МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

Итоговый проект

по курсу «Базы данных»

Сервис по продаже игровых ключей

Выполнил: Зиновьев. Д. И.

Группа: М8О-308Б-22

Преподаватель: Малахов А. В.

Москва, 2024

Оглавление

Постановка задачи

Общие требования к итоговому проекту:

1. При реализации курсового проекта допускается только использование СУБД PostgreSQL.

2. Необходимо выбрать предметную область для создания базы данных. Выбранная предметная область должна быть уникальной для всего потока, а не только в рамках учебной группы.

3. Необходимо описать модели предметной области и уровня инфраструктуры и их назначение в рамках реализуемого проекта (минимальное количество моделей предметной области и уровня инфраструктуры - 5). Также необходимо выполнить проектирование логической структуры базы данных. Все таблицы, связанные с описанными моделями предметной области, должны находиться в 3NF или выше. База данных должна иметь минимум 7 таблиц.

4. Клиентское приложение должно быть в виде WEB или оконного приложения.

5. Необходимо организовать различные роли пользователей и права доступа к данным (например: администратор,редактор, рядовой пользователь). Клиентское приложение, взаимодействующее с базой данных, должно предоставлять функционал для авторизации пользователя по логину и паролю (хранение непосредственно пароля в базе данных запрещено, надо хранить HASH от этого пароля и проверять его).

6. При разработке функционала базы данных следует организовать логику обработки данных не на стороне клиента (Frontend), а на стороне серверного приложения (Backend). Все обработки «SQL запросов», «работа бизнес-логики должны находится в BACKEND части. FRONT только для отображения.

7. Запросы должны быть асинхронны. То есть, при нажатии на форму она не должна зависать. При нажатии на форму одним пользователем,

другой должен иметь возможность свободно пользоваться приложением. То есть действия разных пользователей независимы.

8. Необходимо реализовать возможность создания администратором архивных копий базы данных и восстановления данных из клиентского приложения.

9. Передача параметров в SQL запрос должна происходить только через parameters.

Описание

Проект представляет собой веб-сервис для предоставления услуг по продаже игровых игр в виде ключей, по средстам предоставления их через магазины. Вся информация о пользователях, играх, ключах, магазинах PostgreSQL. Веб-интерфейс для работы с сервисами реализован с использованием Streamlit. Для взаимодействия с базой данных используется psycopg2.

Проект включает в себя сервисы для авторизации и регистрации пользователей, добавления игр, регистрирование магазина, добавления ключей в продажу, их покупка и анализ продаж.

Описание и схема моделей предметной области

Модель базы данных для цифровой платформы продажи игр предоставляет структуру для управления пользователями, магазинами, играми, игровыми ключами, продажами и ценами. Эта система обеспечивает функционал для работы с ключами для игр, продажами и аналитикой. Модель включает несколько ключевых таблиц, а также представления и функции для обеспечения дополнительных возможностей.

1. users (Пользователи)

Таблица содержит информацию о пользователях платформы, включая их роли (покупатель, продавец или администратор).

Основные столбцы:

- **id:** Уникальный идентификатор пользователя.
- **username:** Уникальное имя пользователя.
- **password:** Хэш пароля пользователя.
- **role:** Роль пользователя (покупатель, продавец или администратор).
- **created_at:** Дата и время создания пользователя.

Связи:

Связана с другими таблицами, такими как shops, pending_sellers, и sales.

```
CREATE TABLE IF NOT EXISTS users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role VARCHAR(20) CHECK (role IN ('buyer', 'seller', 'admin')) DEFAULT
```

'buyer',
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

2. pending_sellers (Ожидающие продавцы)

Хранит заявки пользователей, которые хотят стать продавцами.

Основные столбцы:

- **user_id:** Идентификатор пользователя.
- **created_at:** Дата подачи заявки.

Связи:
Связана с таблицей users через user_id.

```
CREATE TABLE IF NOT EXISTS pending_sellers (
  user_id INT PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3. shops (Магазины)

Таблица хранит информацию о магазинах, созданных продавцами.

Основные столбцы:

- **shop_id:** Уникальный идентификатор магазина.
- **seller_id:** Идентификатор продавца (пользователя).
- **name:** Уникальное имя магазина.

- **created_at:** Дата и время создания магазина.

Связи:

Связана с таблицей users через seller_id.

