Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##### **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе (проекту)**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##### (фамилия, имя, отчество)

Группа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы (проекта) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата подписания) (инициалы и фамилия)

### Волгоград 20\_\_ г.

**Лабораторная работа №1.**

**Ход выполнения работы**

1. База данных проектируется для компьютерной игры. Игра представляет собой пошаговую карточную стратегию. Пользователь может собирать колоды, путём покупки новых карт в магазине, искать лобби для игры против других игроков и смотреть статистику матча/игрока.
2. Функциональные требования:
   * Создание аккаунта
   * Покупки (внутриигровые)
   * Сбор статистики
   * Создание кланов
   * Вступление в клан
   * Создание игровой комнаты

Нефункциональные требования:

* + Удобство
  + Стабильность работы
  + Безопасность

1. Список связей:
   * User один к множеству Card
   * User один к множеству Deck
   * User один к одному Statistics
   * User один к множеству MatchStatistics
   * User один к одному MatchmakersRoom
   * User один к множеству ClanActions
   * User множество к одному Clan
   * Clan один к множеству ClanActions
   * Deck один к множеству Card
   * Shop множество к множеству SellItem

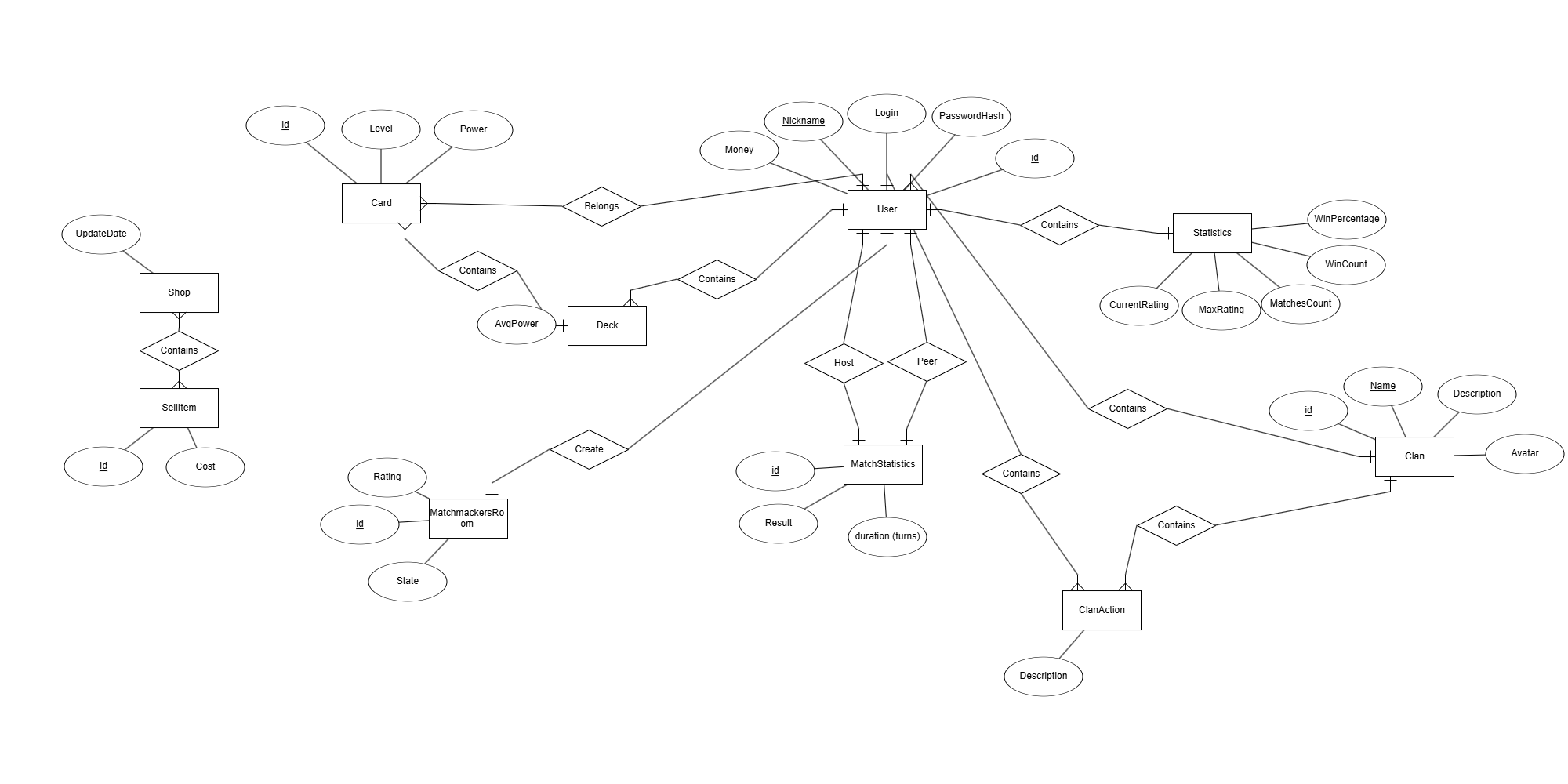
4. Список сущностей:

1. User (пользователь) обладает:  
   – Nickname (имя) – Unique String (от 5 до 20 символов)  
   – Login (логин) – Unique String (от 5 до 20 символов)

– PasswordHash (hash пароля) – String (40 символов)  
– Money (количество денег) - Integer (от 0 до 2147483647)

1. Statistics (статистика) обладает:  
   – WinPercentage (процент побед) – Integer (от 0 до 100)  
   – WinCount (кол-во побед) – Integer (от 0 до 2147483647)  
   – MatchesCount (кол-во матчей) – Integer (от 0 до 2147483647)  
   – MaxRating (макс. кол-во рейтинга) – Integer (от 0 до 2147483647)  
   – CurrentRating (текущий рейтинг) – Integer (от 0 до 2147483647)
2. Clan (клан) обладает:  
   – Name (имя) – Unique String (от 5 до 20 символов)  
   – Description (описание) – String (от 20 до 100 символов)  
   – Avatar(id аватара клана) – Integer (от 0 до 4)
3. ClanAction (событие клана) обладает  
   – Description (описание) – String (от 5 до 100 символов)
4. MatchStatistics (статистика матча) обладает:  
   – Result (результат) – Integer (0 – победил Host, 1 – Победил Peer)  
   – Duration (длительность) – Integer (от 0 до 2147483647)
5. MatchmakersRoom (комната матча) обладает:  
   – Rating (рейтинг) – Integer (от 0 до 2147483647)  
   – State (состояние) – Bool (0 – комната свободна, 1 – идет матч)
6. Deck (дека) обладает:  
   – AvgPower (средняя сила) – Float (от 0 до Infinity)
7. Card (карта) обладает:  
   – Level (уровень) – Integer (от 1 до 10)  
   – Power (сила) – Float (от 0 до 100)
8. Shop (магазин) содержит:  
   – UpdateDate (дата обновления) – Timestamp (от 4713 BC до 294276 AD)
9. SellItem (продаваемый предмет) обладает:  
   – CardId (Id карты (внутриигровой)) – Integer (от 0 до 20)  
   – Cost (стоимость) – Integer (от 0 до 2147483647)

5. Пример ER-диаграммы приведён на следующей странице.



**Лабораторная работа №2.**

**Ход выполнения лабораторной работы:**

1. По своей ER-диаграмме, разработанной в лабораторной работе №1, была спроектирована реляционная модель. Модель приложена в виде приложения №1 в конце работы.
2. Была проведена нормализация, все таблицы нормализованы.
3. Примеры данных для каждой из таблиц:  
     
   – User:  
   Пример 1.

id 0  
nickname superduper12

login andrey12

password\_hash 89sdawWd82398Fkljhfe\*6yWE

money 10225

clan\_id 2323

Пример 2.

id 124

nickname gugobute

login VoVan45

password\_hash jlkkje878FewkjlEHF823NHFG

money 900

clan\_id 66

– Statistics:

Пример 1.

id 12

user\_id 0

win\_count 10

matches\_count 16

max\_rating 1000

current\_rating 1000

Пример 2.

id 1020

user\_id 124

win\_couint 1023

matches\_count 2001

max\_rating 14230

current rating 13620

– Clan:

Пример 1.

id 2323

name Ubivatori

descriptions “Самый крутой клан”

avatar 1

Пример 2.

id 63

name Helsinki

description “Finland’s clan “

avatar 9

– Clan\_action:

Пример 1.

id 23

user\_id 0

clan\_id 2323

description “Игрок купил карту Morture”

Пример 2.

id 10

user\_id 124

clan\_id 63

description “Игрок выйграл сражение”

– Match\_statistics:

Пример 1.

id 0

host\_id 0

peer\_id 124

result 1

duration 16

Пример 2.

id 24252

host\_id 124

peer\_id 0

result 0

duration 22

– Matchmakers\_room:

Пример 1.

id 23

host\_id 124

rating 13220

state 0

Пример 2.

id 11

host\_id 0

rating 1000

state 1

– Deck:

Пример 1.

id 66

user\_id 0

deck\_number 0

Пример 2.

id 6523

user\_id 124

deck\_number 2

– Card:

Пример 1.

id 1201

card\_id 10

user\_id 0

deck\_id 66

level 3

power 10.00

Пример 2.

id 9203

card\_id 8

user\_id 124

deck\_id 6523

level 4

power 7.50

– Shop:

Пример 1.

id 92

update\_date 2025-01-10 14:45:32

Пример 2.

id 1002

update\_date 2025-11-12 12:10:10

– Sell\_item:

Пример 1.

id 649

card\_id 2

cost 12000

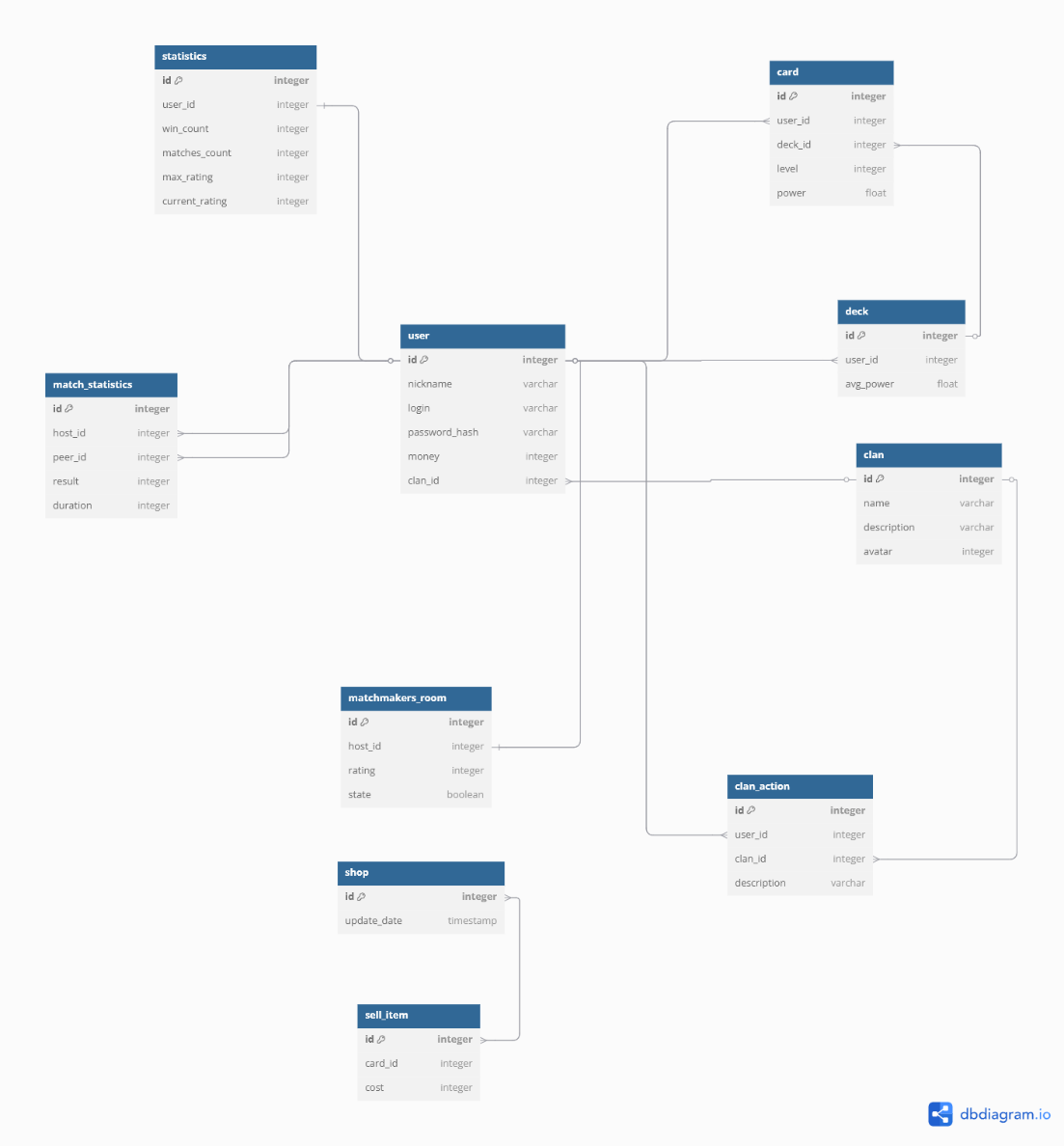
Пример 2.

