

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 базы данных игрового магазина с применением технологии
миграции данных и объектов из одной СУБД в другую»

Выполнил студент Климович Антон Сергеевич
(Ф.И.О.)

Руководитель работы ассист. Нистюк Ольга Александровна
(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
1 Постановка задачи	4
1.1 Веб-сайт «Gabe Store »	4
1.2 Веб-сайт «crxr.ru»	5
1.3 Определение цели и задачи	5
2 Проектирование базы данных	7
2.1 Вывод по проектированию модели базы данных	7
3 Разработка объектов базы данных	8
3.1 Разработка таблиц базы данных	8
3.2 Процедуры	10
3.3 Вывод по разработке объектов базы данных	10
4 Описание процедур импорта и экспорта	11
4.1 Описание процедуры экспорта данных	11
4.2 Описание процедуры импорта данных	11
5 Тестирование производительности	13
6 Описание технологии и ее применения в базе данных	15
7 Краткое описание веб-сайта для демонстрации	17
8 Руководство пользователя	18
Заключение	21
Список используемых источников	22
ПРИЛОЖЕНИЕ А	23
ПРИЛОЖЕНИЕ Б	25
ПРИЛОЖЕНИЕ В	29

Введение

В наше время, когда технологии становятся неотъемлемой частью нашей повседневной жизни, разработка новаторских систем управления выходит на передний план. Одной из сфер, где внедрение технологий может существенно улучшить опыт пользователей, является сегмент магазинов видеоигр. В свете постоянного роста популярности видеоигр возникает потребность в эффективных и современных инструментах для управления продуктами, предлагаемыми магазинами.

Цифровизация и автоматизация процессов в магазинах видеоигр могут привести к значительному улучшению взаимодействия с покупателями и оптимизации управления ассортиментом. Основными задачами в рамках данного проекта будут:

- Управление каталогом продуктов (добавление, редактирование, удаление);
- Обеспечение безопасности и конфиденциальности данных о пользователях;
- Определение ролей сотрудников магазина (администратор, пользователь).

Особое внимание в проекте будет уделяться безопасности личных данных покупателей, а также разработке интуитивно понятного и удобного интерфейса для облегчения взаимодействия с магазином. Цель - создать инновационную информационную систему, которая не только соответствует высоким требованиям современных потребителей, но и обеспечивает комфортный опыт при выборе и приобретении видеоигр.

1 Постановка задачи

Важным этапом в процессе разработки нового продукта, такого как магазин видеоигр, является анализ существующих решений в данной области. Этот этап позволяет выявить, какие инструменты и подходы уже используются в индустрии, выделить их сильные и слабые стороны, а также определить потенциальные направления для усовершенствований. В данном обзоре проведен анализ наиболее популярных решений реализации магазинов видеоигр.

1.1 Веб-сайт «Gabe Store»

Одним из альтернативных решений для магазина видеоигр является интернет-ресурс "Gabe Store". На этом веб-сайте можно найти широкий ассортимент видеоигр для различных платформ, таких как PC, PlayStation, Xbox и другие. Каталог игр на сайте поделен на несколько категорий: экшн, приключения, стратегии, ролевые игры и другие жанры, позволяя покупателям легко находить интересующие их игры.

На рисунке 1.1 предствлен скриншот веб-приложения «Gabe Store».

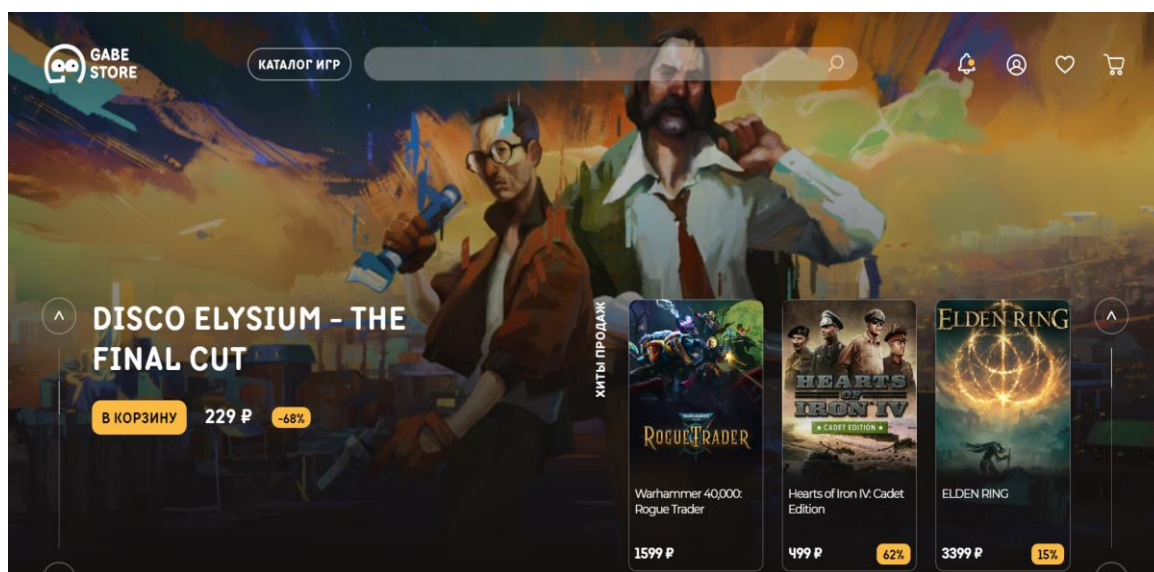


Рисунок 1.1 – Скриншот аналога веб-сайта «Gabe Store»

Один из наиболее популярных веб-сайтов для покупки игр.

