROM DUAL;    -- Вставляем данные в таблицу Object  INSERT INTO Object (ObjectID, ObjectType, Addres, Area, NumRooms, Price)  VALUES (v\_ObjectID, p\_ObjectType, p\_Addres, p\_Area, p\_NumRooms, p\_Price);    -- Вставляем данные в таблицу Client  INSERT INTO Client (ClientID, ObjectID, UserID)  VALUES (v\_ClientID, v\_ObjectID, v\_ObjectID);    -- Выводим сообщение об успешной вставке  DBMS\_OUTPUT.PUT\_LINE('Данные успешно добавлены в таблицы Object и Client.');  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Нарушение уникального ограничения. Дублирующиеся значения.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка ' || SQLCODE || ': ' || SQLERRM);  END;  --- создание покупателя  CREATE OR REPLACE PROCEDURE create\_buyers (  p\_Login IN VARCHAR2,  p\_Password IN VARCHAR2,  p\_Name IN VARCHAR2,  p\_LastName IN VARCHAR2,  p\_Email IN VARCHAR2,  p\_Phone IN VARCHAR2  )  IS  v\_BuyerID INT;  BEGIN  -- Генерируем следующее значение для BuyerID из последовательности  DECLARE  v\_BuyerID INT;  BEGIN  SELECT buyer\_sequence.NEXTVAL INTO v\_BuyerID FROM DUAL;    -- Вставляем данные в таблицу DataUsers  INSERT INTO DataUsers (UserID, Login, Password, Name, LastName, Email, Phone)  VALUES (v\_BuyerID, p\_Login, p\_Password, p\_Name, p\_LastName, p\_Email, p\_Phone);    -- Вставляем данные в таблицу Buyers  INSERT INTO Buyers (BuyerID, UserID)  VALUES (v\_BuyerID, v\_BuyerID);    -- Выводим сообщение об успешной вставке  DBMS\_OUTPUT.PUT\_LINE('Данные успешно добавлены в таблицы DataUsers и Client.');  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Нарушение уникального ограничения. Дублирующиеся значения.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка ' || SQLCODE || ': ' || SQLERRM);  END;  END;  CREATE OR REPLACE PROCEDURE test\_insert\_performance AS  v\_object\_id INT;  BEGIN  FOR i IN 1..10000 LOOP  -- Получаем следующее значение последовательности для ObjectID  SELECT object\_id\_seq.NEXTVAL INTO v\_object\_id FROM DUAL;    INSERT INTO Object (ObjectID, ObjectType, Addres, Area, NumRooms, Price)  VALUES (v\_object\_id, 'Тип объекта ' || i, 'Адрес объекта ' || i, i \* 10, i, i \* 1000);  END LOOP;  END test\_insert\_performance;  BEGIN  test\_insert\_performance;  END; |

# Приложение В. Листинг функций базы данных

|  |
| --- |
| CREATE OR REPLACE FUNCTION calculate\_vat (  p\_amount DECIMAL  )  RETURN DECIMAL  IS  v\_vat\_rate DECIMAL := 1.2;  v\_vat DECIMAL;  BEGIN  v\_vat := p\_amount \* v\_vat\_rate;  RETURN v\_vat;  END calculate\_vat; |

# Приложение Г. Листинг представлений базы данных

|  |
| --- |
| --Client\_View  CREATE OR REPLACE VIEW Client\_View AS  SELECT Client.ClientID, Client.UserID, DataUsers.Name, DataUsers.LastName, DataUsers.Email, DataUsers.Phone  FROM Client  INNER JOIN DataUsers ON Client.UserID = DataUsers.UserID;  --- сделки Deal  CREATE OR REPLACE VIEW Deal\_View AS  SELECT D.DealID, O.ObjectID , O.ObjectType AS "Объект", D.DealDate AS "Дата Сделки", D.Amount AS "Сумма", DU.Name AS "Имя Покупателя", DU.LastName AS "Фамилия Покупателя", DU.Phone AS "Номер Телефона"  FROM Deal D  INNER JOIN Buyers B ON B.BuyerID = D.BuyerID  INNER JOIN DataUsers DU ON DU.UserID = B.UserID  INNER JOIN Object O ON O.ObjectID = D.ObjectID;  --все объекты и клиенты  CREATE OR REPLACE VIEW Object\_View AS  select ObjectType as Type, Addres, Area, NumRooms, Price, Name, LastName, Email, Phone  from Object  inner join Client on Object.ObjectID = Client.ObjectID  inner join DataUsers on Client.UserID = DataUsers.UserID;  --- объекты  CREATE OR REPLACE VIEW All\_Objects\_View AS  select \* from Object;  --- все покупатели  CREATE OR REPLACE VIEW Buyers\_View AS  SELECT Buyers.BuyerID, Buyers.UserID, DataUsers.Name, DataUsers.LastName, DataUsers.Email, DataUsers.Phone  FROM Buyers  INNER JOIN DataUsers ON Buyers.UserID = DataUsers.UserID; |