Id 11

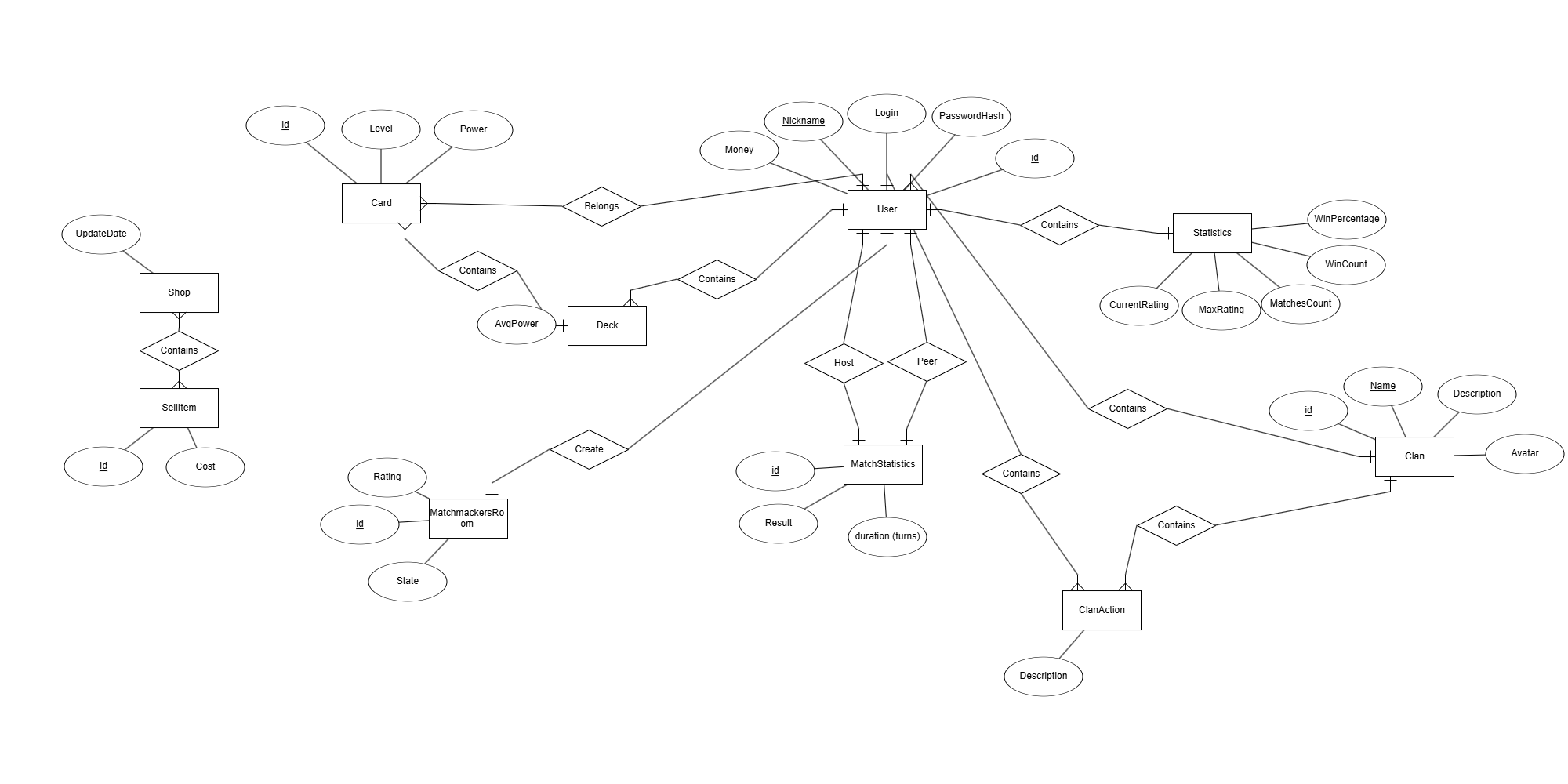
card\_id 7

cost 6000

1. Был создан репозиторий на GitHub и добавлены ER-диаграмма и реляционная модель.  
   Ссылка на репозиторий: <https://github.com/TWTwoFace/db_learning.git>
2. ER-диаграмма находится в конце документа (Приложение №2)  
   Была проведена нормализация:  
   – Из таблицы Statistics убран атрибут WinPercentage (считается как win\_count / matches\_count)  
   – Из таблицы Deck убран атрибут AvgPower (считается как сумма силы всех карт в колоде / количество карт)  
   – Добавлены атрибуты для создания отношений между таблицами

Приложение №1.

Приложение №2.



**Лабораторная работа №3.**

Цель работы: научиться физическому проектированию баз данных используя выбранную СУБД.

Задачи работы: выполнить физическое проектирование для собственной базы данных; выполнить физическое проектирования для базы данных одногруппника.

Ход выполнения лабораторной работы:

Часть 1.

1. Была выбрана СУБД PostgreSQL.

2. Была выбрана графическая оболочка для СУБД: pgAdmin.

3. Была установлена СУБД PostgreSQL на компьютер:

4. Были созданы таблицы в БД используя скрипты.

Пример создания таблицы:

5. Были созданы 11 таблиц (10 – сущностей, 1 для many-to-many отношения):

6. Была создана реляционная модель средствами pgAdmin, она совпадает с той моделью, что была сделана в лаб. работе №2 (кроме таблицы many-to-many)

7. Заполнил данные в таблицы:

Таблица clan:

Таблица users:

Таблица statistics:

Таблица shop:

Таблица sell\_items:

Таблица matchmakers\_room:

Таблица match\_statistics:

Таблица deck:

Таблица clan\_actions:

Таблица cards:

8. Реляционная модель, созданная в лабораторной работе №2.

Часть 2.

Реляционная модель одногруппника:

1. Создание базы данных:

CREATE DATABASE db3\_popov\_potapovich

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

2. Создание таблиц:

-- Таблица поставщиков

CREATE TABLE Supplier (

supplierID INT PRIMARY KEY,

companyName VARCHAR(255),

companyAddress VARCHAR(255)

);

-- Таблица клиентов

CREATE TABLE Client (

clientID INT PRIMARY KEY,

fullName VARCHAR(255),

phoneNumber VARCHAR(255)

);

-- Таблица менеджеров

CREATE TABLE Manager (

managerID INT PRIMARY KEY,

fullName VARCHAR(255)

);

-- Таблица запчастей

CREATE TABLE SparePart (

sparePartID INT PRIMARY KEY,

name VARCHAR(255),

price INT,

remainingStock INT

);

-- Таблица автомобилей

CREATE TABLE Automobile (

automobileID INT PRIMARY KEY,

supplierID INT,

name VARCHAR(255),

mileage INT,

transmissionType VARCHAR(255),

engineType VARCHAR(255),

powerAutomobile INT,

price INT,

engineCapacity INT,

yearOfIssue INT,

color VARCHAR(255),

FOREIGN KEY (supplierID) REFERENCES Supplier(supplierID)

);

-- Таблица финансовых транзакций

CREATE TABLE FinancialTransaction (

financialTransactionID INT PRIMARY KEY,

clientID INT,

dateFinancialTransaction DATE,

amount INT,

typeFinancialTransaction VARCHAR(255),

FOREIGN KEY (clientID) REFERENCES Client(clientID)

);

-- Таблица обслуживания автомобилей

CREATE TABLE ServiceMaintenance (

serviceMaintenanceID INT PRIMARY KEY,

automobileID INT,

clientID INT,

sparePartID INT,

dateServiceMaintenance DATE,

price INT,

FOREIGN KEY (automobileID) REFERENCES Automobile(automobileID),

FOREIGN KEY (clientID) REFERENCES Client(clientID),

FOREIGN KEY (sparePartID) REFERENCES SparePart(sparePartID)

);

-- Таблица продаж

CREATE TABLE Sale (

saleID INT PRIMARY KEY,

clientID INT,

managerID INT,

dateSale DATE,

FOREIGN KEY (clientID) REFERENCES Client(clientID),

FOREIGN KEY (managerID) REFERENCES Manager(managerID)

);

-- Таблица проданных автомобилей

CREATE TABLE SaleAutomobile (

saleAutomobileID INT PRIMARY KEY,

saleID INT,

automobileID INT,

FOREIGN KEY (saleID) REFERENCES Sale(saleID),

FOREIGN KEY (automobileID) REFERENCES Automobile(automobileID)

);

-- Таблица проданных запчастей

CREATE TABLE SaleSparePart (

saleSparePartID INT PRIMARY KEY,

saleID INT,

sparePartID INT,

quantity INT,

FOREIGN KEY (saleID) REFERENCES Sale(saleID),

FOREIGN KEY (sparePartID) REFERENCES SparePart(sparePartID)

);

-- Таблица жалоб

CREATE TABLE Complaint (

complaintID INT PRIMARY KEY,

clientID INT,

submissionDate DATE,

status BOOLEAN,

FOREIGN KEY (clientID) REFERENCES Client(clientID)

);

-- Таблица тест-драйвов

CREATE TABLE TestDrive (

testDriveID INT PRIMARY KEY,

clientID INT,

automobileID INT,

startTime TIME,

endTime TIME,

dateTestDrive DATE,

FOREIGN KEY (clientID) REFERENCES Client(clientID),

FOREIGN KEY (automobileID) REFERENCES Automobile(automobileID)

);

3. Заполнение таблиц данными:

INSERT INTO Supplier (companyName, companyAddress) VALUES(

'VAZ',

'ul mira dom 1'

);

INSERT INTO Client (fullName, phoneNumber) VALUES(

'Фёдор Петрович',

'+7 800 555 35 35'

);

INSERT INTO Manager (fullName) VALUES(

'Николай Семеныч'

);

INSERT INTO SparePart (name, price, remainingStock) VALUES(

'Fuze',

23000,

12

);

INSERT INTO Automobile (supplierID, name, mileage, transmissionType, engineType, powerAutomobile, price, engineCapacity, yearOfIssue, color) VALUES(

2,

'VAZ 2112',

2002,

'MG',

'16V',

200,

500000,

123122,

2020,

'yellow'

);

INSERT INTO FinancialTransaction (clientID, dateFinancialTransaction, amount, typeFinancialTransaction) VALUES(

2,

'1999-01-10',

2,

'Покупка машины'

);

INSERT INTO ServiceMaintenance (automobileID, clientID, sparePartID, dateServiceMaintenance, price) VALUES(

3,

2,

2,

'2023-02-13',

6000000

);

INSERT INTO Sale (clientID, managerID, dateSale) VALUES(

0,

0,

'2025-02-20'

);

INSERT INTO SaleSparePart (saleID, sparePartID, quantity) VALUES(

1,

2,

12

);

INSERT INTO Complaint (clientID, submissionDate, status) VALUES(

2,

'2024-04-23',

true

);

INSERT INTO TestDrive (clientID, automobileID, startTime, endTime, dateTestDrive) VALUES(

2,

3,

'16:20:12',

'16:50:23',

'2020-12-10'

);

4. Изменение таблиц:

ALTER TABLE Client

ADD age INT;

ALTER TABLE SaleSparePart

ADD COLUMN date DATE;

ALTER TABLE SaleSparePart

RENAME COLUMN saleSparePartID TO id;

ALTER TABLE Sale

RENAME COLUMN datesale TO date;

ALTER TABLE automobile

ADD COLUMN ordered\_by INT;

ALTER TABLE automobile

ADD CONSTRAINT ordered\_by FOREIGN KEY (ordered\_by) REFERENCES Client(clientID)

ALTER TABLE automobile

DROP CONSTRAINT ordered\_by

Для удаления PRIMARY KEY из таблицы, для начала стоит узнать название ограничения, для этого воспользуемся командой psql \d [table\_name]

Название ограничения – client\_pkey

Теперь исполним SQL запрос:

ALTER TABLE client

DROP CONSTRAINT client\_pkey;

Для добавления первичного ключа выполним SQL запрос:

ALTER TABLE client

ADD PRIMARY KEY (clientID)

ALTER TABLE Automobile

ADD CONSTRAINT price CHECK (price > 0)

ALTER TABLE Automobile

ADD CONSTRAINT name\_unique UNIQUE (name)

ALTER TABLE client

ALTER COLUMN phoneNumber TYPE VARCHAR(20)

5. Сгенерированная реляционная модель:

Модель совпадает с изначально данной, за исключением таблиц many-to-many отношений.

6. Удаление таблиц и базы данных:

DROP TABLE IF EXISTS Supplier CASCADE;

DROP TABLE IF EXISTS Client CASCADE;

DROP TABLE IF EXISTS Manager CASCADE;

DROP TABLE IF EXISTS SparePart CASCADE;

DROP TABLE IF EXISTS Automobile CASCADE;

DROP TABLE IF EXISTS FinancialTransaction CASCADE;

DROP TABLE IF EXISTS ServiceMaintenance CASCADE;

DROP TABLE IF EXISTS Sale CASCADE;

DROP TABLE IF EXISTS SaleAutomobile CASCADE;

DROP TABLE IF EXISTS SaleSparePart CASCADE;

DROP TABLE IF EXISTS Complaint CASCADE;

DROP TABLE IF EXISTS TestDrive CASCADE;

DROP DATABASE db3\_popov\_potapovich

Также к команде удаления базы данных можно добавить параметр FORCE,

для удаления без доп. Проверок

7. Все скрипты можно найти в репозитории по ссылке из лабораторной работы №2. Файлы находятся в папке docs/PostgreSQL

Вывод: выполнив лабораторную работу №3 я научился физическому проектированию баз данных. Создал свою базу данных в СУБД pgAdmin для своего проекта, а также по реляционной модели одногруппника.

**Лабораторная работа №4.**

**Цель работы:** научиться писать правильные SQL запросы к базе данных.

**Задачи работы:** написать 10 SQL запросов по функциональным требованиям проекта; написать сложные SQL запросы, включающие в себя различные операторы, поддерживаемые выбранной базой данных.

**Ход выполнения работы:**

Функциональные требования:

1. Создание аккаунта:  
     
   Создание аккаунта может быть доступно только, если пользователь с таким login и nickname не существует, поэтому нужно написать запрос:  
     
   SELECT COUNT(\*) FROM public.users  
    WHERE login='Login' OR nickname='Nickname'  
     
   (Можно разделить на два запроса, для того, чтобы явно уточнить пользователю, что пошло не так)  
     
   Если кол-во в результате запроса равно нулю, значит мы можем создать аккаунт (валидация пароля происходит на стороне сервера)  
     
   INSERT INTO users(nickname, login, password\_hash, money)  
    VALUES (‘Nickname’, ‘Login’, ‘Password\_hash’, 1000)  
     
   password\_hash генерируется на сервере, перед отправкой запроса на сервер, money задаётся стандартный, при создании аккаунта (1000 монет)
2. Авторизация пользователя:  
     
   Авторизация пользователя происходит только тогда, когда Login и Password\_hash совпадают, password\_hash генерируется на сервере, а значит нужно проверить, есть ли пользователь с таким Login и password\_hash:  
     
   SELECT COUNT(\*) FROM users  
    WHERE login=’Login’ AND password\_hash='Password\_hash'
3. Изменение никнейма игрока:  
     
   Так как nickname должен быть уникальным, проверим, есть ли уже пользователь с таким nickname’ом:  
     
   SELECT COUNT(\*) FROM users  
    WHERE nickname=’NewNickname’  
     
   Если таких пользователей нет, то изменим пользователя, для идентификации пользователя, используя текущий Nickname:  
     
   UPDATE users SET nickname=’NewNickname’  
    WHERE nickname=’Nickname’  
     
   (по факту, на сервере будет еще аутентификация пользователя, используя id, так что появится еще одно обращения к базе. Далее таких аутентификаций будет много, о них я упоминать не буду)
4. Покупка карт:  
     
   Сначала, проверим, есть ли такая карта уже у игрока:  
     
   1) SELECT \* FROM sell\_items   
    WHERE id=’sell\_item\_id’  
     
     
   2) SELECT COUNT(\*) FROM cards  
    WHERE user\_id=’user\_id’ AND card\_id=’card\_id’  
     
   Для покупки, сначала, мы должны убедиться, что хватает валюты у пользователя, для этого сначала получим кол-во его валюты.  
     
   SELECT money FROM users  
    WHERE id=’id’  
   (Поле id пользователя нам уже известно, благодаря аутентификации)  
     
   Далее, если валюты достаточно, отнимем у пользователя сумму валюты, равную цене покупаемой карты и создадим карту в базе данных. (Продаваемые предметы (sell\_items) мы уже получали заранее, и знаем какой у нее id)  
     
   1) UPDATE users SET money=money-‘cards\_cost’   
    WHERE id=’user\_id’  
     
   2) INSERT INTO cards (user\_id, card\_level, card\_power, card\_id)  
    VALUES (‘user\_id’, ‘1’, ‘card\_power’, ‘card\_id’)  
     
   Card\_power нам известно с клиента игры. (Знаю, не совсем хорошо, правильней было бы в БД хранить общую информацию о картах. В моем случае характеристики карт хранятся на клиенте, БД знает только об их id, можно в выводе семестровки занести в TODO блок)
5. Сбор статистики:  
     
   После создания аккаунта, к нему создаётся строка в таблице statistics (можно сделать по триггеру, либо на бэке. Лучше по триггеру, так как мы избавимся от лишнего обращения к БД и ответ от бэка пользователь будет получать быстрее, т.к. бэку всё равно на стек задач БД, после выполнения запроса о создании пользователя)  
     
   Пользователь может:  
    1) Обновить статистику (в основном после сыгранного матча)  
    2) Просмотреть статистику (по нажатию на кнопку, открывается pop-up элемент со статистикой)  
     
   Обновление статистики:  
     
   UPDATE statistics   
    SET win=win+’result’,   
    matches\_count=matches\_count+1,  
    max\_rating=MAX(max\_rating, max\_rating+'points'),  
    current\_rating=current\_rating+’points’  
    WHERE user\_id=’user\_id’  
     
   Просмотр статистики:  
     
   SELECT \* FROM statistics  
    WHERE user\_id=’user\_id’
6. Создание кланов:  
     
   Была создана колонка owner\_id в таблице clan, являющаяся внешним ключом на таблицу users. (Была создана для удобства удаления клана).  
     