Удобство использования: Интуитивно понятный интерфейс, легкая навигация и удобные функции поиска игр являются важными аспектами для обеспечения удобства использования магазина. Пользователи должны легко находить нужные игры, просматривать информацию о них и осуществлять покупки без лишних сложностей.

Мультиплатформенность: Предоставление доступа к играм на различных платформах (компьютеры, консоли, мобильные устройства) позволяет привлечь больше игроков и обеспечить им возможность играть на предпочитаемой платформе.

Технические проблемы: неполадки в работе магазина, такие как сбои сервера, медленная загрузка страниц, проблемы с оплатой или скачиванием игр, могут создать негативный опыт для пользователей и ухудшить их впечатление о магазине.

Сервис предлагает базовый функционал, который включает в себя возможность добавления товара в корзину, его удаление и оформление заказа. Тем не менее, следует учесть, что у данного сервиса отсутствуют расширенные возможности для организации взаимодействия между пользователями, в частности, функционал для написания отзывов о товарах отсутствует. Внедрение таких возможностей могло бы способствовать более эффективной обратной связи со стороны пользователей и повысить уровень их удовлетворенности.

1.2 Веб-сайт «urxpr.ru»

На рисунке 1.2 представлен скриншот веб-сайта «urxpr.ru».

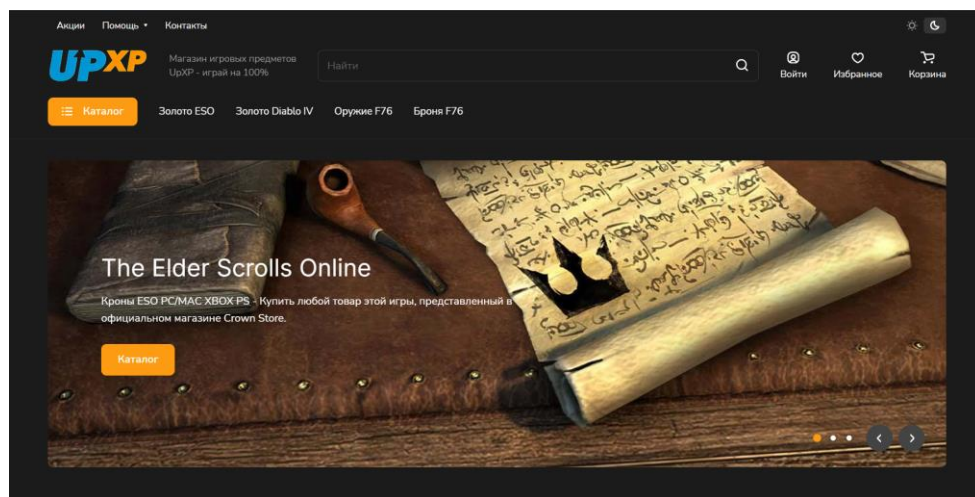


Рисунок 1.2 – Скриншот аналога веб-сайта «urxpr.ru»

Простой и безопасный сайт для покупки видеоигр с возможностями покупки игровых предметов. Тем не менее, сайт имеет ограниченный каталог.

- Бесплатность: предоставляет базовые функции без оплаты.
- Ограниченный функционал: отсутствует возможность управления доступом или отслеживания активности кода.

Для приложения важным аспектом является удобство и скорость работы, а текущая модель базы данных обеспечивает эти требования, что позволяет комфортно взаимодействовать с сайтом. Данный веб-сайт предлагает такие возможности. Из этого следует, что данный веб-сайт удобный и быстрый, что дает пользователям возможность быстро и без потери времени взаимодействовать с данным веб-сайтом.

1.3 Определение цели и задачи

На текущий момент рынок веб-сайтов для продажи видеоигр предоставляет множество решений с различным уровнем функционала. Однако, многие из них не могут предложить полноценного комплекса инструментов, сочетающего в себе удобство, гибкость и высокие стандарты безопасности, что оставляет простор для новых разработок в этой сфере.

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

Должны быть выполнены следующие требования:

- база данных должна быть реализована в СУБД PostgreSQL.
- доступ к данным должен осуществляться только через соответствующие процедуры;
- должен быть проведен импорт данных из JSON файлов, экспорт данных в формат JSON;
- необходимо протестировать производительность базы данных на таблице, содержащей не менее 100000 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.

Технологии: PostgreSQL, NodeJS обеспечили надежность и гибкость веб-сайта. Таким образом, выбор технологий был тщательно обоснован на основе анализа аналогов и требований проекта.

2 Проектирование базы данных

2.1 Вывод по проектированию модели базы данных

Архитектура базы данных в данном проекте реализована в третьей нормальной форме. Такой подход обеспечивает ясное разделение ответственности между различными сущностями и позволяет оптимизировать процесс работы приложения.

Диаграмма базы данных, спроектированной в ходе разработки, представлена на рисунке 2.1.