```
CREATE TABLE IF NOT EXISTS shops (
  shop_id SERIAL PRIMARY KEY,
  seller_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
  name VARCHAR(255) UNIQUE NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. platforms (Платформы)

Таблица платформ (например, PlayStation, Xbox, Steam).

Основные столбцы:

- **platform_id:** Уникальный идентификатор платформы.
- **name:** Название платформы.

Связи:

Используется в таблице games для указания платформы игры.

```
CREATE TABLE IF NOT EXISTS platforms (
  platform_id SERIAL PRIMARY KEY,
  name VARCHAR(50) NOT NULL UNIQUE
);
```

5. games (Игры)

Хранит информацию об играх, которые можно приобрести.

Основные столбцы:

- **game_id:** Уникальный идентификатор игры.
- **title:** Название игры.
- **publisher:** Издатель игры.
- **genre:** Жанр игры.
- **platform_id:** Идентификатор платформы.
- **release_date:** Дата выпуска игры.
- **shop_id:** Идентификатор магазина, связанного с игрой.

Связи:
Связана с таблицами platforms и shops.

```
CREATE TABLE IF NOT EXISTS games (
  game_id SERIAL PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  publisher VARCHAR(255),
  genre VARCHAR(255),
  platform_id INT REFERENCES platforms(platform_id) ON DELETE CASCADE,
  release_date DATE,
  shop_id INTEGER REFERENCES shops(shop_id) ON DELETE SET NULL
);
```

6. keys (Ключи)

Содержит информацию о цифровых ключах для игр.

Основные столбцы:

- **key_id:** Уникальный идентификатор ключа.
- **game_id:** Идентификатор игры.
- **key_value:** Уникальный ключ.
- **status:** Статус ключа (доступен, продан и т. д.).
- **added_at:** Дата добавления ключа.
- **shop_id:** Магазин, предоставивший ключ.

Связи:

Связана с таблицами games и shops.

```
CREATE TABLE IF NOT EXISTS keys (
   key_id SERIAL PRIMARY KEY,
   game_id INT REFERENCES games(game_id) ON DELETE CASCADE,
   key_value VARCHAR(255) UNIQUE NOT NULL,
   status VARCHAR(20) DEFAULT 'available',
   added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   shop_id INTEGER REFERENCES shops(shop_id) ON DELETE CASCADE
);
```

7. sales (Продажи) и sales_details (Детали продаж)

sales: Информация о покупках.
sales_details: Список ключей, приобретенных в рамках одной продажи.

```
CREATE TABLE IF NOT EXISTS sales (
   sale_id SERIAL PRIMARY KEY,
   buyer_id INT REFERENCES users(id) ON DELETE SET NULL,
   sale_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE IF NOT EXISTS sales_details (
    sale_id INT REFERENCES sales(sale_id) ON DELETE CASCADE,
    key_id INT REFERENCES keys(key_id) ON DELETE CASCADE,
    PRIMARY KEY (sale_id, key_id)
);
```

Представления и функции

v_available_keys

Представление отображает доступные ключи с их информацией.

```
CREATE VIEW v_available_keys AS
SELECT
    k.key_id,
    k.key_value,
    g.title AS game_title,
    p.name AS platform_name,
    k.status,
    s.name AS shop_name,
    pr.price
FROM
    keys k
    JOIN games g ON k.game_id = g.game_id
    JOIN platforms p ON g.platform_id = p.platform_id
    JOIN shops s ON k.shop_id = s.shop_id
    LEFT JOIN prices pr ON k.key_id = pr.key_id
WHERE
```

```
    k.status = 'available' AND pr.price IS NOT NULL;
```

v_sales_statistics

Отображает статистику продаж по играм и платформам.

```
CREATE VIEW v_sales_statistics AS
SELECT
    s.sale_date,
    g.title AS game_title,
    pl.name AS platform_name,
    COUNT(k.key_id) AS sold_keys_count,
    SUM(pr.price) AS total_revenue
FROM
    sales s
    JOIN sales_details sd ON s.sale_id = sd.sale_id
    JOIN keys k ON sd.key_id = k.key_id
    JOIN games g ON k.game_id = g.game_id
    JOIN platforms pl ON g.platform_id = pl.platform_id
    JOIN prices pr ON k.key_id = pr.key_id
WHERE
    s.sale_date BETWEEN pr.start_date AND COALESCE(pr.end_date, '5999-
12-31'::date)
GROUP BY
    s.sale_date, g.title, pl.name;
```