   Запрос на создание клана:  
     
   INSERT INTO clan (name, description, avatar, owner\_id)  
    VALUES ('clan\_name', 'clan\_description', '1', 20778)  
     
   Запрос на просмотр характеристик клана:  
     
   SELECT \* FROM clan  
    WHERE id = ‘clan\_id’  
     
   clan\_id можно получить из таблицы users, зная текущего пользователя.  
     
   Запрос на просмотр участников клана:  
     
   SELECT \* FROM users  
    WHERE clan\_id=’clan\_id’  
     
     
     
     
   Запрос на изменение имени клана (должно быть уникальным)  
     
   Проверка на уникальность:  
     
   SELECT COUNT(\*) FROM clan  
    WHERE name=’new\_name’  
     
   Если имя уникальное:  
     
   UPDATE clan SET name=’new\_name'  
    WHERE id=’clan\_id’  
     
   Удаление клана:  
   Состоит из двух этапов: удаление отношений из users – clan\_id и удаление clan\_actions – clan\_id по текущему id клана; удаление строки из clans:  
     
   UPDATE users

SET clan\_id=NULL   
 WHERE clan\_id=’clan\_id’  
  
DELETE FROM clan\_actions

WHERE clan\_id=’clan\_id’  
  
DELETE FROM clan  
 WHERE id=’clan\_id’

(реализован CRUD)

2. UPDATE в разных таблицах, с WHERE, можно условно, например, изменить заранее созданные некорректные данные (5 шт.)  
  
11. Обновление описания клана

UPDATE clans

SET description = 'Элитный клан чемпионов'

WHERE name = 'Dragons';

12. Обнуление рейтинга людям по условию

UPDATE statistics

SET current\_rating = 100, max\_rating = 100

WHERE user\_id IN (

SELECT id FROM users

WHERE current\_rating < 50 OR matches\_count = 0

);

13. Увеличение денег у пользователей без клана

UPDATE users

SET money = money + 500

WHERE clan\_id IS NULL AND money < 1000;

14. Увеличение счетчика побед

UPDATE statistics

SET win\_count = win\_count + 1

WHERE user\_id = 20;

15. Обновление стоимости карты в магазине

UPDATE sell\_items

SET cost = 200

WHERE card\_id = 7;

3. DELETE в разных таблицах, с WHERE, можно условно, например, удалить заранее созданные некорректные данные (5 шт.)

16. Удаление пользователя

DELETE FROM users

WHERE id = 15;

17. Удаление информации о старых магазинах

DELETE FROM shops

WHERE update\_date < '2023-01-01';

18. Удаление слишком дорогих предметов

DELETE FROM sell\_items

WHERE cost > 1000;

19. Удаление клана по имени.

DELETE FROM clans

WHERE name = 'СтарыйКлан';

20. Удаление неиспользуемых предметов в магазине

DELETE FROM sell\_items

WHERE id NOT IN (SELECT sell\_item\_id FROM shops\_sell\_items);

4. SELECT, DISTINCT, WHERE, AND/OR/NOT, IN, BETWEEN, различная работа с датами и числами, преобразование данных, IS NULL, AS для таблиц и столбцов и др. в различных вариациях (15 шт. +)

21. Получение уникальных карт выше какого-то уровня

SELECT DISTINCT card\_id

FROM cards

WHERE level > 3;

22. Получить ник и логин пользователей, у кого больше 50 побед

SELECT nickname, login

FROM users

WHERE id IN (SELECT user\_id FROM statistics WHERE win\_count > 50);

23. Получение id и месяца создания магазина, созданных в 2023 году

SELECT id, EXTRACT(MONTH FROM update\_date) AS month

FROM shops

WHERE EXTRACT(YEAR FROM update\_date) = 2023;

24. Получение id и силы карты приведенной к целочисленному типу

SELECT id, CAST(power AS INT) AS int\_power

FROM cards

25. Получение пользователей, которые не в клане

SELECT id, nickname

FROM users

WHERE clan\_id IS NULL;

26. Получение рейтинга и разницы с максимальным

SELECT id, current\_rating, (max\_rating - current\_rating) AS rating\_diff

FROM statistics

WHERE current\_rating > 100;

27. Получение строки по шаблону

SELECT id, description

FROM clan\_actions

WHERE description LIKE '%приобрёл%';

28. Получение количества пользователей

SELECT COUNT(\*) AS total\_users

FROM users;

29. Получение пользователей в промежутке по деньгам

SELECT nickname, money

FROM users

WHERE money BETWEEN 1000 AND 5000;

30. Получение магазинов за последние 7 дней

SELECT id

FROM shops

WHERE update\_date > CURRENT\_DATE - INTERVAL '7 days';

31. Получение пользователей с рейтингом в диапазоне

SELECT id, current\_rating

FROM statistics

WHERE current\_rating BETWEEN 100 AND 200;

32. Получение пользователей, у которых больше или равно 5 побед

SELECT id, nickname

FROM users

WHERE id NOT IN (SELECT user\_id FROM statistics WHERE win\_count < 5);

33. Получение округленной силы карт

SELECT id, ROUND(power) AS rounded\_power

FROM cards;

34. Получение пользователей, длина ника которых больше 5 символов

SELECT nickname

FROM users

WHERE LENGTH(nickname) > 5;

35. Получение свободных комнат

SELECT id

FROM matchmakers\_rooms

WHERE peer\_id IS NULL;

5. LIKE и другая работа со строками (5-7 шт.+)

36. Поиск никнеймов из 5 символов

SELECT nickname

FROM users

WHERE nickname LIKE '\_\_\_\_\_';

37. Найти кланы, в описании которых есть слово "сильнейший"

SELECT name, description

FROM clans

WHERE description LIKE '%сильнейший%';

38. Найти ники, содержащие символ '\_’

SELECT nickname

FROM users

WHERE nickname LIKE '%\\_%' ESCAPE '\';

39. Получить по 3 первых символа ника

SELECT LEFT(nickname, 3) AS nickname\_prefix

FROM users;

40. Заменить слова в описании клана и получить измененные строки

SELECT id, REPLACE(description, 'старый', 'новый') AS updated\_description

FROM clan\_actions;

6. SELECT INTO или INSERT SELECT, что поддерживается СУБД (2-3 шт.). Для использования запроса INSERT SELECT вначале можно создать новую тестовую таблицу или несколько, в которые будут скопированы данные из существующих таблиц с помощью данного запроса. Код создания таблиц также приложить в лабораторную работу

Таблица для копирования списка пользователей

CREATE TABLE user\_copy (

id INT,

nickname VARCHAR(20)

);

Таблица для копирования карт

CREATE TABLE card\_copy (

card\_id INT,

level INT

);

41. Копирование пользователей в другую таблицу

INSERT INTO user\_copy (id, nickname)

SELECT id, nickname FROM users;

42. Копирование карт в другую таблицу только выше 3го уровня

INSERT INTO card\_copy (card\_id, level)

SELECT card\_id, level FROM cards WHERE level > 3;

43. Создание таблицы пользователей, деньги которых превышают 2000

SELECT id, nickname INTO rich\_users FROM users WHERE money > 2000;

7. JOIN: INNER, OUTER (LEFT, RIGHT, FULL), CROSS, NATURAL, разных, в различных вариациях, несколько запросов с более, чем одним JOIN. Обязательно сделать запросы со связями многие ко многим, если таких нет, придумать и добавить в свою БД хотя бы одну (15 шт.+)

44. Получение имени пользователя и его количества побед

SELECT u.nickname, s.win\_count

FROM users u

INNER JOIN statistics s ON u.id = s.user\_id;

45.Получение пользователей и их имени кланов

SELECT u.nickname, c.name AS clan\_name

FROM users u

LEFT JOIN clans c ON u.clan\_id = c.id;

46. Получение кланов и их владельцов (ники)

SELECT c.name, u.nickname AS owner

FROM users u

RIGHT JOIN clans c ON u.id = c.owner\_id;

47. Получение всех кланов и всех игроков

SELECT u.nickname, c.name AS clan\_name

FROM users u

FULL JOIN clans c ON u.clan\_id = c.id;

48. Все возможные комбинации игроков и кланов

SELECT u.nickname, c.name AS clan\_name

FROM users u

CROSS JOIN clans c

LIMIT 100;

49. Получение игроков и количества их побед

SELECT nickname, win\_count

FROM users

NATURAL JOIN statistics;

50. Получение id матча и его хоста и пира

SELECT m.id, h.nickname AS host, p.nickname AS peer

FROM match\_statistics m

JOIN users h ON m.host\_id = h.id

JOIN users p ON m.peer\_id = p.id;

51. Получение среднего рейтинга по кланам

SELECT c.name, AVG(s.current\_rating) AS avg\_rating

FROM clans c

JOIN users u ON c.id = u.clan\_id

JOIN statistics s ON u.id = s.user\_id

GROUP BY c.name;

52. Получить игроков с одинаковым количеством денег

SELECT a.nickname AS user1, b.nickname AS user2, a.money

FROM users a

JOIN users b ON a.money = b.money AND a.id < b.id;

53. Получение карт пользователей со стоимостью более 300.

SELECT u.nickname, c.card\_id, c.level

FROM users u

JOIN cards c ON u.id = c.user\_id

JOIN sell\_items s ON c.card\_id = s.card\_id

WHERE s.cost > 300;

54. Получение карт в текущем магазине

SELECT si.card\_id, si.cost, s.update\_date

FROM sell\_items si

INNER JOIN shops\_sell\_items ssi ON si.id = ssi.sell\_item\_id

INNER JOIN shops s ON ssi.shop\_id = s.id

WHERE s.update\_date = (SELECT MAX(update\_date) FROM shops);

55. Получить пользователей без кланов

SELECT u.nickname

FROM users u

LEFT JOIN clans c ON u.clan\_id = c.id

WHERE c.id IS NULL;

56. Получить информацию о матче

SELECT m.id, h.nickname AS host, p.nickname AS peer,

m.result, m.duration, r.state

FROM match\_statistics m

JOIN users h ON m.host\_id = h.id

JOIN users p ON m.peer\_id = p.id

JOIN matchmakers\_rooms r ON m.room\_id = r.id;

57. Получить количество карт пользователей

SELECT u.nickname, COUNT(c.id) AS cards\_count

FROM users u

LEFT JOIN cards c ON u.id = c.user\_id

GROUP BY u.nickname;

58. Получить полную информацию о кланах

SELECT c.name,

COUNT(u.id) AS members,

AVG(s.current\_rating) AS avg\_rating,

SUM(u.money) AS total\_money

FROM clans c

LEFT JOIN users u ON c.id = u.clan\_id

LEFT JOIN statistics s ON u.id = s.user\_id

GROUP BY c.name;

59. Получить товары, находящиеся хоть в одном магазине, стоимостью более 300

SELECT DISTINCT si.card\_id, si.cost

FROM sell\_items si

JOIN shops\_sell\_items ssi ON si.id = ssi.sell\_item\_id

WHERE si.cost > 300;

8. GROUP BY (некоторые с HAVING), с LIMIT, ORDER BY (ASC|DESC) вместе с COUNT, MAX, MIN, SUM, AVG в различных вариациях, можно по отдельности (15 шт.+)

60. Количество пользователей в каждом клане

SELECT clan\_id, COUNT(\*) AS users\_count

FROM users

WHERE clan\_id IS NOT NULL

GROUP BY clan\_id

ORDER BY users\_count DESC;

61. Получить средний рейтинг кланов, у которых более 5 пользователей

SELECT c.name, AVG(s.current\_rating) AS avg\_rating

FROM users u

JOIN clans c ON u.clan\_id = c.id

JOIN statistics s ON u.id = s.user\_id

GROUP BY c.name

HAVING COUNT(u.id) >= 5

ORDER BY avg\_rating DESC;

62. Получить 5 самых дорогих карт в магазинах

SELECT card\_id, MAX(cost) AS max\_price

FROM sell\_items

GROUP BY card\_id

ORDER BY max\_price DESC

LIMIT 5;

63. Получить количество карт каждого уровня у пользователя

SELECT level, COUNT(\*) AS cards\_count

FROM cards

WHERE user\_id = 10

GROUP BY level

ORDER BY level ASC;

64. Получить 10 игроков с максимальными уровнями карт

SELECT u.nickname, COUNT(c.id) AS cards\_count

FROM users u

JOIN cards c ON u.id = c.user\_id

GROUP BY u.id

ORDER BY cards\_count DESC

LIMIT 10;

65. Средняя стоимость карт в магазине по уровням

SELECT c.level, AVG(si.cost)::numeric(10,2) AS avg\_price

FROM cards c

JOIN sell\_items si ON c.card\_id = si.card\_id

GROUP BY c.level

ORDER BY c.level;

66. Получить кланы с суммарным количеством денег более 10000

SELECT c.name, SUM(u.money) AS total\_money

FROM clans c

JOIN users u ON c.id = u.clan\_id

GROUP BY c.name

HAVING SUM(u.money) > 10000

ORDER BY total\_money DESC;

67. Получить количество матчей по результатам

SELECT result, COUNT(\*) AS matches\_count

FROM match\_statistics

GROUP BY result

ORDER BY matches\_count DESC;

68. Получить 5 игроков с максимальным рейтингом в кланах

SELECT c.name, u.nickname, MAX(s.current\_rating) AS max\_rating

FROM clans c

JOIN users u ON c.id = u.clan\_id

JOIN statistics s ON u.id = s.user\_id

GROUP BY c.name, u.nickname

ORDER BY max\_rating DESC

LIMIT 5;

69. Получить среднее количество ходов в матчах по кланам

SELECT c.name, AVG(m.duration)::numeric(10,2) AS avg\_duration

FROM match\_statistics m

JOIN users h ON m.host\_id = h.id

JOIN clans c ON h.clan\_id = c.id

GROUP BY c.name

ORDER BY avg\_duration DESC;

70. Получить количество карт с разным id (типом)

SELECT card\_id, COUNT(\*) AS total\_cards

FROM cards

GROUP BY card\_id

ORDER BY total\_cards DESC;

71. Получить пользователей с количеством побед выше среднего значения

SELECT u.nickname, s.win\_count

FROM users u

JOIN statistics s ON u.id = s.user\_id

WHERE s.win\_count > (SELECT AVG(win\_count) FROM statistics)

ORDER BY s.win\_count DESC;

72. Получить 3 самые популярные карты в колодах игроков

SELECT card\_id, COUNT(\*) AS in\_decks\_count

FROM cards

WHERE deck\_id IS NOT NULL

GROUP BY card\_id

ORDER BY in\_decks\_count DESC

LIMIT 3;

73. Получить топ 3 клана по количеству побед

SELECT c.name, SUM(s.win\_count) AS total\_wins

FROM clans c

JOIN users u ON c.id = u.clan\_id

JOIN statistics s ON u.id = s.user\_id

GROUP BY c.name

ORDER BY total\_wins DESC

LIMIT 3;

74. Получить количество карт по уровням

SELECT level, COUNT(\*) AS cards\_count

FROM cards

GROUP BY level

ORDER BY level;

9. UNION, EXCEPT, INTERSECT, что поддерживается СУБД (3-5 шт.)

75. Получить пользователей, которые никогда не были хостом в матче

SELECT id, nickname FROM users

EXCEPT

SELECT DISTINCT host\_id, '' FROM match\_statistics;

76. Получить карты, которые есть и у игроков и в магазине

SELECT DISTINCT card\_id FROM cards

INTERSECT

SELECT card\_id FROM sell\_items;

77. Получение ников игроков и названий кланов

SELECT nickname AS name FROM users

UNION

SELECT name FROM clans

ORDER BY name;

10. Вложенные SELECT с GROUP BY, ALL, ANY, EXISTS (3-5 шт.)