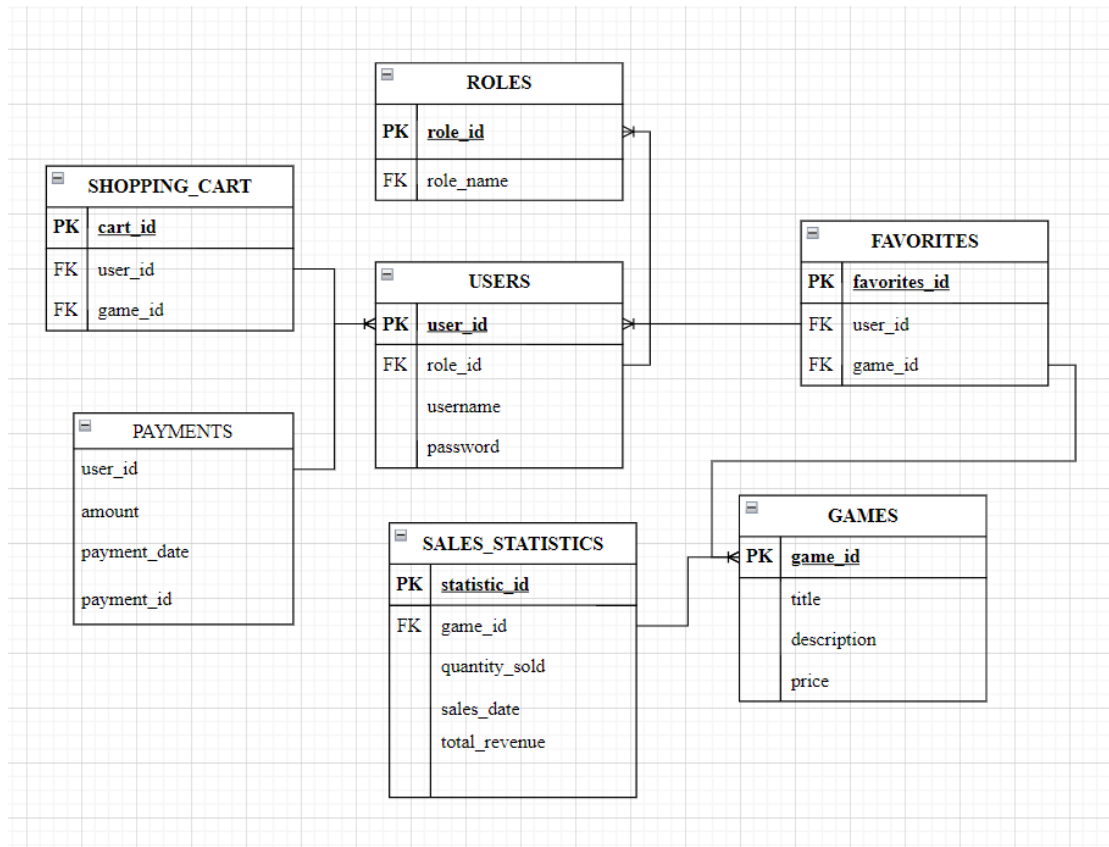


Рисунок 2.1 – Диаграмма базы данных

Логическая структура таблиц позволяет быстро и легко получать нужную информацию, а также расширять функциональность системы без пересмотра всей архитектуры, что позволит ускорить масштабирование системы.

Диаграмма вариантов использования для пользователя, администратора, гостя и создателя взаимодействия с играми представлены в приложении Г

3 Разработка объектов базы данных

3.1 Разработка таблиц базы данных

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

В базе данных представлены 7 таблиц: USERS, ROLES, SHOPPING_CART, PAYMENTS, FAVORITES, GAMES, SALES_STATISTICS.

Таблица USERS хранит информацию пользователей. Состоит из следующих столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы USERS

Наименование	Описание	Тип
USER_ID	Уникальный идентификатор пользователя, первичный ключ	INTEGER
ROLE_ID	Идентификатор роли, внешний ключ	INTEGER
USERNAME	Имя пользователя	VARCHAR(50)
PASSWORD	Пароль пользователя	VARCHAR(255)

Таблица SALES_STATISTICS содержит информацию о статистике продаж. Состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы SALES_STATISTICS

Наименование	Описание	Тип
STATISTIC_ID	Уникальный идентификатор статистики продаж, первичный ключ	INTEGER
GAME_ID	Уникальный идентификатор игры, внешний ключ	INTEGER
QUANTITY_SOLD	Количество проданных товаров	INTEGER
TOTAL_REVENUE	Общая сумма всех проданных игр	NUMERIC(10,2)
SALES_DATE	Дата последней покупки	TIMESTAMP

Таблица ROLE хранит роли пользователей (таблица 3.3):

Таблица 3.3 – Столбцы таблицы ROLE

Наименование	Описание	Тип
ROLE_ID	Уникальный идентификатор роли, первичный ключ	INTEGER
ROLE_NAME	Наименование роли	VARCHAR(50)

Таблица SHOPPING_CART содержит информацию о товарах в корзине. Состоит из следующих столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы SHOPPING_CART

Наименование	Описание	Тип
CART_ID	Уникальный идентификатор корзины, первичный ключ	INTEGER
GAME_ID	Уникальный идентификатор игры	INTEGER
USER_ID	Уникальный идентификатор пользователя, внешний ключ	INTEGER

Таблица PAYMENTS содержит информацию об оплаченных товарах. Состоит из следующих столбцов (таблица 3.5):

