МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет

имени академика С.П. Королева» (Самарский университет)

Институт информатики и кибернетики Кафедра технической кибернетики

**ОТЧЕТ ПО ЛР1**

**Проектирование приложения «ArcadeZone»**

Выполнили студенты группы 6303-010302D

Щетинин Михаил Маргарян Гор

САМАРА 2025

1. Описание базы данных

На рисунке 1 представлена логическая схема базы данных.

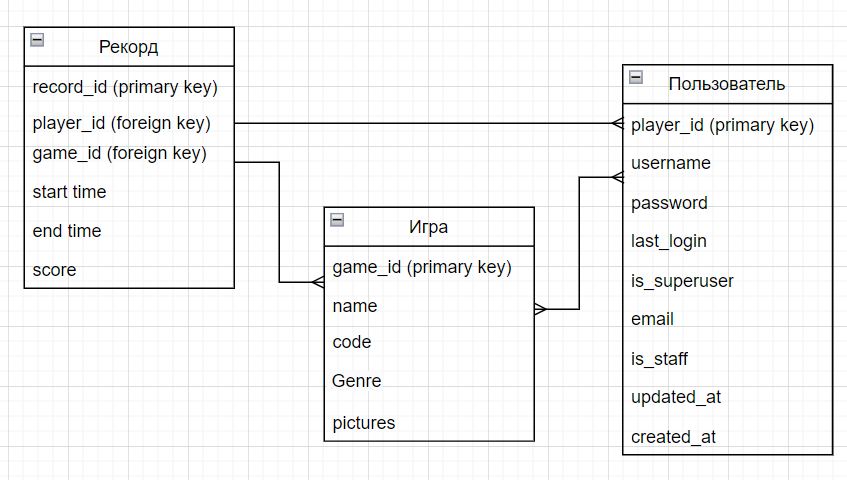


Рисунок 1 – Логическая схема базы данных

В таблице Рекорд находятся все данные о рекордах пользователей в играх со следующими полями:

1. record\_id – поле, содержащее id рекорда (первичный ключ)
2. player\_id – поле, содержащее id пользователя (внешний ключ)
3. game\_id – поле, содержащее id игры (внешний ключ)
4. start\_time – поле, содержащее время запуска игры пользователем
5. end\_time – поле, содержащее время завершения игры
6. score – поле, содержащее количество набранных очков

В таблице Игра находятся все данные об играх со следующими полями:

1. game\_id – поле, содержащее id игры (первичный ключ)
2. name – поле, содержащее название игры
3. code – поле, ссылающееся на файл с кодом игры
4. Genre – поле, содержащее наименование жанра игры
5. pictures – поле, содержащее ссылку на изображение игры

В таблице Пользователь находятся все данные пользователей со следующими полями:

1. player\_id – поле, содержащее id пользователя (первичный ключ)
2. username – поле, содержащее имя пользователя
3. password – поле, содержащее пароль игрока (захешированный)
4. last\_login – поле, содержащее дату и время последней авторизации игрока
5. is\_superuser – поле, содержащее булеву переменную, определяющую является ли пользователь суперюзером (администратором)
6. email – поле, содержащее электронную почту пользователя
7. is\_staff – поле, содержащее булеву переменную, определяющую статус подачи заявки пользователя на становление суперюзером
8. updated\_at – поле, содержащее дату и время последнего обновления данных аккаунта
9. created\_at – поле, содержащее дату и время регистрации аккаунта

На рисунках 2, 3, 4 представлено отображение таблиц базы данных в программном средстве DBeaver.