78. Получить имена пользователей и их количество карт

SELECT nickname,

(SELECT COUNT(\*) FROM cards WHERE user\_id = users.id) AS card\_count

FROM users

ORDER BY card\_count DESC;

79. Получить пользователя с максимальным количеством денег

SELECT nickname, money

FROM users

WHERE money >= ALL (SELECT money FROM users WHERE money IS NOT NULL);

80. Получить имена кланов, в которых есть хоть один игрок с текущим рейтингом более 1000

SELECT name

FROM clans

WHERE id = ANY (SELECT clan\_id FROM users

WHERE id IN (SELECT user\_id FROM statistics

WHERE current\_rating > 1000));

81. Получить имена пользователей, у которых есть карта 5 уровня

SELECT nickname

FROM users u

WHERE EXISTS (SELECT 1 FROM cards

WHERE user\_id = u.id AND level = 5);

11. GROUP\_CONCAT и другие разнообразные функции SQL (2-3 шт.)

82. Получить список участников клана в одной строке

SELECT

c.name AS clan\_name,

STRING\_AGG(u.nickname, ', ' ORDER BY u.nickname) AS members

FROM clans c

JOIN users u ON c.id = u.clan\_id

GROUP BY c.name

HAVING COUNT(u.id) > 0;

83. Получение всех карт пользователя в формате Json

SELECT

u.nickname,

JSON\_AGG(

JSON\_BUILD\_OBJECT(

'card\_id', c.card\_id,

'level', c.level,

'power', c.power

)

) AS cards\_json

FROM users u

JOIN cards c ON u.id = c.user\_id

GROUP BY u.id

LIMIT 5;

12. Запросы с WITH (2-3 шт.)

84. Получение 5 пользователей и их карт

WITH user\_cards AS (

SELECT user\_id, COUNT(\*) as card\_count

FROM cards

GROUP BY user\_id

)

SELECT u.nickname, uc.card\_count

FROM users u

JOIN user\_cards uc ON u.id = uc.user\_id

ORDER BY uc.card\_count DESC

LIMIT 5;

85. Получение пользователя с количеством денег выше среднего значение

WITH avg\_money AS (

SELECT AVG(money) as average

FROM users

)

SELECT nickname, money

FROM users, avg\_money

WHERE money > average

ORDER BY money DESC;

13. Запросы со строковыми функциями СУБД, с функциями работы с датами временем (форматированием дат), с арифметическими функциями (5-7 шт.)

86.Получение 5 пользователей и значение их кланов

SELECT CONCAT(nickname, ' (', COALESCE(clan\_id::TEXT, 'без клана'), ')') AS player\_info

FROM users

LIMIT 5;

87. Получение форматированной к строке даты обновления магазина и его id

SELECT id, TO\_CHAR(update\_date, 'DD.MM.YYYY HH24:MI') AS formatted\_date

FROM shops

LIMIT 3;

88. Получение разницы между текущей датой и датой обновления магазина для 3х магазинов

SELECT id, CURRENT\_DATE - update\_date::DATE AS days\_since\_update

FROM shops

LIMIT 3;

89. Получение увлеченных денег пользователей на 10%

SELECT nickname, money, (money \* 1.1)::INT AS new\_balance

FROM users

LIMIT 5;

90. Получение округленных значений сил карт для 5 карт

SELECT card\_id, ROUND(power, 1) AS rounded\_power

FROM cards

LIMIT 5;

91. Получение стоимости всех продаваемых предметов

SELECT SUM(cost) AS total\_shop\_value

FROM sell\_items;

92. Получение средней силы карт по уровням

SELECT level, AVG(power)::numeric(10,2) AS avg\_power

FROM cards

GROUP BY level

ORDER BY level;

93. Расчёт стоимости улучшения 5 карт

SELECT id, card\_id, level, (level \* 100) AS upgrade\_cost

FROM cards

WHERE user\_id=’2’

LIMIT 5;

94. Получить общее количество денег пользователей

SELECT SUM(money) AS total\_money\_in\_system

FROM users;

95. Получить пользователей с количеством денег в диапазоне 10% от среднего значения

SELECT nickname, money

FROM users

WHERE money BETWEEN

(SELECT AVG(money)\*0.9 FROM users)

AND

(SELECT AVG(money)\*1.1 FROM users);

14. Сложные запросы, входящие в большинство групп выше, т.е. SELECT ... JOIN ... JOIN ... WHERE ... GROUP BY ... ORDER BY ... LIMIT ...; (5-7 шт. +), можно написать больше вместо простых.

96. Получение всех результатов матчей пользователей в одном клане (Cross)

SELECT

c.name AS clan\_name,

h.nickname AS host,

p.nickname AS peer,

m.result,

m.duration

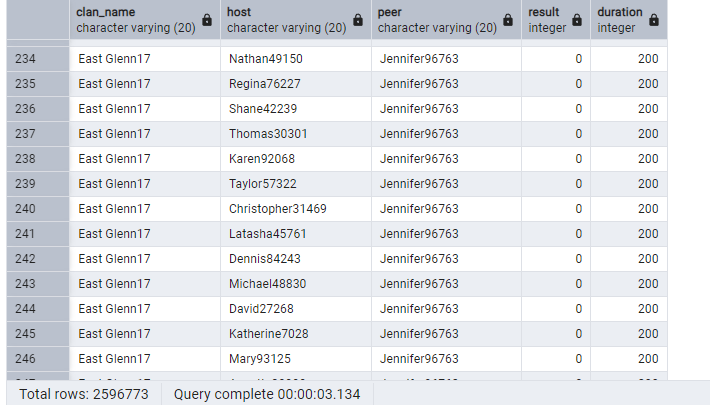
FROM match\_statistics m

CROSS JOIN users h

JOIN users p ON m.peer\_id = p.id

JOIN clans c ON h.clan\_id = c.id AND p.clan\_id = c.id

ORDER BY m.duration DESC;



97. Получение подробной информации о 1000 пользователях

SELECT

u.nickname,

c.name AS clan\_name,

s.win\_count,

s.current\_rating,

(SELECT COUNT(\*) FROM cards WHERE user\_id = u.id) AS total\_cards,

(SELECT COUNT(\*) FROM decks WHERE user\_id = u.id) AS decks\_count,

(SELECT COUNT(\*) FROM match\_statistics

WHERE host\_id = u.id OR peer\_id = u.id) AS total\_matches

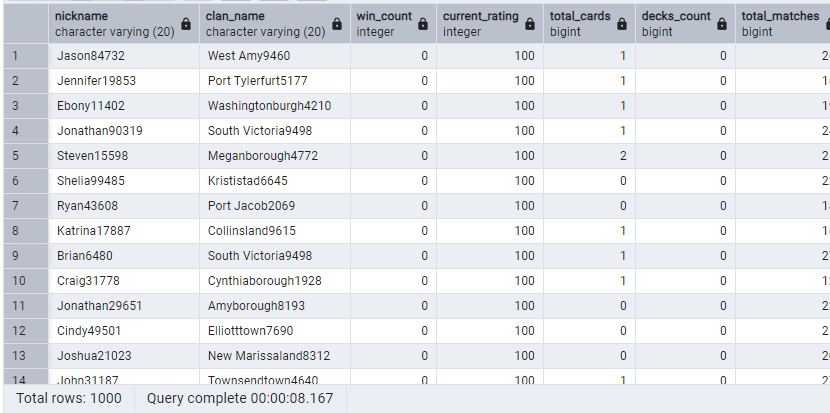
FROM users u

LEFT JOIN clans c ON u.clan\_id = c.id

JOIN statistics s ON u.id = s.user\_id

ORDER BY s.win\_count DESC, s.current\_rating DESC

LIMIT 1000



98. Топ игроков по рейтингу, среди тех, кто входит в клан со средним рейтингом более 1200

WITH TopClans AS (

SELECT c.id

FROM clans c

INNER JOIN users u ON c.id = u.clan\_id

LEFT JOIN statistics s ON u.id = s.user\_id

GROUP BY c.id

HAVING AVG(s.current\_rating) > 1200

)

SELECT u.nickname, AVG(s.current\_rating) AS average\_rating

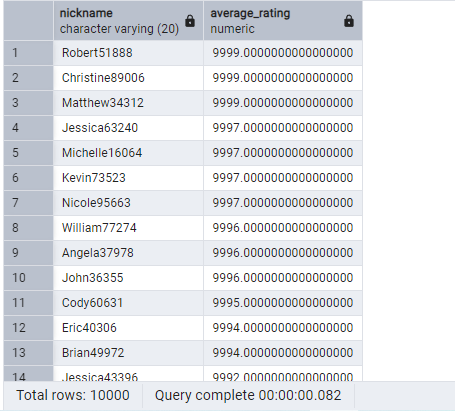
FROM users u

INNER JOIN statistics s ON u.id = s.user\_id

WHERE EXISTS (SELECT \* FROM TopClans tc WHERE tc.id = u.clan\_id)

GROUP BY u.nickname

ORDER BY average\_rating DESC



99. Средняя продолжительность матчей для хоста и гостя (пира)

SELECT

host\_user.nickname AS host\_nickname,

AVG(ms.duration) FILTER (WHERE ms.host\_id = host\_user.id) AS avg\_host\_duration,

guest\_user.nickname AS guest\_nickname,

AVG(ms.duration) FILTER (WHERE ms.peer\_id = guest\_user.id) AS avg\_guest\_duration

FROM match\_statistics ms

LEFT JOIN users host\_user ON ms.host\_id = host\_user.id

LEFT JOIN users guest\_user ON ms.peer\_id = guest\_user.id

GROUP BY host\_user.nickname, guest\_user.nickname

ORDER BY avg\_host\_duration DESC, avg\_guest\_duration DESC;



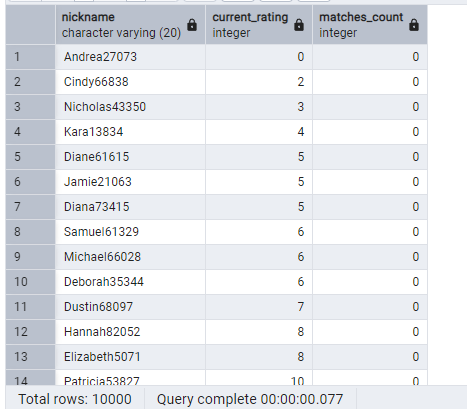
100. Получение топа игроков по наименьшему количеству матчей

SELECT u.nickname, s.current\_rating, s.matches\_count

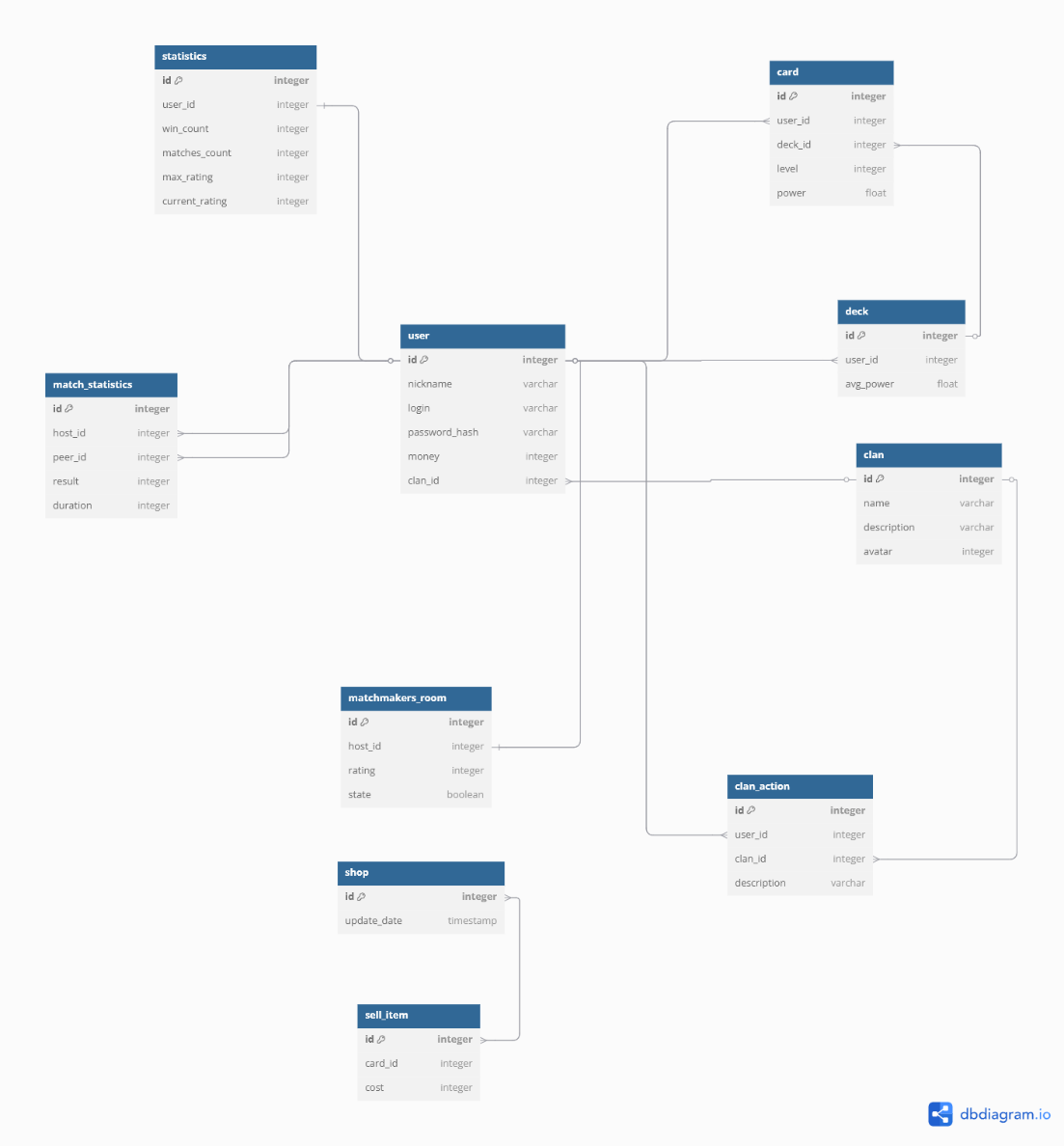
FROM users u

INNER JOIN statistics s ON u.id = s.user\_id

ORDER BY s.current\_rating ASC, s.matches\_count DESC



**Реляционная модель из лабораторно работы №3:**



**Вывод:** в лабораторной работе я научился делать запросы к базе данных используя различные запросы. Научился объединять запросы в длинные цепочки.