Таблица 3.5 – Столбцы таблицы PAYMENTS

Наименование	Описание	Тип
USER_ID	Уникальный идентификатор пользователя	INTEGER
AMOUNT	Количество проданных товаров	NUMERIC(10,2)
PAYMENT_DATE	Дата последней покупки	TIMESTAMP
PAYMENT_ID	Уникальный идентификатор оплаты, первичный ключ	INTEGER

Таблица FAVORITES о товарах в избранном. Состоит из следующих столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы FAVORITES

Наименование	Описание	Тип
USER_ID	Уникальный идентификатор пользователя, внешний ключ	INTEGER
GAME_ID	Уникальный идентификатор игры, внешний ключ	INTEGER
FAVORITES_ID	Уникальный идентификатор избранного, первичный ключ	INTEGER

Таблица GAMES содержит информацию об играх. Состоит из следующих столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы GAMES

Наименование	Описание	Тип
GAME_ID	Уникальный идентификатор игры, первичный ключ	INTEGER
TITLE	Название игры	VARCHAR(255)
DESCRIPTION	Описание игры	TEXT
PRICE	Цена игры	NUMERIC(10,2)

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

3.2 Процедуры

Процедуры, представленные ниже, обеспечивают базовые операции управления данными в базе данных, такие как вставка, обновление и удаление. Эти процедуры охватывают ключевые аспекты системы, включая управление пользователями, корзиной, списком избранного.

Листинг PL/SQL-кода для создания процедур находится в приложении Б.

Процедуры, разработанные в рамках курсового проекта:

- add_game – Добавление игры в каталог;
- add_to_cart – добавление игры в корзину;
- authenticate_user – аутентификация пользователя;
- update_game_by_id – обновление карточки с игрой используя game_id;
- import_from_json – импортирует данные из json файла;
- export_game_to_json – экспортирует данные в json файл;
- delete_game – процедура удаляет игру из общего списка игр;
- get_all_games – выводит весь список игр из таблицы games;
- get_game_with_pagination – выводит игры с учетом пагинации;
- register_user – регистрирует пользователя;
- update_sales_statistics – обновляет данные на странице статистики;
- search_games – осуществляет поиск игры по названию или описанию;
- get_total_page_count – делает подсчет страниц с элементами, для упрощения использования пагинации;

3.3 Вывод по разработке объектов базы данных

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 быстрое действие и целостность данных.

4 Описание процедур импорта и экспорта

В данной курсовой работе реализованы процедуры экспорта и импорта данных из JSON файла в базу данных таблицы Games и наоборот. При данных операциях работает с файловой системой приложение, а разбором и генерацией JSON занимается Data Grip.

4.1 Описание процедуры экспорта данных

В представленном блоке кода описаны две основные процедуры: `export_to_json` и `import_from_json`. Эти процедуры позволяют экспортировать и импортировать данные из таблицы Games в формат JSON.

Код процедура импорта данных в формат JSON представлена в листинге 4.1.

```
CREATE OR REPLACE FUNCTION export_games_to_json(p_export_path text)
RETURNS VOID
LANGUAGE plpgsql
AS $$
DECLARE
    json_text text;
BEGIN
    SELECT json_agg(row_to_json(games))::text INTO json_text FROM
games;
    EXECUTE 'COPY (SELECT ' || quote_literal(json_text) || '::text)
TO ''' || p_export_path || ''';
END;
$$;
```

Листинг 4.1 – Процедура импорта данных из JSON в таблицу Users

Процедура `export_game_to_json`: принимает один параметр `p_export_path`, который является именем файла, в который будут экспортироваться данные. Данные из таблицы GAMES выбираются и преобразуются в формат JSON. Каждая строка данных добавляется в файл с учетом правильного форматирования JSON. После завершения экспорта файла закрывается. Процедура завершается вызовом `export_game_to_json`, что инициирует процесс экспорта.

4.2 Описание процедуры импорта данных

Процедура `import_from_json`: принимает один параметр `p_file_path`, который является путем к файлу, из которого будут импортироваться данные. Данные из таблицы GAMES выбираются и преобразуются в формат JSON. Каждая строка данных добавляется в файл с учетом правильного форматирования JSON. Код процедуры импорта представлена в листинге 4.2.

```
CREATE PROCEDURE import_from_json(p_file_path text)
LANGUAGE plpgsql
AS $$
```

```
BEGIN
  CREATE TABLE IF NOT EXISTS temp (data json);
  COPY temp FROM 'p_file_path';

  CREATE TABLE IF NOT EXISTS games (
    game_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL
  );

  INSERT INTO games (title, description, price)
  SELECT
    value->>'title' as title,
    value->>'description' as description,
    (value->>'price')::numeric(10, 2) as price
  FROM temp, json_array_elements(data) as value;

  DROP TABLE IF EXISTS temp;
END;
$$;
```

Листинг 4.2 – Процедура экспорта

5 Тестирование производительности

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

Для данной задачи мы создали анонимный блок, и так как работаем именно с действующей базой данных, то можем использовать готовую процедуру для вставки данных в таблицу. Разработанный анонимный блок позволяет добавить большое количество строк за одно выполнение (рисунок 5.1).

```
DO $$
DECLARE
    i INTEGER;
BEGIN
    FOR i IN 1..100000 LOOP
        INSERT INTO games (title, description, price)
        VALUES
            ('Game ' || i,
             'Description for Game ' || i,
             (random() * 100)::numeric(10, 2));
    END LOOP;
END $$;
```

Рисунок 5.1 –Заполнение таблицы большим количеством данных;

Ниже на рисунке 5.2 представлен результат статистического анализа и компиляции DataGrip до изменения таблицы

```
pub_stark_public> DO $$
    DECLARE
        i INTEGER;
    BEGIN
        FOR i IN 1..100000 LOOP
            INSERT INTO games (title, description, price)
            VALUES
                ('Game ' || i,
                 'Description for Game ' || i,
                 (random() * 100)::numeric(10, 2));
        END LOOP;
    END $$
[2023-12-20 23:31:59] completed in 5 s 571 ms
```

Рисунок 5.2 – Статический анализ перед изменением таблицы

Как можно увидеть, затраченное время достаточно мало, поэтому дополнительные индексы не требуются.