Функция удаления пользователя

Функция проверяет права администратора перед удалением.

```
CREATE OR REPLACE FUNCTION delete_user(admin_id INT,
target_user_id INT)
RETURNS VOID AS $$
DECLARE
   admin_role VARCHAR(20);
BEGIN
   SELECT role INTO admin_role FROM users WHERE id = admin_id;
   IF admin_role <> 'admin' THEN
      RAISE EXCEPTION 'Only an admin can delete users.';
   END IF;
   DELETE FROM users WHERE id = target_user_id;
END;
$$ LANGUAGE plpgsql;
```

Общая схема моделей предметной области:

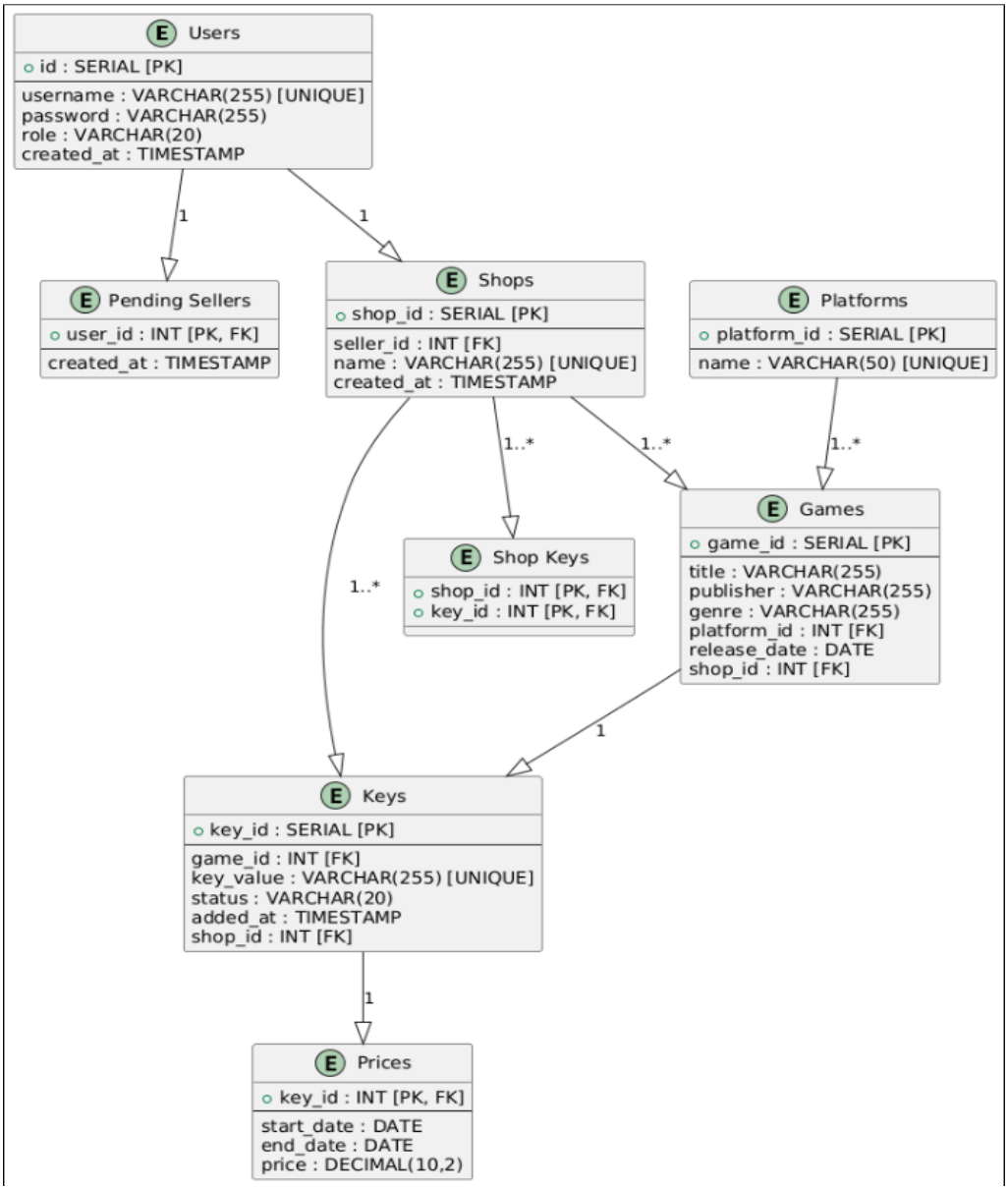

рис.1 “Схема моделей предметной области”