**Лабораторная работа №5**

**Цель работы:** Научиться работать с индексами, хранимыми процедурами, представлениями и триггерами.

**Задачи работы:** Оптимизировать сложные запросы из лаб. работы №4; создать свои процедуры, представления и триггеры

**Ход выполнения работы:**

1. Были добавлены по 10000+ записей в основные таблицы.

2. Замер времени выполнения.

1) Средняя продолжительность матчей для хоста и гостя (пира)

SELECT

host\_user.nickname AS host\_nickname,

AVG(ms.duration) FILTER (WHERE ms.host\_id = host\_user.id) AS avg\_host\_duration,

guest\_user.nickname AS guest\_nickname,

AVG(ms.duration) FILTER (WHERE ms.peer\_id = guest\_user.id) AS avg\_guest\_duration

FROM match\_statistics ms

LEFT JOIN users host\_user ON ms.host\_id = host\_user.id

LEFT JOIN users guest\_user ON ms.peer\_id = guest\_user.id

GROUP BY host\_user.nickname, guest\_user.nickname

ORDER BY avg\_host\_duration DESC, avg\_guest\_duration DESC;



Время выполнения 480 мс.

2) Топ игроков по рейтингу, среди тех, кто входит в клан со средним рейтингом более 1200

WITH TopClans AS (

SELECT c.id

FROM clans c

INNER JOIN users u ON c.id = u.clan\_id

LEFT JOIN statistics s ON u.id = s.user\_id

GROUP BY c.id

HAVING AVG(s.current\_rating) > 1200

)

SELECT u.nickname, AVG(s.current\_rating) AS average\_rating

FROM users u

INNER JOIN statistics s ON u.id = s.user\_id

WHERE EXISTS (SELECT \* FROM TopClans tc WHERE tc.id = u.clan\_id)

GROUP BY u.nickname

ORDER BY average\_rating DESC



Время выполнения 85 мс.

3) Получение подробной информации о 1000 пользователях

SELECT

u.nickname,

c.name AS clan\_name,

s.win\_count,

s.current\_rating,

(SELECT COUNT(\*) FROM cards WHERE user\_id = u.id) AS total\_cards,

(SELECT COUNT(\*) FROM decks WHERE user\_id = u.id) AS decks\_count,

(SELECT COUNT(\*) FROM match\_statistics

WHERE host\_id = u.id OR peer\_id = u.id) AS total\_matches

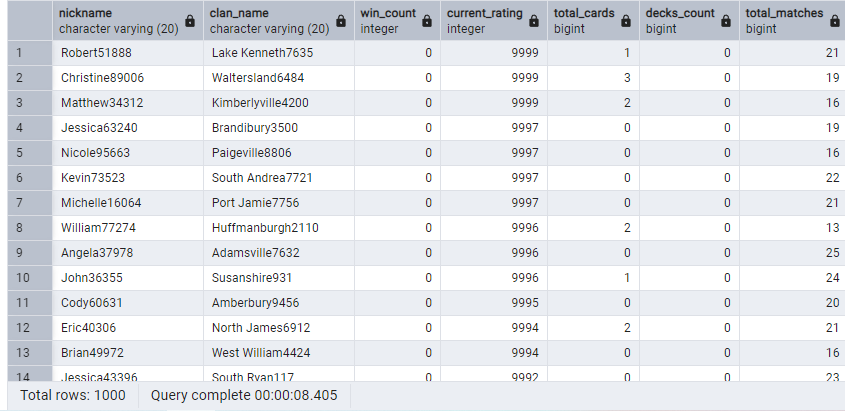
FROM users u

LEFT JOIN clans c ON u.clan\_id = c.id

JOIN statistics s ON u.id = s.user\_id

ORDER BY s.win\_count DESC, s.current\_rating DESC

LIMIT 1000



Время выполнения 8.405 сек.

4) Получение всех результатов матчей пользователей в одном клане (Cross)

SELECT

c.name AS clan\_name,

h.nickname AS host,

p.nickname AS peer,

m.result,

m.duration

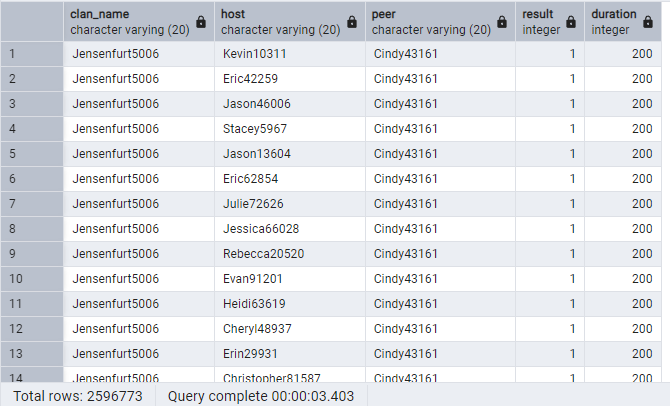
FROM match\_statistics m

CROSS JOIN users h

JOIN users p ON m.peer\_id = p.id

JOIN clans c ON h.clan\_id = c.id AND p.clan\_id = c.id

ORDER BY m.duration DESC;



Время выполнения 3.403 сек.

5) Получить список участников клана в одной строке

SELECT

c.name AS clan\_name,

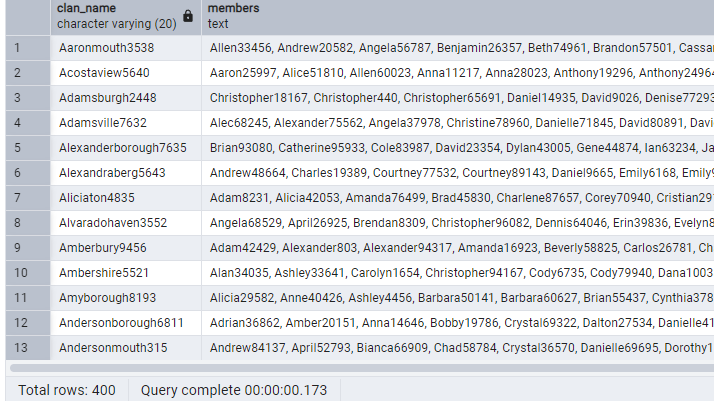
STRING\_AGG(u.nickname, ', ' ORDER BY u.nickname) AS members

FROM clans c

JOIN users u ON c.id = u.clan\_id

GROUP BY c.name

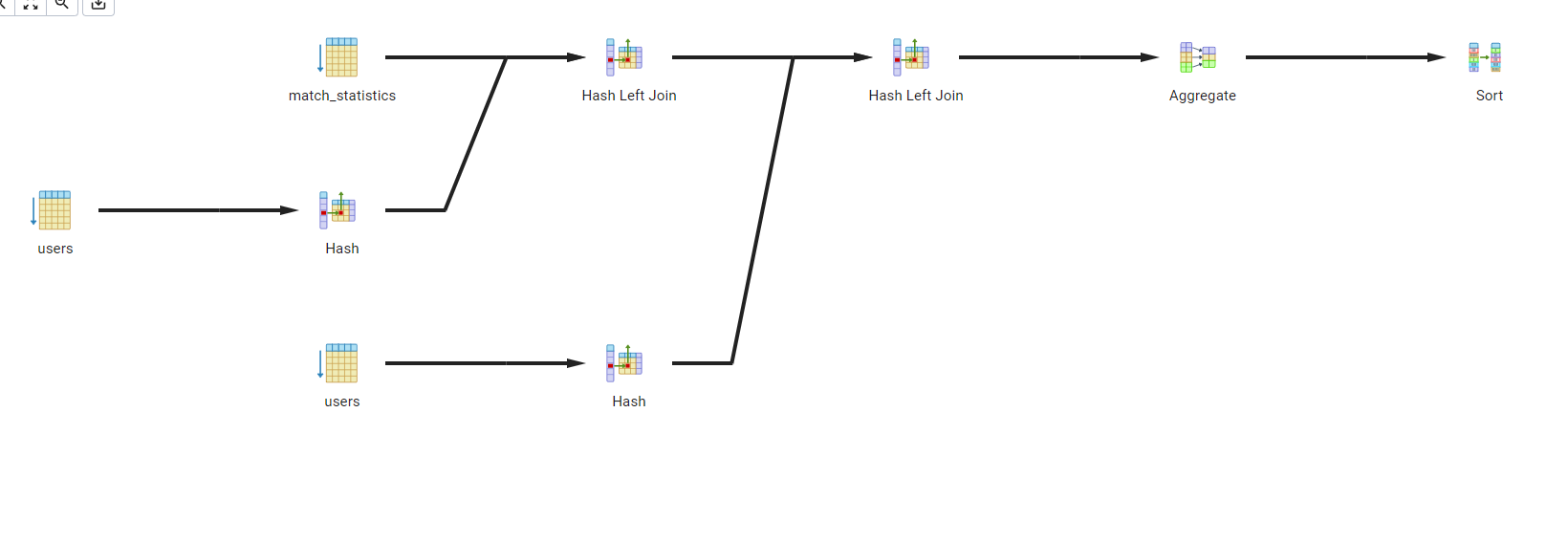
HAVING COUNT(u.id) > 0;

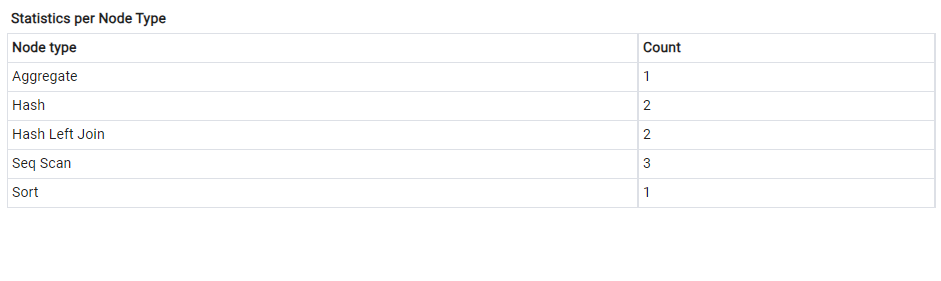


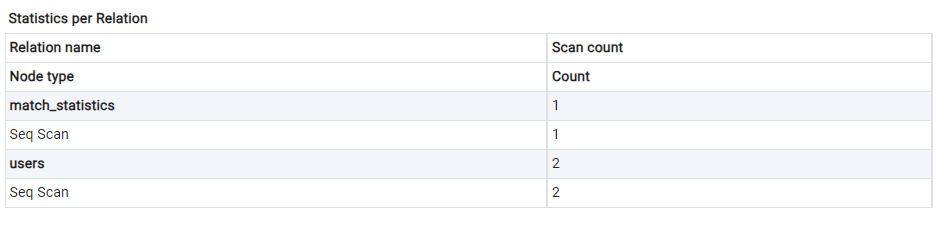
Время выполнения 173 мс.

3. Анализ скорости выполнения

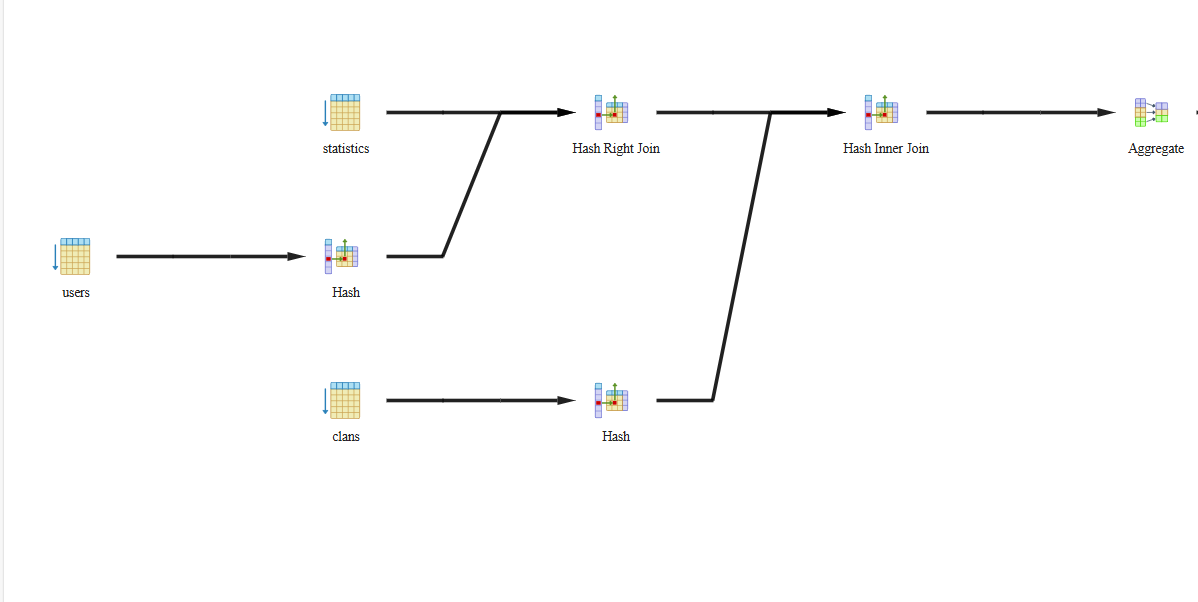
1)

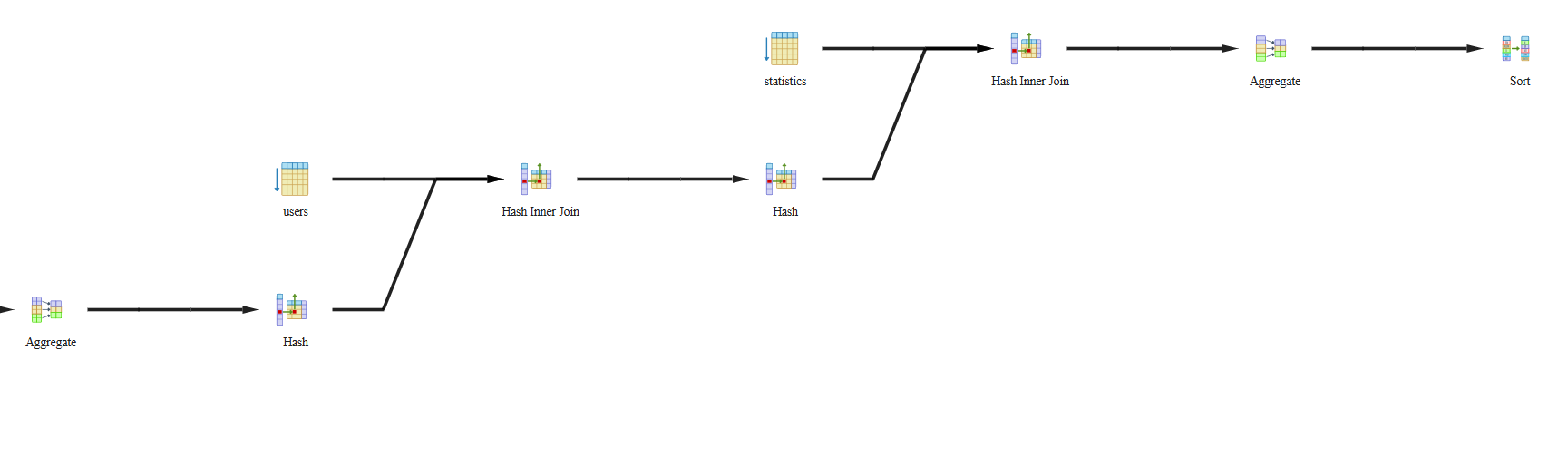




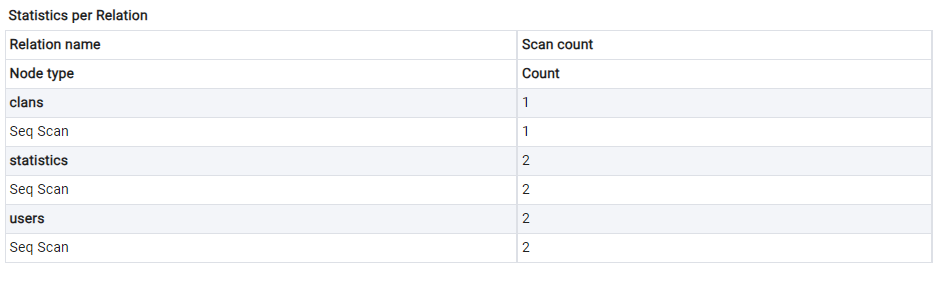


2)