Для того чтобы правильно организовать процесс тестирования БД, тестировщики должны обладать хорошими знаниями SQL и DML и иметь ясное представление о внутренней структуре БД. Это самый лучший и надежный способ тестирования БД особенно для приложений с низким и средним уровнем сложности. Данный метод не только дает уверенность, что тестирование выполнено качественно, но также повышает мастерство написания SQL-запросов.

В итоге, использование данных индексов позволяет значительно сократить время поиска данных, улучшить производительность запросов и эффективность

использования ресурсов базы данных. Это приводит к более отзывчивой системе и повышению общей производительности.

6 Описание технологии и ее применения в базе данных

сказывается Перенос базы данных из одной системы управления базами данных (СУБД) в другую может быть сложным и требует тщательного планирования и выполнения определенных шагов. Процесс миграции может варьироваться в зависимости от СУБД, используемых в исходной и целевой системах. Есть множество инструментов и способов миграции.

Этапы миграции и инструменты миграции описаны далее.

Первым этапом является подготовка исходной базы данных. Рекомендуется перед началом миграции создать резервную копию исходной базы данных для возможности восстановления в случае неудачи.

Второй этап – это анализ целевой СУБД. Каждая СУБД имеет свои уникальные особенности и форматы хранения данных. Исследуйте структуру и функциональность целевой СУБД для адаптации миграции.

Третий этап – перенос таблиц и данных из таблиц. В данном примере миграция происходит между СУБД PostgreSQL и MSSQL. Перед началом миграции требуется создать пустую базу данных в MSSQL. Для миграции таблиц и данных будет использоваться инструмент MSSQL SQL Server Import and Export Wizard. Для соединения с PostgreSQL требуется установить драйвер psqldbcs. После установки можно приступить к процессу переноса. Зайдите в инструмент SQL Server Import and Export Wizard, в Data source выберите раздел .Net Framework Data Provider for Odbc. Для подключения к PostgreSQL используйте пример из листинга 6.1.

```
Driver={PostgreSQL
UNICODE};server=localhost;port=5432;database=univer;uid=admin;pwd=1
23
```

Листинг 6.1 – строка подключения к PostgreSQL

В следующем разделе в пункте Destination выберите Microsoft OLE DB Provider for SQL Server. Для параметра Server name адрес имя вашего сервера, в параметре Database укажите недавно созданную базу данных.

Четвёртый этап – редактирования запросов. При данном варианте миграции требуется ручное исправление SQL запросов. Типы данных PostgreSQL и MSSQL не совпадают, также отличается синтаксис. Рекомендуется вместо изменения типов данных полей таблицы полностью переписать запросы. Пример приведён в листинге 6.2.

```
PostgreSQL
CREATE TABLE games (
    game_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
MSSQL
CREATE TABLE games (
    game_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
```

Листинг 6.2 – пример изменения запросов

Пятый этап – создание дампа объектов переносимой базы данных. MSSQL и PostgreSQL не имеет инструментов для автоматического переноса объектов базы данных при миграции. Это весьма усложняет процесс, потому что нам надо будет производить перенос объектов вручную. В этом есть небольшой плюс, можно будет контролировать весь процесс, чтобы не потерять данные. Пример создания дампа приведён в Листинге 6.3.

```
pg_dump -h localhost -d univer -U admin -f backup.sql
```

Листинг 6.3 – создание дампа

После успешного создания дампа базы данных, следующим шагом является адаптация всех SQL-запросов к синтаксису MSSQL. Этот этап требует внимательности и тщательности, поскольку каждая СУБД имеет свои особенности и различия в синтаксисе запросов. Однако, эта фаза является неотъемлемой частью процесса миграции и необходима для гарантированного функционирования базы данных в новой среде.

Переписывание SQL-запросов для совместимости с MSSQL – это творческий процесс, который требует глубокого знания структуры данных и специфики языка запросов. Необходимо учесть особенности MSSQL, такие как различия в синтаксисе JOIN-операторов, функциях агрегирования и другие аспекты, чтобы обеспечить точное выполнение запросов.

Миграция базы данных представляет собой сложный и многогранный процесс. Ряд уникальных аспектов требуют внимательного планирования и внедрения, чтобы обеспечить успешную и беспроblemную миграцию. В данном разделе приводился пример ручной миграции базы данных из одной СУБД в другую. Это весьма сложный и долгий процесс.

7 Краткое описание веб-сайта для демонстрации

Веб-сайт для продажи видеоигр. Вот несколько особенностей. Создавайте свой уникальный список избранного, добавление товара в корзину. Просматривайте и управляйте данными игры через страницу администратора. Добавлять новые игры, редактируйте уже существующие игры и удалять их. Для администраторов доступна панель статистики, где отображаются данные по продаже видеоигр, включая количество проданных игр, веб-сайт разработан с упором на простоту использования и интуитивный интерфейс. Навигация легкая, а функциональность — расширена для различных нужд.