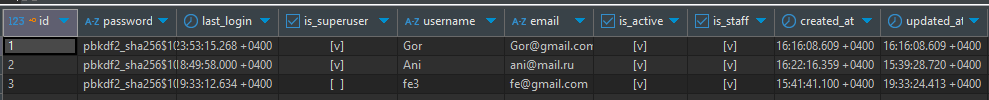


Рисунок 2 – Таблица Пользователь в DBeaver

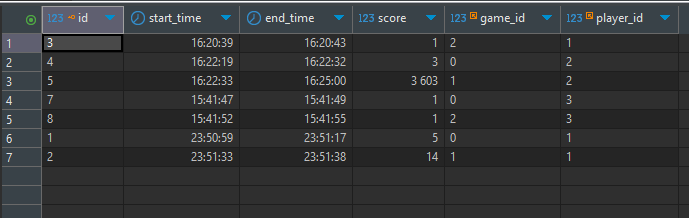


Рисунок 3 – Таблица Рекорд в DBeaver

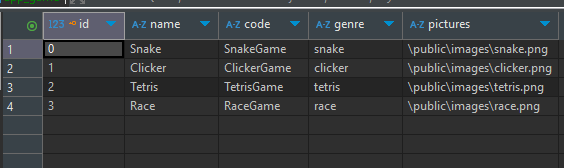


Рисунок 4 – Таблица Игра в DBeaver

1. Скрипты для демонстрации работы с данными

Для демонстрации работы с данными были написаны следующие скрипты:

1. fill\_models.py – заполнение таблиц данными
2. psql\_query.py – запрос данных из БД
3. test\_request.py – тестовый API запрос

Код вышеописанных скриптов представлен в приложения А, Б, В соответственно.

**Приложение А**

import os  
import sys  
import django  
from django.db.models import Q  
# Добавляем путь к корневой директории проекта  
PROJECT\_ROOT = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..'))  
sys.path.insert(0, PROJECT\_ROOT)  
print(sys.path)  
# Настройка Django окружения (укажите путь к settings вашего проекта)  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'arcadezonedb.settings')  
django.setup()  
  
  
from app.models import Player, Game, Record  
  
  
# Заполнение таблицы Game  
  
game1 = Game(1, "Game1", "Game1Game", "genre1", "\public\images\game1.png")  
game2 = Game(2, "Game2", "Game2Game", "genre2", "\public\images\game2.png")  
game3 = Game(3, "Game3", "Game3Game", "genre3", "\public\images\game3.png")  
  
game1.save()  
game2.save()  
game3.save()  
  
print("Таблица Game успешно заполнена.")  
  
# Заполнение таблицы Player  
  
player1 = Player(1, 'Password', 'False', "Player1", 'player1@mail.ru', 'True', 'False', '16:01:00', '16:05:00')  
player2 = Player(2, 'Password', 'False', "Player2", 'player2@mail.ru', 'True', 'True', '16:02:00', '16:06:00')  
player3 = Player(3, 'Password', 'False', "Player3", 'player3@mail.ru', 'True', 'False', '16:03:00', '16:07:00')  
  
player1.save()  
player2.save()  
player3.save()  
  
print("Таблица Player успешно заполнена.")  
  
  
records\_data = [  
 {"id": 0, "start\_time": "9:20", "end\_time": "12:23", "score": "10", "game\_id": "2", "player\_id": "1"},  
 {"id": 1, "start\_time": "8:20", "end\_time": "13:30", "score": "20", "game\_id": "1", "player\_id": "2"},  
 {"id": 2, "start\_time": "7:25", "end\_time": "13:40", "score": "20", "game\_id": "1", "player\_id": "3"},  
 {"id": 3, "start\_time": "6:25", "end\_time": "13:40", "score": "40", "game\_id": "3", "player\_id": "4"},  
]  
  
for record in records\_data:  
 Record.objects.create(\*\*record)

**Приложение Б**

import os  
import sys  
import django  
from django.db.models import Q  
PROJECT\_ROOT = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..'))  
sys.path.insert(0, PROJECT\_ROOT)  
# Настройка Django окружения (укажите путь к settings вашего проекта)  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'arcadezonedb.settings')  
django.setup()  
from app.models import Player, Game, Record  
# Запрос к модели Record с INNER JOIN  
records = Record.objects.filter(  
 Q(game\_\_name="Game1") | Q(game\_\_name="Game2") | Q(game\_\_name="Game3")  
).select\_related('player', 'game').order\_by('-score')[:5]  
  
# Вывод результатов  
for record in records:  
 print(f"Nickname: {record.player.nickname}, Game: {record.game.name}, Score: {record.score}")

**Приложение В**

import requests  
  
url = "http://127.0.0.1:8000/players/login/"  
data = {  
 "email": "Misha@gmail.com",  
 "password": "3"  
}  
  
response = requests.post(url, json=data)  
print(response.json())