# Приложение Д. Листинг кода создания пользователей

|  |
| --- |
| create user Agent  identified by Agent  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Agent  account unlock  password expire;  create user Buyers  identified by Buyers  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Buyers  account unlock  password expire;  create user Client  identified by Client  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_Client  account unlock  password expire;  create user AdminDB  identified by Admin  default tablespace agency\_ts  temporary tablespace agency\_ts\_tmp  profile pf\_AdminDB  account unlock  password expire;  ALTER USER AdminDB IDENTIFIED BY Admin1;  create role AdminDB\_role;  grant  create session,  create table,  create view,  create procedure,  create user,  create role,  create profile  to AdminDB\_role with admin option;  grant dba to AdminDB;  CREATE PROFILE pf\_AdminDB LIMIT  PASSWORD\_LIFE\_TIME 640  SESSIONS\_PER\_USER 200  FAILED\_LOGIN\_ATTEMPTS 7  PASSWORD\_LOCK\_TIME 1  PASSWORD\_REUSE\_TIME 10  PASSWORD\_GRACE\_TIME DEFAULT  CONNECT\_TIME 180  IDLE\_TIME 30;  CREATE PROFILE pf\_Agent LIMIT  PASSWORD\_LIFE\_TIME 640  SESSIONS\_PER\_USER 200  FAILED\_LOGIN\_ATTEMPTS 7  PASSWORD\_LOCK\_TIME 1  PASSWORD\_REUSE\_TIME 10  PASSWORD\_GRACE\_TIME DEFAULT  CONNECT\_TIME 180  IDLE\_TIME 30;  CREATE PROFILE pf\_Buyers LIMIT  PASSWORD\_LIFE\_TIME 640  SESSIONS\_PER\_USER 200  FAILED\_LOGIN\_ATTEMPTS 7  PASSWORD\_LOCK\_TIME 1  PASSWORD\_REUSE\_TIME 10  PASSWORD\_GRACE\_TIME DEFAULT  CONNECT\_TIME 180  IDLE\_TIME 30;  CREATE PROFILE pf\_Client LIMIT  PASSWORD\_LIFE\_TIME 640  SESSIONS\_PER\_USER 200  FAILED\_LOGIN\_ATTEMPTS 7  PASSWORD\_LOCK\_TIME 1  PASSWORD\_REUSE\_TIME 10  PASSWORD\_GRACE\_TIME DEFAULT  CONNECT\_TIME 180  IDLE\_TIME 30;  create role Agent\_role;  create role Buyers\_role;  create role Client\_role;  grant  create session,  create table,  create view  to Agent\_role;  grant select on Object\_View TO Agent\_role;  grant select on Buyers\_View TO Agent\_role;  grant select on All\_Objects\_View to Agent\_role;  grant select on Client\_View to Agent\_role;  grant select on Deal\_View to Agent\_role;  grant execute on create\_object to Agent\_role;  grant execute on create\_client to Agent\_role;  grant execute on create\_buyers to Agent\_role;  grant execute on test\_insert\_performance to Agent\_role;  grant execute on create\_deal to Agent\_role;  grant execute on GenerateAgencyXML to Agent\_role;  grant  create session  to Buyers\_role;  grant  create session  to Client\_role; |

# Приложение Е. Листинг процедур для импорта и экспорта в xml

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GenerateAgencyXML IS  xml\_data CLOB;  BEGIN  SELECT XMLELEMENT("Agency",  XMLFOREST(AgencyID, AgentID, Name, Address, Email, Phone)  ).getClobVal()  INTO xml\_data  FROM Agency;  DBMS\_OUTPUT.PUT\_LINE(xml\_data);  END; |