Веб-сайт разработан с акцентом на простоту использования, предоставляя интуитивно понятный интерфейс для удобства каждого пользователя.

8 Руководство пользователя

При первом посещении клиентской части вас встретит страница регистрации, где вы можете создать учетную запись, вводя необходимые данные. После успешной регистрации вы можете войти в систему, используя свой email и пароль на странице авторизации. Она приведена на рисунке 8.1.

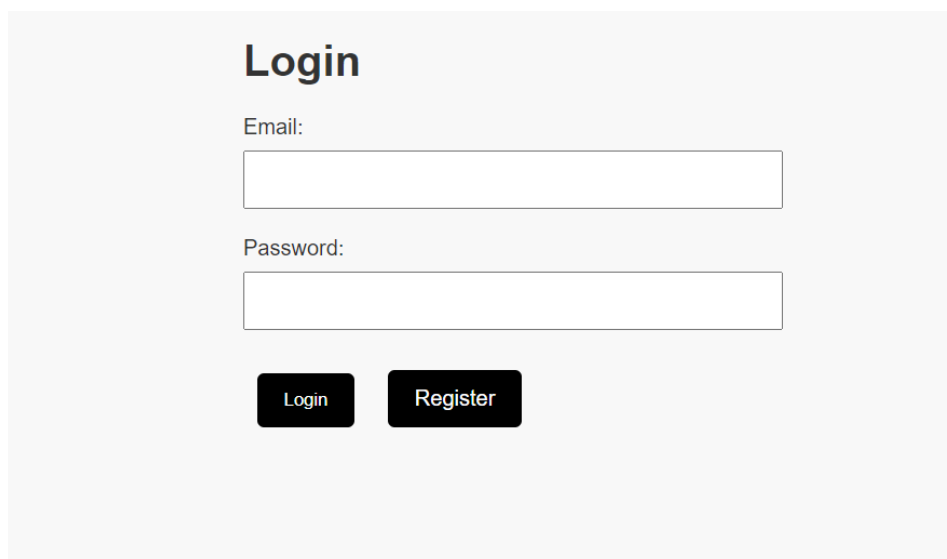
The image shows a web form titled "Login" in a bold, dark font. Below the title, there are two input fields. The first is labeled "Email:" and the second is labeled "Password:". Both labels are in a smaller, regular font. Below the input fields, there are two buttons: "Login" and "Register". Both buttons are dark with white text. The entire form is set against a light gray background.

Рисунок 8.1 – Страница входа в аккаунт

Пользователь может зарегистрироваться на веб-сайте выбрав соответствующую кнопку рядом с кнопкой входа а после успеха получить доступ. Страница регистрации приведена на рисунках 8.2.

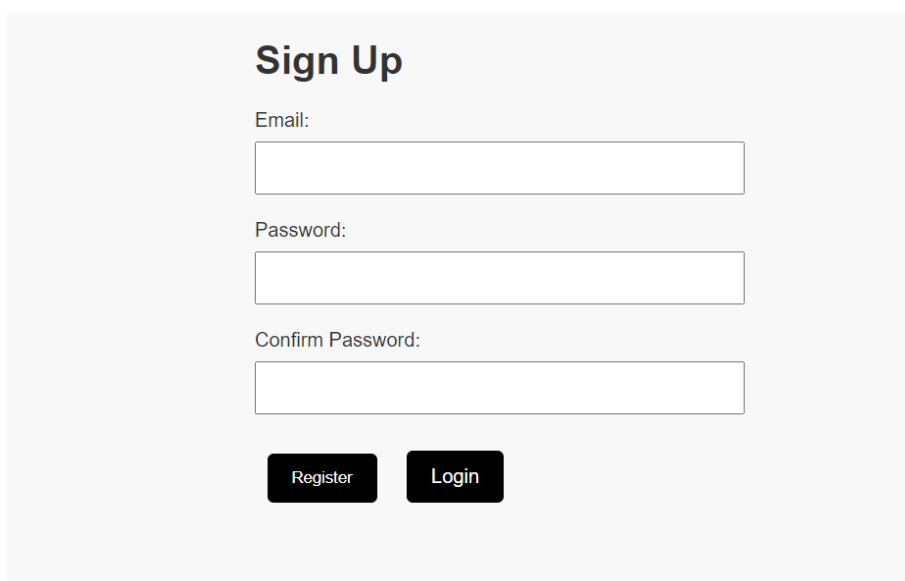
The image shows a web form titled "Sign Up" in a bold, dark font. Below the title, there are three input fields. The first is labeled "Email:", the second is labeled "Password:", and the third is labeled "Confirm Password:". All labels are in a smaller, regular font. Below the input fields, there are two buttons: "Register" and "Login". Both buttons are dark with white text. The entire form is set against a light gray background.

Рисунок 8.2 – Регистрация пользователя

При регистрации и авторизации происходит проверки по всем необходимым полям. Также осуществляется проверка на существование пользователя.

После успешной авторизации система автоматически перенаправит вас на страницу создания со списком игр. Здесь вы можете получить необходимую информацию об играх. Нажав на кнопку добавления в корзину, товар добавится в каталог корзины. Страница приведена на рисунке 8.3.