- **Users --|> PendingSellers : "1"**

  Каждый пользователь может быть связан максимум с одной записью в таблице PendingSellers. Это означает, что пользователь может быть зарегистрирован как ожидающий продавец.

- **Users --|> Shops : "1"**

  Каждый пользователь (в роли продавца) может владеть максимум одним магазином. Один пользователь связан только с одной записью в таблице Shops.

- **Shops --|> Games : "1..*"**
  Один магазин может продавать одну или несколько игр. Каждая игра должна быть привязана как минимум к одному магазину.
- **Platforms --|> Games : "1..*"**
  Одна платформа может быть связана с одной или несколькими играми. Каждая игра доступна как минимум на одной платформе.
- **Games --|> ShopGames : "1..*"**
  Каждая игра может быть добавлена в один или несколько магазинов через таблицу ShopGames. Эта связь представляет собой много-к-многим.
- **Games --|> Keys : "1"**
  Каждая игра связана с одним или несколькими цифровыми ключами. Каждый ключ принадлежит только одной игре.
- **Shops --|> Keys : "1..*"**
  Каждый магазин может иметь один или несколько цифровых ключей. Ключи принадлежат конкретному магазину.
- **Sales --|> SalesDetails : "1..*"**
  Каждая продажа связана с одной или несколькими деталями продажи (например, проданными ключами).
- **Keys --|> SalesDetails : "1"**
  Каждый ключ может участвовать только в одной детали продажи. Один ключ продается только один раз.
- **Keys --|> Prices : "1"**
  У каждого ключа есть одна запись в таблице цен Prices. Цена описывает период действия стоимости.
- **Shops --|> ShopKeys : "1..*"**
  Каждый магазин может быть связан с одним или несколькими ключами через таблицу ShopKeys. Это связь много-к-многим между магазинами и ключами.

Описание моделей уровня инфраструктуры

Сервис авторизации/регистрации

Сервис авторизации и регистрации пользователей предоставляет функциональность для создания новых аккаунтов и проверки учетных данных для входа в систему. Он отвечает за безопасное хранение паролей и управление пользовательскими данными.

Основные функции:

- Регистрация новых пользователей.
- Авторизация пользователей (проверка учетных данных).
- Обработка и хранение паролей с использованием алгоритмов хеширования.
- Управление ролями пользователей ("buyer", "seller", “admin”).

Сервис бронирования

Сервис бронирования обрабатывает все операции, связанные с поиском рейсов, бронированием мест на рейсах и управлением статусами бронирований. Он отвечает создание рейсов, управление доступностью билетов и бронированиями, а также создание возвратных рейсов.

Основные функции:

- Поиск рейсов по различным параметрам (дата, аэропорты отправления и прибытия).
- Бронирование мест на рейсах.
- Управление статусами бронирования ("подтверждено", "отменено").
- Создание возвратных рейсов и управление их доступностью.
- Обновление количества доступных мест на рейсах после бронирования.

Сервис платежей

Сервис платежей занимается созданием и обновлением статусов платежей.

Основные функции:

- Обработка платежей за покупку ключей.
- Обновление статуса ключей, available или sold.

Сервис администрирования БД

Сервис администрирования БД управляет процессо удаления игр и регистрирования игр PostgreSQL.

SELECT s.shop_id, s.name AS shop_name, g.title, k.key_value, k.status, p.name AS platform FROM keys k JOIN games g ON k.game_id = g.game_id JOIN platforms p ON g.platform_id = p.platform_id JOIN shop_keys sk ON k.key_id = sk.key_id JOIN shops s ON sk.shop_id = s.shop_id WHERE k.status = 'available';

Разработка запросов к базе данных

Основные запросы к базе данных, используемые для реализации сервиса бронирования авиабилетов.