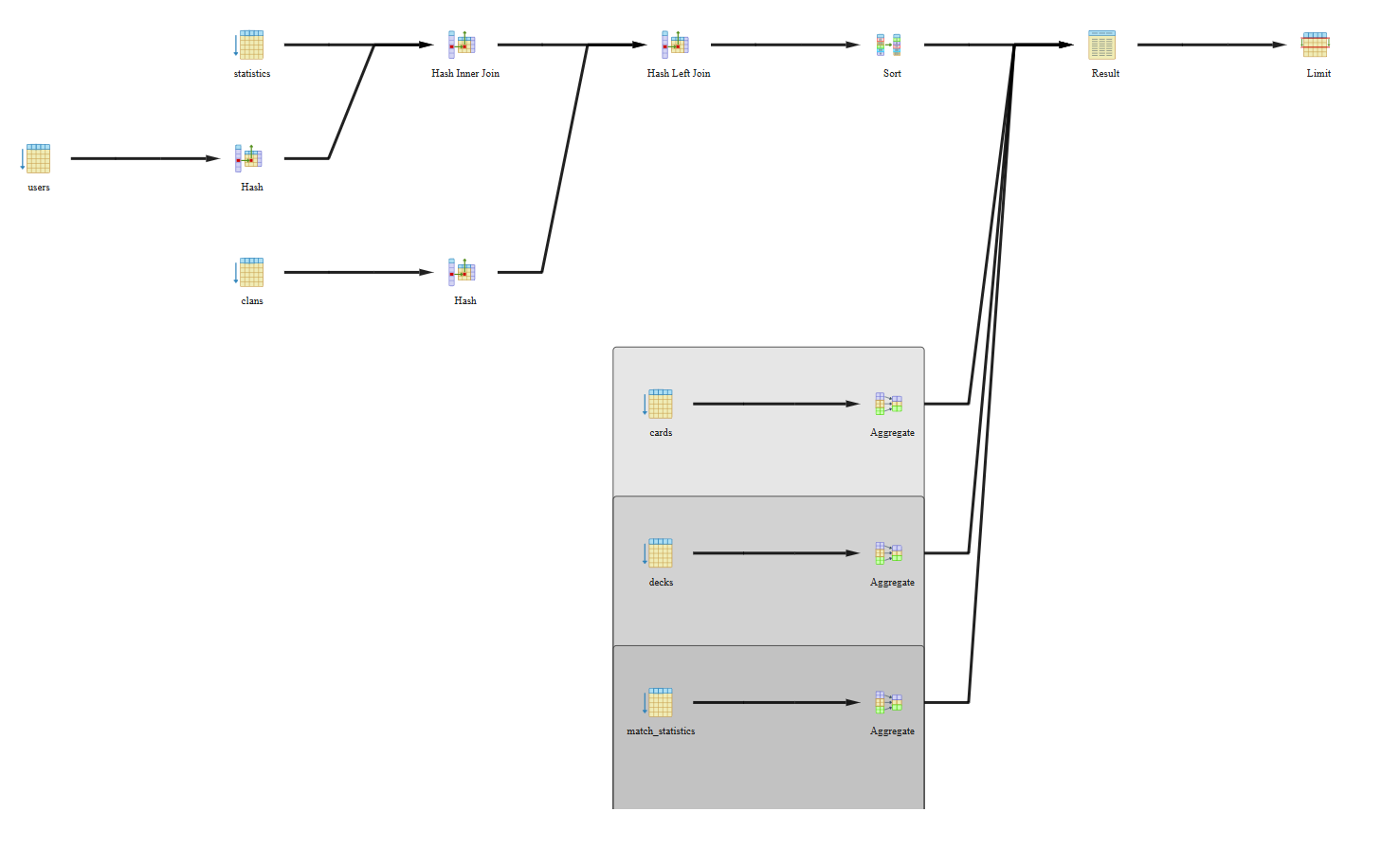


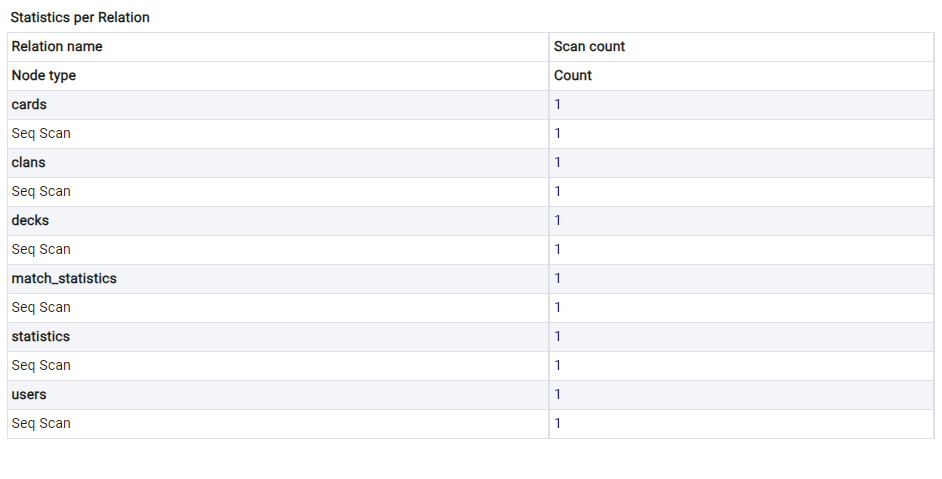




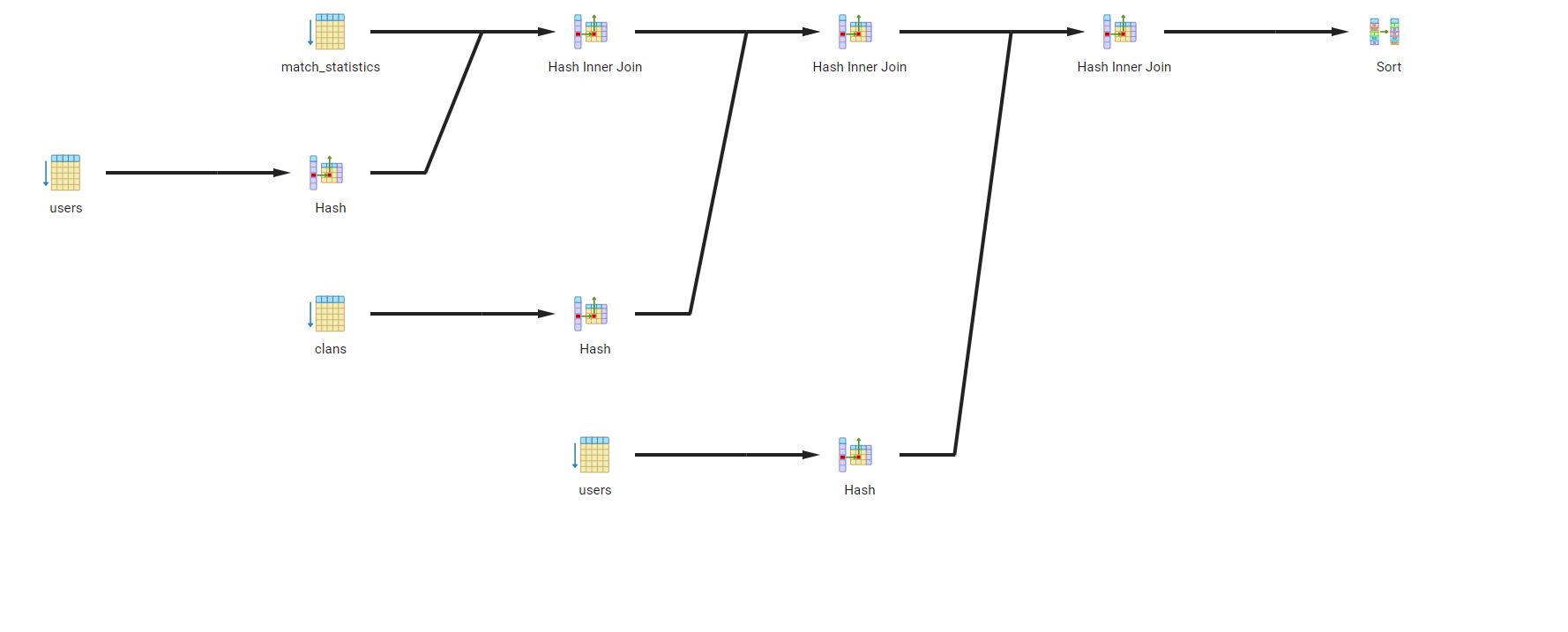


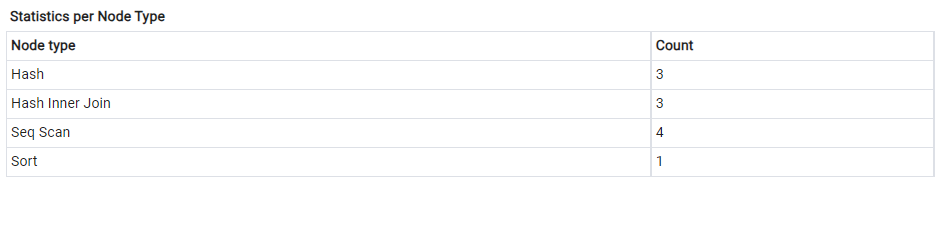
3)

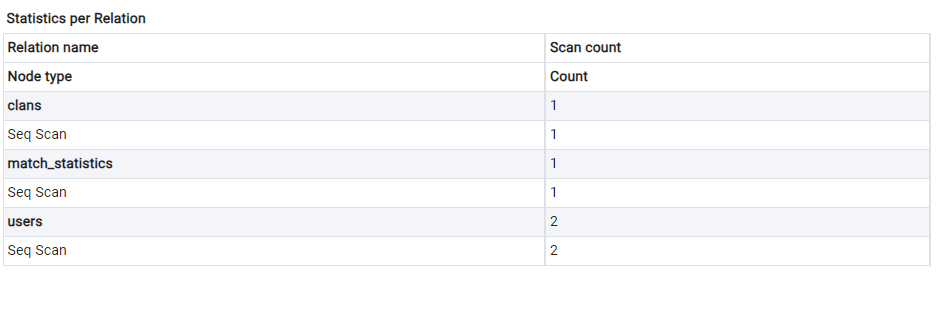


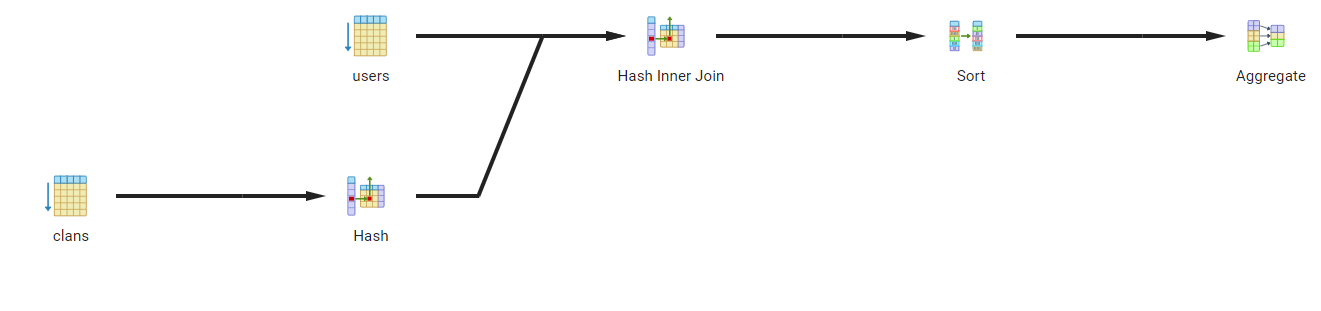
4)

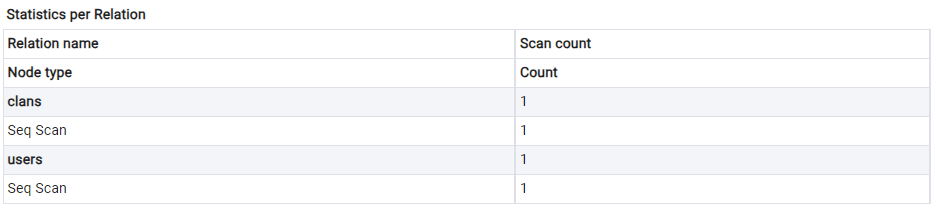
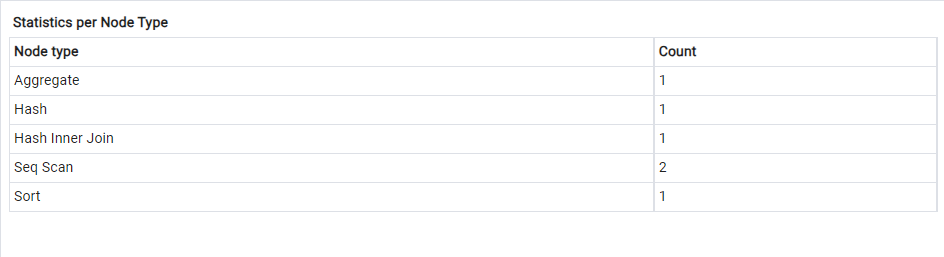






5)