User List

Enter keyword

Search

View Cart

Favorites

Id	Title	Description	Price	Add to Cart	Add to Favorites
100048	Game 202345678io	Description for Game 20	\$3856789.00	Add to Cart	Add to Favorites
100053	Game 25	Description for Game 25	\$70.37	Add to Cart	Add to Favorites
100054	Game 26	Description for Game 26	\$61.24	Add to Cart	Add to Favorites

Рисунок 8.3 – Страница со списком игр

После добавления товара в корзину, вы можете перейти на станицу с товарами, которые были добавлены в этот каталог. Страница с контентом представлены на рисунке 8.4.

Shopping Cart

Id	Title	Description	Price	Buy	Delete
100048	Game 202345678io	Description for Game 20	\$3856789.00	Buy Now	Remove
100061	Game 3456789op[Description for Game 33	\$9134.00	Buy Now	Remove
100153	Game 125	Description for Game 125	\$44.03	Buy Now	Remove

Рисунок 8.4 – Страница корзины

Находясь на страницу со списком игр, вы можете добавить понравившуюся игру на страницу избранного. Страница со списком избранного представлены на рисунке 8.5.

Favorites

Id	Title	Description	Price	Action
100048	Game 202345678io	Description for Game 20	\$3856789.00	Remove
100053	Game 25	Description for Game 25	\$70.37	Remove

Рисунок 8.5 – Страница избранного

Используя данные администратора для входа, вы перейдете на страницу с редактированием списка игр и просмотром статистики. Страница администратора представлена на рисунке 8.6.

Games List

Enter keyword

Search

Add game

Sales Statistics

Id	Title	Description	Price	Update	Delete
100048	Game 202345678io	Description for Game 20	\$3856789.00	<div>Edit</div>	<div>Delete</div>
100053	Game 25	Description for Game 25	\$70.37	<div>Edit</div>	<div>Delete</div>
100054	Game 26	Description for Game 26	\$61.24	<div>Edit</div>	<div>Delete</div>

Рисунок 8.6 – Страница администратора

Заключение

База данных является ключевым элементом любой современной организации, обеспечивая надежное хранение и управление информацией. В данной работе была поставлена задача разработки базы данных университета с применением технологии миграция данных и объектов из одной СУБД в другую в СУБД PostgreSQL.

В процессе выполнения работы были использованы различные объекты, включая таблицы, триггеры и функции, чтобы обеспечить структурированное хранение данных и своевременный доступ к ним. В результате, цель работы была успешно достигнута, и база данных готова к использованию. Были разработаны роли для управления доступом к данным и обеспечения безопасности.

Тестирование базы данных было проведено при использовании большого объема данных, и результаты были положительными. Были реализованы процедуры для импорта и экспорта данных в формате JSON, что обеспечило удобство использования и управления данными.

Одной из ключевых особенностей разработанной базы данных является технология миграции базы данных из одной СУБД в другую.

Таким образом, была успешно выполнена задача по разработке базы данных университета на основе СУБД PostgreSQL. Разработанная база данных позволяет хранить и управлять большим объемом данных, обеспечивает безопасный доступ к ним и предоставляет возможность импорта и экспорта данных в различных форматах. Кроме того, технология миграции базы данных позволяет мигрировать базу данных из одной СУБД на другую.

Список используемых источников

1. PostgreSQL Сайт о программировании [Электронный ресурс] / Режим доступа: <https://postgrespro.ru/docs/postgresql.com> – Дата доступа: 18.09.2023.
2. Postgresqtutorial.com [Электронный ресурс] / Режим доступа: <https://www.postgresqtutorial.com/> – Дата доступа: 21.09.2023.
3. Stackoverflow.com [Электронный ресурс] / Режим доступа: <https://stackoverflow.com> – Дата доступа: 22.10.2023.
4. Миграция PostgreSQL. <https://learn.microsoft.com/en-us/sql/integration-services/import-export-data/connect-to-a-postgresql-data-source-sql-server-import-and-export-wizard?view=sql-server-ver16> – Дата доступа: 02.12.2023.

ПРИЛОЖЕНИЕ А

Создание базы данных

```
CREATE TABLE roles (  
    role_id SERIAL PRIMARY KEY,  
    role_name VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE users (  
    user_id SERIAL PRIMARY KEY,  
    username VARCHAR(50) NOT NULL,  
    role_id INT,  
    password VARCHAR(255),  
    FOREIGN KEY (role_id) REFERENCES roles(role_id)  
);  
  
CREATE TABLE games (  
    game_id SERIAL PRIMARY KEY,  
    title VARCHAR(255) NOT NULL,  
    description TEXT,  
    price NUMERIC(10, 2) NOT NULL,  
    CONSTRAINT unique_game_title UNIQUE (title)  
);  
  
CREATE TABLE shopping_cart (  
    cart_id SERIAL PRIMARY KEY,  
    user_id INT,  
    game_id INT,  
    FOREIGN KEY (user_id) REFERENCES users(user_id),  
    FOREIGN KEY (game_id) REFERENCES games(game_id)  
);  
  
CREATE TABLE payments (  
    payment_id SERIAL PRIMARY KEY,  
    user_id INT,  
    amount DECIMAL(10, 2) NOT NULL,  
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (user_id) REFERENCES users(user_id)  
);  
  
CREATE TABLE sales_statistics (  
    statistic_id SERIAL PRIMARY KEY,  
    game_id INT,  
    quantity_sold INT,  
    total_revenue DECIMAL(10, 2),  
    sales_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (game_id) REFERENCES games(game_id)  
);  
  
ALTER TABLE favorites  
ADD CONSTRAINT favorites_game_id_fkey
```

```
FOREIGN KEY (game_id) REFERENCES games (game_id)
ON DELETE CASCADE;
ALTER TABLE favorites
DROP CONSTRAINT favorites_game_id_fkey;

CREATE TABLE payments (
    payment_id SERIAL PRIMARY KEY,
    user_id INT,
    game_id INT,
    amount DECIMAL(10, 2),
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (game_id) REFERENCES games(game_id)
);
```