INSERT INTO platforms (name) VALUES
('Steam'),
('PlayStation Store'),
('Xbox Store'),
('Epic Games Store'),
('Nintendo eShop');

Запрос добавляет платформы

Этот запрос взаимодействует с базой данных, связанной с продажами видеоигр. Основная цель — извлечь статистику продаж для конкретной игры по её идентификатору (game_id). Запрос охватывает данные о продажах, играх, платформах, ключах активации и ценах.

SELECT s.sale_date, g.title AS game_title, pl.name AS platform_name, COUNT(k.key_id) AS sold_keys_count, SUM(pr.price) AS total_revenue FROM sales s JOIN sales_details sd ON s.sale_id = sd.sale_id JOIN keys k ON sd.key_id = k.key_id JOIN games g ON k.game_id = g.game_id JOIN platforms pl ON g.platform_id = pl.platform_id JOIN prices pr ON g.game_id = pr.barcode WHERE g.game_id = %s AND s.sale_date BETWEEN pr.start_date AND COALESCE(pr.end_date, '5999-12-31'::date) GROUP BY s.sale_date,

g.title, pl.name ORDER BY s.sale_date; """, (game_id,)) results = cur.fetchall()

Вывод

В рамках проекта с использованием базы данных для системы управления продажами и ключами для видеоигр, была разработана структура с несколькими ключевыми таблицами и функциями, которые обеспечивают эффективную работу с пользователями, магазинами, играми и платежами.

Роли пользователей

Система поддерживает несколько ролей пользователей:

- **Покупатель (buyer)**: Стандартная роль для пользователей, которые совершают покупки.
- **Продавец (seller)**: Роль для пользователей, которые управляют магазинами и продают ключи.
- **Администратор (admin)**: Роль для пользователей, имеющих доступ к управлению системой, включая удаление пользователей и управление магазином.

Безопасность и управление пользователями

- **Пароли** пользователей хранятся в зашифрованном виде, что обеспечивает безопасность данных.
- **Удаление пользователей**: Специальная функция delete_user позволяет администраторам удалять пользователей, включая связанные с ними магазины и ключи. Такая функциональность требует проверки роли пользователя, чтобы предотвратить неправомерные действия.

Магазины и игры

- Магазины и их продавцы связаны в таблице **shops**, где указаны продавцы, которые управляют магазинами, и их товары (игры).
- Каждая игра в системе привязана к платформе через таблицу **games**, которая также ссылается на магазин, в котором игра доступна.

Ключи и цены

- Ключи для игр хранятся в таблице **keys**, где каждый ключ связан с игрой, магазином и имеет статус (например, "available" для доступных ключей).
- Для ключей предусмотрена таблица **prices**, которая хранит цену для каждого ключа в разные периоды.

Продажи и статистика

- Таблица **sales** фиксирует информацию о каждой продаже, включая дату и покупателя. Детали продажи (ключи и их стоимость) хранятся в таблице **sales_details**.
- Для получения статистики о продажах используется представление **v_sales_statistics**, которое агрегирует данные по количеству проданных ключей и общей выручке за каждый день.
- Представление **v_available_keys** предоставляет список доступных ключей с информацией о цене, игре и магазине.

Механизм работы

- Система использует каскадное удаление для поддержания целостности данных. Например, при удалении продавца или магазина автоматически удаляются связанные с ним записи в других таблицах (например, ключи и игры).
- Все операции, такие как создание платежей или отмена бронирований, выполняются с учетом атомарности, что гарантирует консистентность данных.

Резервное хранение и масштабируемость

- Для обеспечения надежности и восстановления данных предусмотрены механизмы резервного копирования, что позволяет минимизировать риски потери информации.
- Система настроена на поддержку масштабируемости, что позволяет эффективно работать с большим количеством данных, обеспечивая стабильную работу как для пользователей, так и для администраторов.

Эта база данных представляет собой устойчивую и безопасную систему для управления продажами и ключами для игр, с возможностью масштабирования и оптимизацией работы для различных ролей пользователей.

Список использованных источников

1.https://docs.streamlit.io/develop

2. https://www.postgresql.org/docs/17/index.html

3. https://www.psycopg.org/docs/usage.html

4. GitHub - CraneHere/sample_course_work