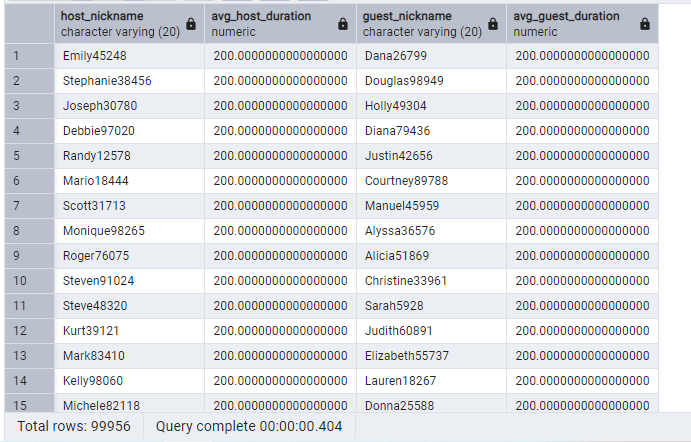


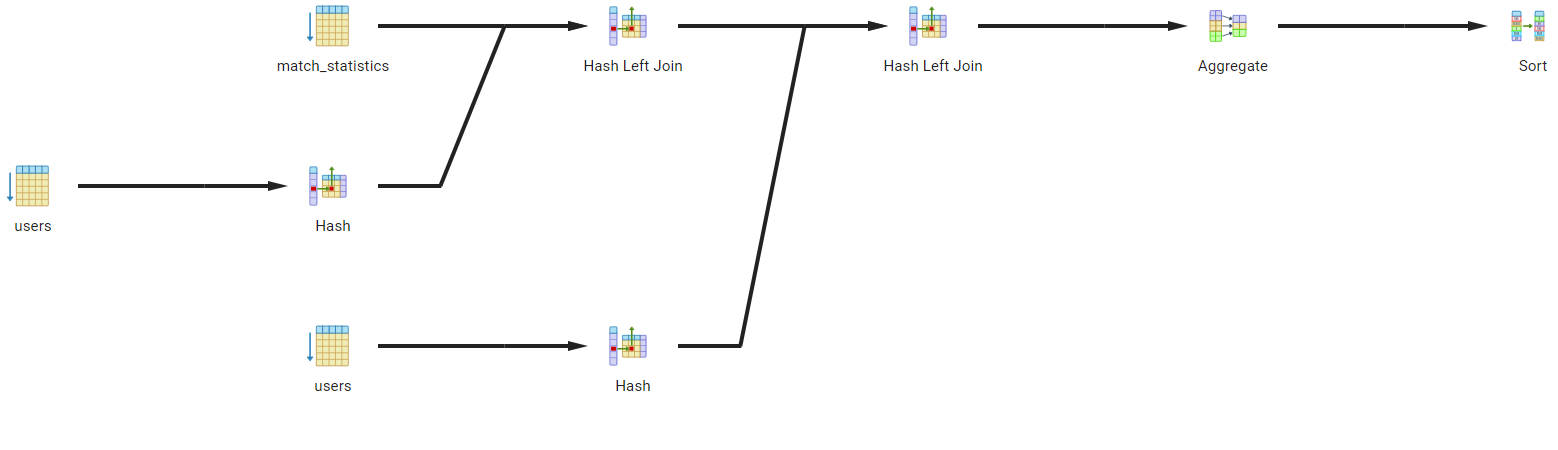
4. Добавление индексов

1)

CREATE INDEX idx\_users\_nickname ON users (nickname)

CREATE INDEX idx\_users\_id ON users (id)





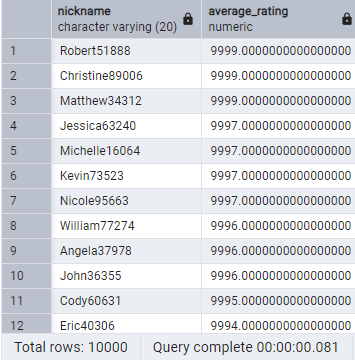
Выбран, т.к. часто используется в условиях (WHERE, ON)

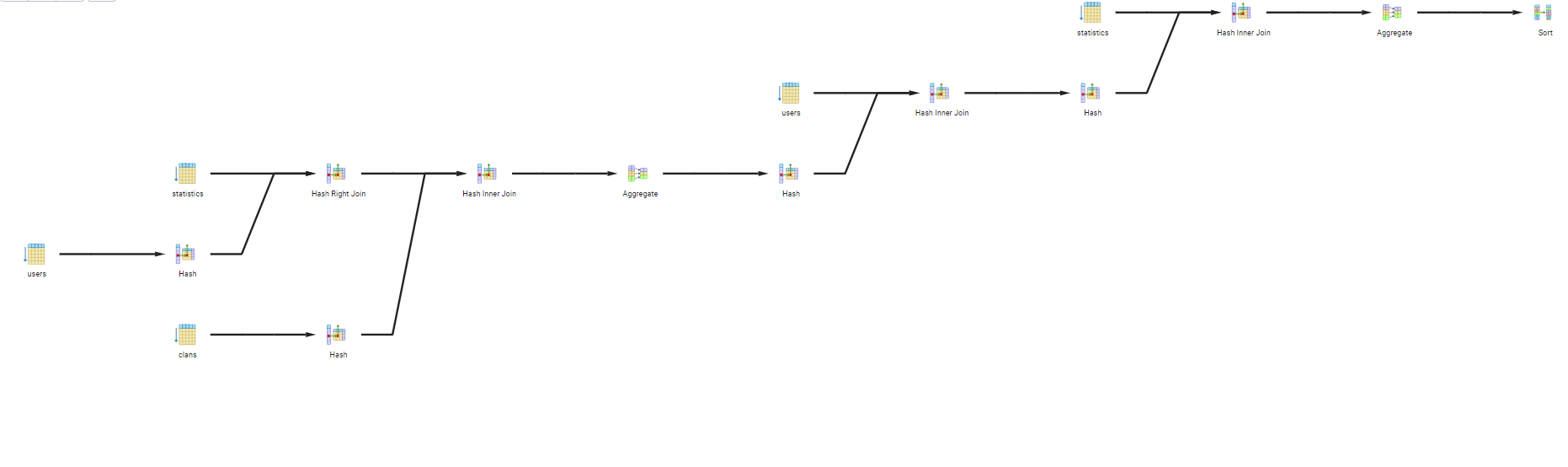
2)

CREATE INDEX idx\_clans\_id ON clans (id)

CREATE INDEX idx\_statistics\_user\_id ON statistics (user\_id)

CREATE INDEX idx\_users\_clan\_id ON users (clan\_id)

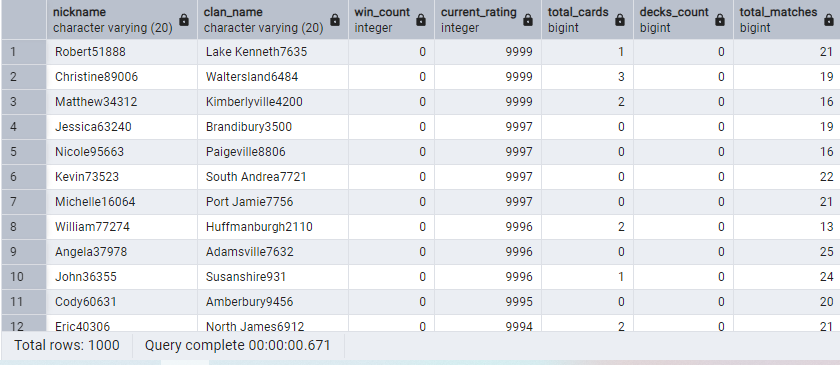


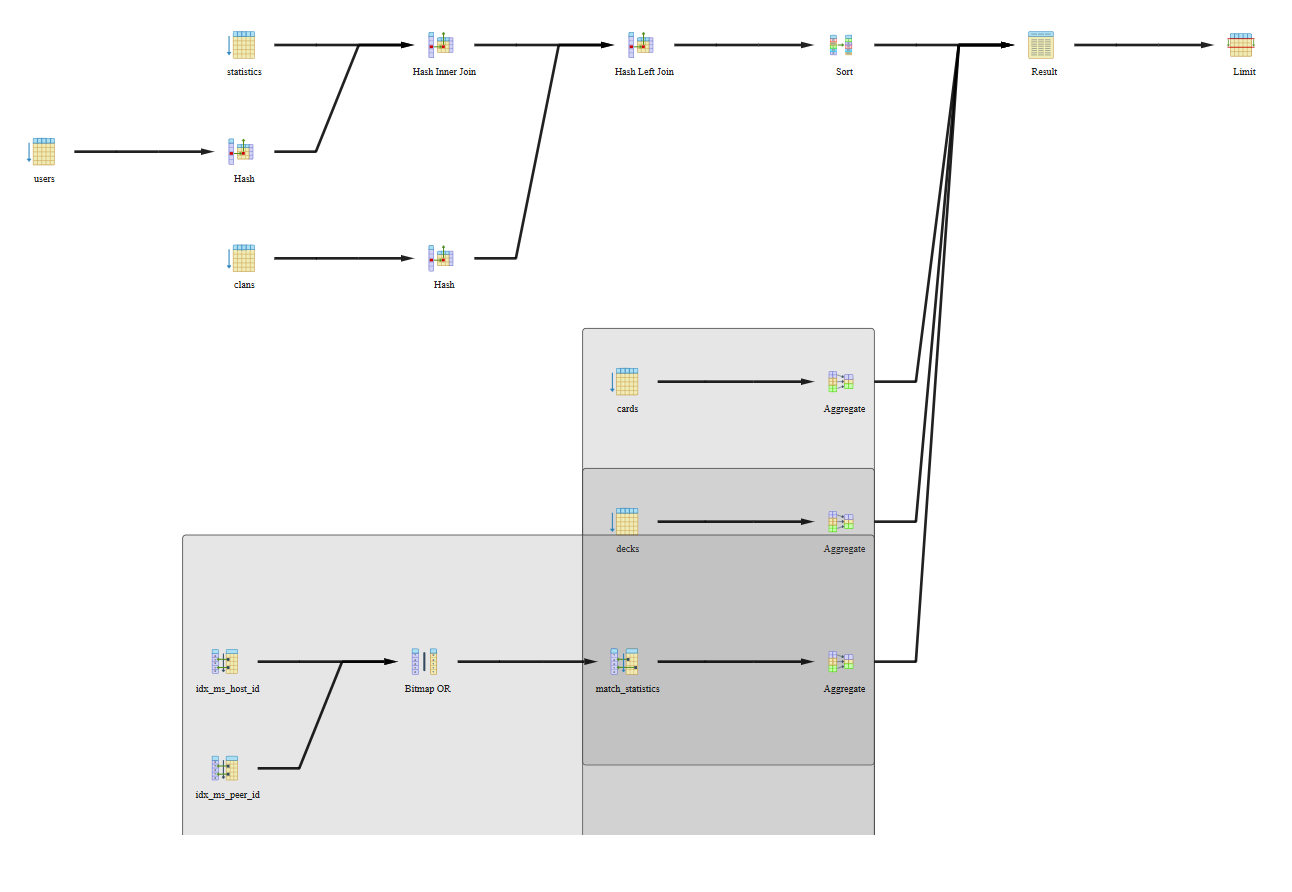


Выбраны, т.к. используются в условиях (WHERE, JOIN ON)

3)

После добавления предыдущих индексов время выполнения сильно уменьшилась:

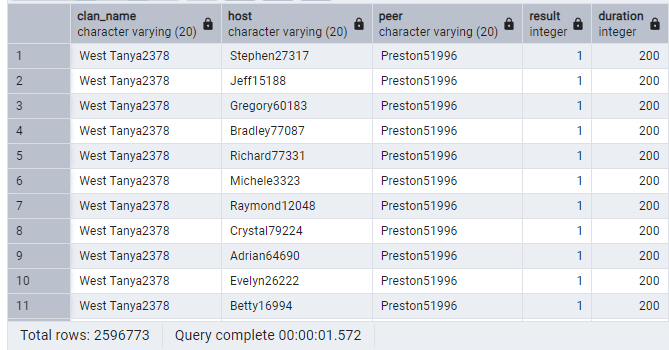


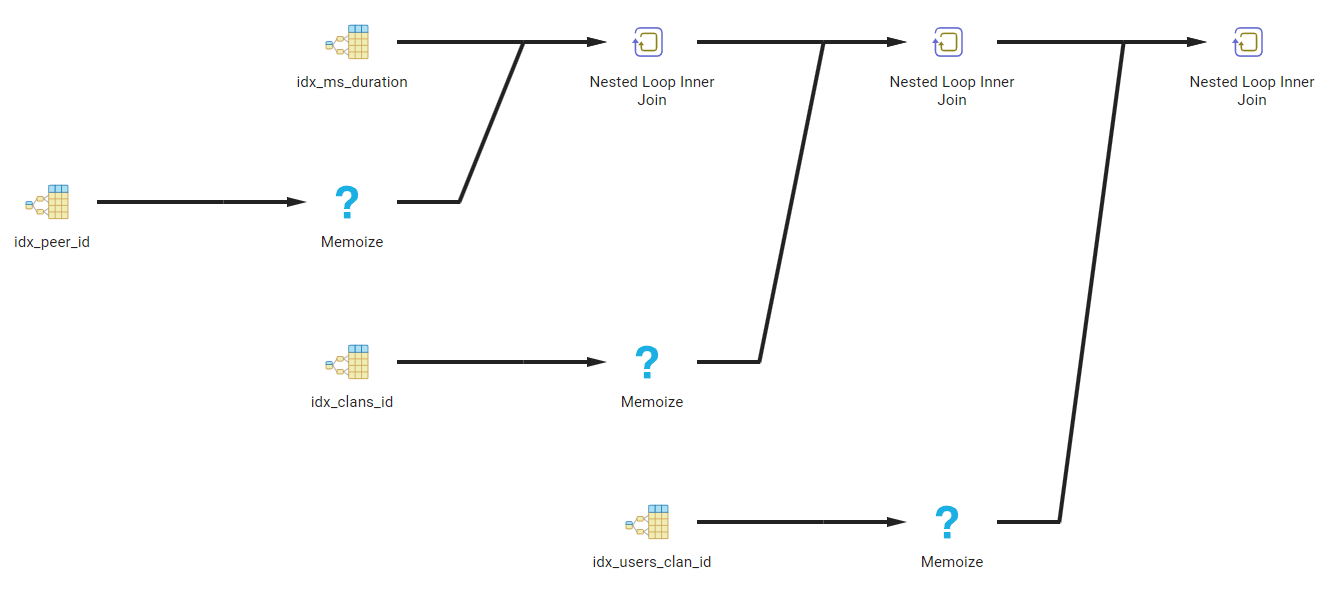


Связано с тем, что много условий (WHERE, JOIN ON)

4)

CREATE INDEX idx\_ms\_duration ON match\_statistics (duration)

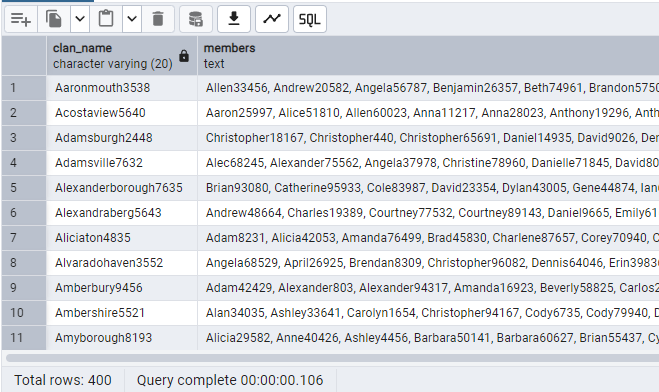


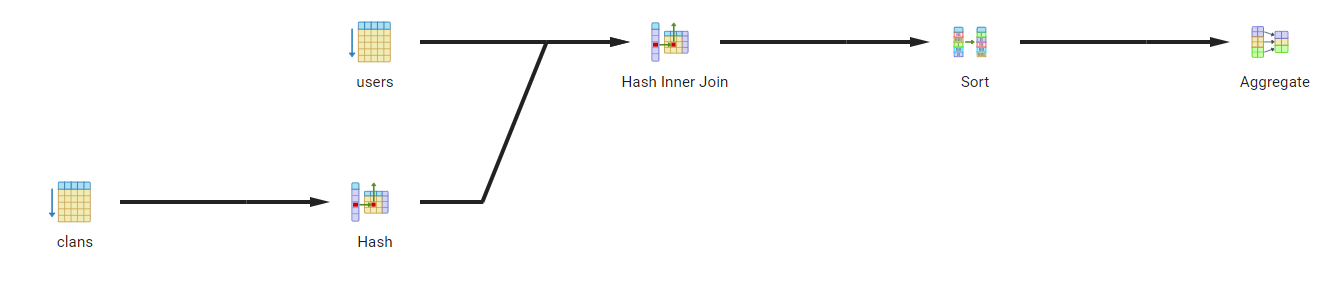


Был выбран, т.к. столбец участвует в сортировке

5)

После добавления прошлых индексов, время уменьшилось почти в двое





Связано с тем, что индексы используются в условиях (WHERE, JOIN) и для сортировки (ORDER BY).