ПРИЛОЖЕНИЕ Б

Создание процедур

```

create
create procedure get_total_games_count(OUT total_games_count integer)
    language plpgsql
as
$$
BEGIN
    -- Возврат общего количества игр
    SELECT COUNT(*) INTO total_games_count FROM games;
END;
$$;
create function add_to_cart(p_user_id integer, p_game_id integer)
returns void
    language plpgsql
as
$$
BEGIN
    INSERT INTO shopping_cart (user_id, game_id) VALUES (p_user_id,
p_game_id);
END;
$$;
CREATE OR REPLACE PROCEDURE authenticate_user(
    p_username VARCHAR(50),
    p_password VARCHAR(255),
    OUT p_user_id INT,
    OUT p_role_id INT
) AS $$
DECLARE
    user_data RECORD;
BEGIN
    -- Поиск пользователя по имени и паролю
    SELECT user_id, role_id
    INTO user_data
    FROM users
    WHERE username = p_username AND password = p_password;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'User not found: %', p_username;
    END IF;

    -- Возвращение данных пользователя
    p_user_id := user_data.user_id;
    p_role_id := user_data.role_id;
END;
$$ LANGUAGE plpgsql;
create procedure delete_game(IN p_keyword character varying)
    language plpgsql
as
$$

```

```

DECLARE
    rows_affected integer;
BEGIN
    -- Используем ILIKE для поиска по названию и описанию
    DELETE FROM games WHERE title ILIKE '%' || p_keyword || '%' OR
description ILIKE '%' || p_keyword || '%'
    RETURNING * INTO rows_affected;

    -- Проверяем количество удаленных строк
    IF rows_affected = 0 THEN
        -- Если нет удаленных строк, генерируем исключение
        RAISE EXCEPTION 'Game Not Found';
    END IF;
END;
$$;
create procedure import_from_json()
    language plpgsql
as
$$
BEGIN
    CREATE TABLE IF NOT EXISTS temp (data json);
    COPY temp FROM 'D:/db_course_project/json/games.json';

    CREATE TABLE IF NOT EXISTS games (
        game_id SERIAL PRIMARY KEY,
        title VARCHAR(255) NOT NULL,
        description TEXT,
        price DECIMAL(10, 2) NOT NULL
    );

    INSERT INTO games (title, description, price)
    SELECT
        value->>'title' as title,
        value->>'description' as description,
        (value->>'price')::numeric(10, 2) as price
    FROM temp, json_array_elements(data) as value;

    DROP TABLE IF EXISTS temp;
END;
$$;
create procedure register_user(IN p_username character varying, IN
p_role_id integer, IN p_password character varying, OUT user_id
integer)
    language plpgsql
as
$$
BEGIN
    -- Проверка существования role_id
    PERFORM role_id FROM roles WHERE role_id = p_role_id;

    -- Вставка пользователя и возврат user_id
    INSERT INTO users (username, role_id, password)
    VALUES (p_username, p_role_id, p_password)

```

```

RETURNING users.user_id INTO user_id;

-- Если вы хотите вернуть user_id, используйте ключевое слово OUT
EXCEPTION
    WHEN others THEN
        RAISE EXCEPTION 'Error in register_user: %', SQLERRM;
END;
$$;
create function remove_from_cart(p_cart_id integer) returns void
    language plpgsql
as
$$
BEGIN
    DELETE FROM shopping_cart WHERE cart_id = p_cart_id;
END;
$$;
create function search_games(keyword text)
    returns TABLE(game_id integer, title character varying,
description text, price numeric)
    language plpgsql
as
$$
BEGIN
    RETURN QUERY
    SELECT
        game_id,
        title,
        description,
        price
    FROM
        games
    WHERE
        LOWER(title) LIKE LOWER('%' || keyword || '%')
        OR LOWER(description) LIKE LOWER('%' || keyword || '%');
END;
$$;

alter function search_games(text) owner to postgres;
create procedure update_sales_statistics(IN p_game_id integer, IN
p_amount numeric)
    language plpgsql
as
$$
BEGIN
    INSERT INTO sales_statistics (game_id, quantity_sold,
total_revenue, sales_date)
    VALUES (
        p_game_id,
        1,
        p_amount,
        CURRENT_TIMESTAMP
    )
    ON CONFLICT (game_id) DO UPDATE

```

```
        SET
            quantity_sold = sales_statistics.quantity_sold + 1,
            total_revenue = sales_statistics.total_revenue + p_amount,
            sales_date = CURRENT_TIMESTAMP;
    END;
$$;

alter procedure update_sales_statistics(integer, numeric) owner to
postgres;
```

ПРИЛОЖЕНИЕ В

Диаграмма вариантов использования