5. Процедуры и функции

Процедуры:

1) Добавить денег пользователю

CREATE PROCEDURE add\_money\_to\_user(

user\_id INT,

add\_money INT

)

LANGUAGE plpgsql

AS $$

DECLARE

new\_money INT;

BEGIN

SELECT money INTO new\_money FROM users WHERE id=user\_id;

new\_money := new\_money + add\_money

IF add\_money < 0 THEN

RAISE EXCEPTION 'Money should be positive integer';

END IF;

UPDATE users

SET money=new\_money

WHERE id=user\_id;

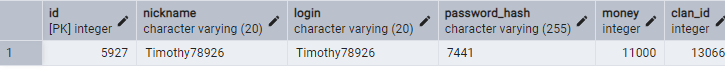
COMMIT;

END;

$$;

Вызов:

CALL add\_money\_to\_user(5927, 10000)



2) Смена ника пользователя

CREATE PROCEDURE change\_nickname(

user\_id INT,

new\_nickname VARCHAR(20)

)

LANGUAGE plpgsql

AS $$

DECLARE

old\_nickname VARCHAR(20);

BEGIN

-- Запоминаем старый ник

SELECT nickname INTO old\_nickname FROM users WHERE id = user\_id;

-- Меняем ник

UPDATE users SET nickname = new\_nickname WHERE id = user\_id;

RAISE NOTICE 'Пользователь #% сменил ник с "%" на "%"',

user\_id, old\_nickname, new\_nickname;

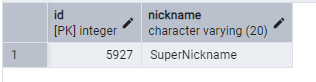
END;

$$;

Вызов процедуры:

CALL change\_nickname(5927, 'SuperNickname');

Результат:



3)

CREATE OR REPLACE PROCEDURE count\_clan\_members(

clan\_name VARCHAR(20)

)

LANGUAGE plpgsql

AS $$

DECLARE

member\_count INT;

BEGIN

-- Считаем участников клана

SELECT COUNT(\*) INTO member\_count

FROM users

WHERE clan\_id = (SELECT id FROM clans WHERE name = clan\_name);

RAISE NOTICE 'В клане "%" % участников', clan\_name, member\_count;

END;

$$;

Вызов процедуры:

CALL count\_clan\_members('Shelbyland204')

Результат:



Функции:

1) Процент побед пользователя

CREATE OR REPLACE FUNCTION get\_winrate(u\_id INT)

RETURNS DECIMAL(5,2)

LANGUAGE plpgsql

AS $$

DECLARE

wins INT;

total INT;

BEGIN

-- Получаем статистику побед и общего числа матчей

SELECT win\_count, matches\_count INTO wins, total

FROM statistics WHERE user\_id = u\_id;

-- Если матчей не было, возвращаем 0

IF total = 0 THEN

RETURN 0;

END IF;

-- Рассчитываем процент побед

RETURN (wins::DECIMAL / total) \* 100;

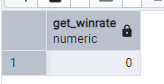
END;

$$;

Вызов функции:

SELECT get\_winrate(5927)

Результат:



2) Может ли игрок создать клан (500 денег и отсутствие клана)

CREATE FUNCTION can\_create\_clan(u\_id INT)

RETURNS BOOLEAN

LANGUAGE plpgsql

AS $$

DECLARE

user\_money INT;

user\_has\_clan BOOLEAN;

BEGIN

SELECT money, clan\_id IS NOT NULL INTO user\_money, user\_has\_clan

FROM users WHERE id = u\_id;

RETURN user\_money >= 500 AND NOT user\_has\_clan;

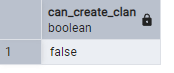
END;

$$;

Вызов функции:

SELECT can\_create\_clan(5927)

Результат:



3) Проверка, владелец ли клана игрок

CREATE OR REPLACE FUNCTION is\_clan\_owner(user\_id INT)

RETURNS BOOLEAN

LANGUAGE plpgsql

AS $$

DECLARE

owner\_status BOOLEAN;

BEGIN

SELECT EXISTS(

SELECT 1 FROM clans WHERE owner\_id = user\_id

) INTO owner\_status;

RETURN owner\_status;

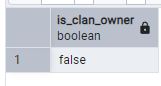
END;

$$;

Вызов функции:

SELECT is\_clan\_owner(5927)

Результат:



6. Представления

1) Данные пользователя

CREATE OR REPLACE VIEW user\_profiles AS

SELECT

u.id,

u.nickname,

u.money,

c.name AS clan\_name,

s.win\_count,

s.matches\_count,

s.current\_rating,

s.max\_rating,

ROUND((s.win\_count::DECIMAL / NULLIF(s.matches\_count, 0)) \* 100, 2) AS win\_rate

FROM

users u

LEFT JOIN

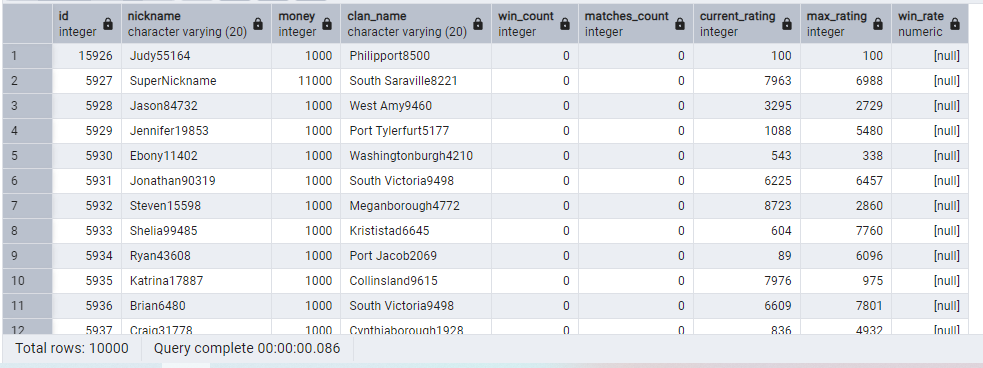
clans c ON u.clan\_id = c.id

JOIN

statistics s ON u.id = s.user\_id;

Проверка:

SELECT \* FROM user\_profiles



2) Кланы и их владельцы

CREATE OR REPLACE VIEW clan\_details AS

SELECT

c.id,

c.name AS clan\_name,

c.description,

u.nickname AS owner\_nickname,

u.id AS owner\_id,

COUNT(m.id) AS member\_count

FROM

clans c

JOIN

users u ON c.owner\_id = u.id

LEFT JOIN

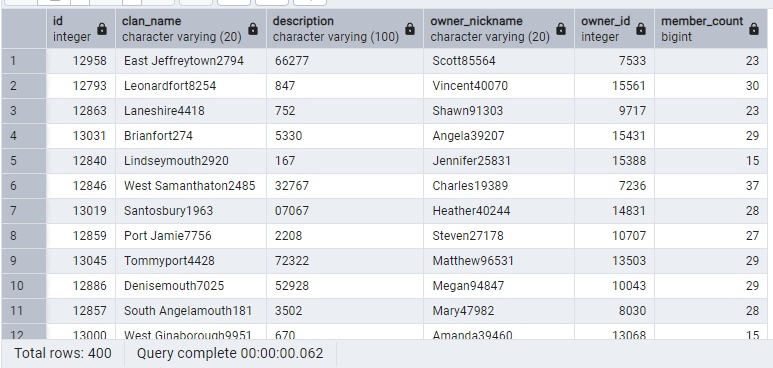
users m ON c.id = m.clan\_id

GROUP BY

c.id, c.name, c.description, u.nickname, u.id;

Проверка:

SELECT \* FROM clan\_details



3) Топ игроков по рейтингу

CREATE OR REPLACE VIEW top\_players AS

SELECT

u.nickname,

s.current\_rating,

s.max\_rating,

s.win\_count,

s.matches\_count,

c.name AS clan\_name,

RANK() OVER (ORDER BY s.current\_rating DESC) AS rank

FROM

users u

JOIN

statistics s ON u.id = s.user\_id

LEFT JOIN

clans c ON u.clan\_id = c.id

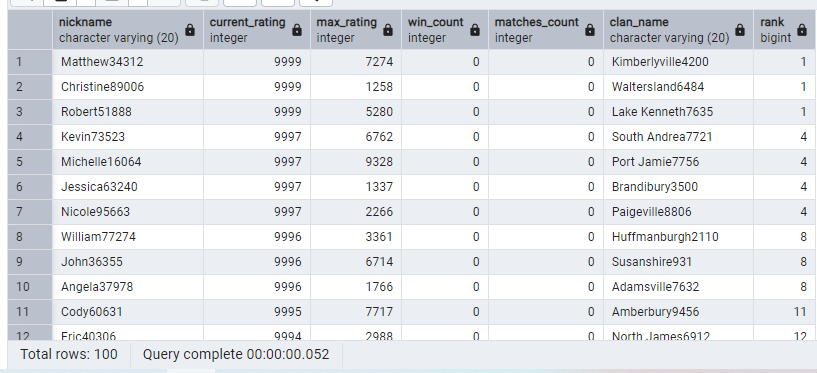
ORDER BY

s.current\_rating DESC

LIMIT 100;

Проверка:

SELECT \* FROM top\_players



7. Триггер

Триггер, создающий таблицу статистики, при создании пользователя

-- 1. Сначала создаем триггерную функцию

CREATE OR REPLACE FUNCTION create\_user\_statistics()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

-- Вставляем новую запись в statistics для нового пользователя

INSERT INTO statistics (user\_id, win\_count, matches\_count, max\_rating, current\_rating)

VALUES (NEW.id, 0, 0, 100, 100);

RETURN NEW;

END;

$$;

-- 2. Затем создаем триггер, который вызывает эту функцию

CREATE TRIGGER trg\_create\_statistics

AFTER INSERT ON users

FOR EACH ROW

EXECUTE FUNCTION create\_user\_statistics();

Проверка:

-- 1. Создаем тестового пользователя

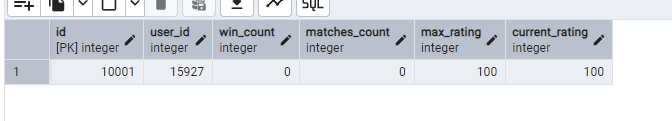
INSERT INTO users (nickname, login, password\_hash)

VALUES ('test\_user', 'test\_login', 'hashed\_password');

-- 2. Проверяем, создалась ли статистика

SELECT \* FROM statistics WHERE user\_id = (SELECT id FROM users WHERE login = 'test\_login');

Результат:



**Вывод:** после выполнения лаб. работы я научился работать с индексами, хранимыми процедурами, представлениями и триггерами.

**Лабораторная работа №6**

**Цель работы:** создать программу с графическим интерфейсом для своего проекта с использованием базы данных

**Задачи работы:** Создать приложение, взаимодействующее с базой данных; Создать графический интерфейс для этого приложения.

**Ход выполнения работы:**

1. Был создан бэкенд на языке программирования Python с использованием фреймворка FastAPI.

Для взаимодействия с базой данных была выбрана библиотека asyncpg, предоставляющая возможность совершать асинхронные запросы к базе данных.

Также были использованы библиотеки Pydantic, bcrypt и линтер Black (к сожалению, не везде)

Ссылка на репозиторий (GitHub): <https://github.com/TWTwoFace/card-game-backend>

2. Был создан Docker контейнер для бэкенд-приложения, а также собран Docker compose для воссоздания posgresql базы данных и изолирования приложений друг от друга.

Были использованы переменные окружения, для избежания попадания чувствительных данных в репозиторий.

Приложение хостится на арендованном виртуальном сервере.

3. Был создан клиент на Unity 6 с использованием:

– Extenject (Zenject) – DI контейнер

– DOTween – библиотека для воспроизведения анимаций из кода

– Newtonsoft.Json – для сериализации объектов

– System.Net.Http – для Http запросов к бэкенду

– UnityWebRequests – пришлось использовать, т.к. Unity до сих пор не поддерживает Patch запросы (патч реализован костылём)

Шаблоны проектирования:

В основном клиентский код писался на голых MonoBehaviour классах, но с использованием DI контейнера. Повсюду используется Observer (наблюдатель). Было бы хорошо использовать MVP или MVVM, но с такими темпами, эта работа была бы написана к концу августа.

Нижний уровень взаимодействия с бэкендом реализован с помощью шаблона проектирования Репозиторий.

Также используются объекты провайдеры, для получения конфигурационных данных вложенными сущностями.

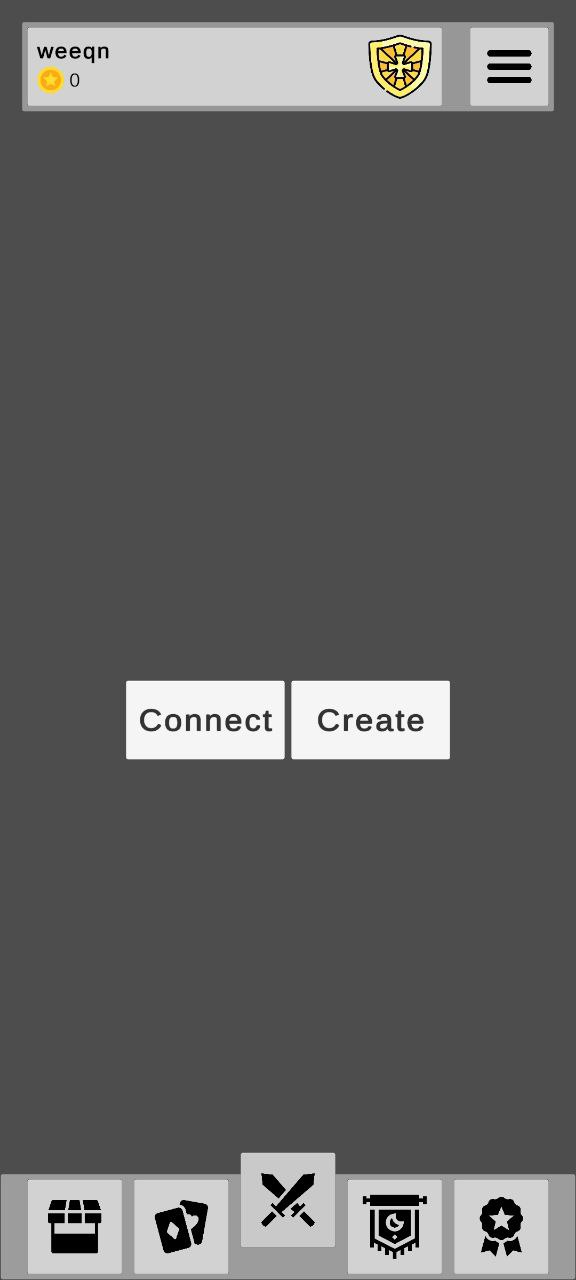
UI сделан адаптивным, подcтраивающимся под различные соотношения сторон экрана. Портретная ориентация. Адаптивность под “островки” и блок камеры отсутствует.

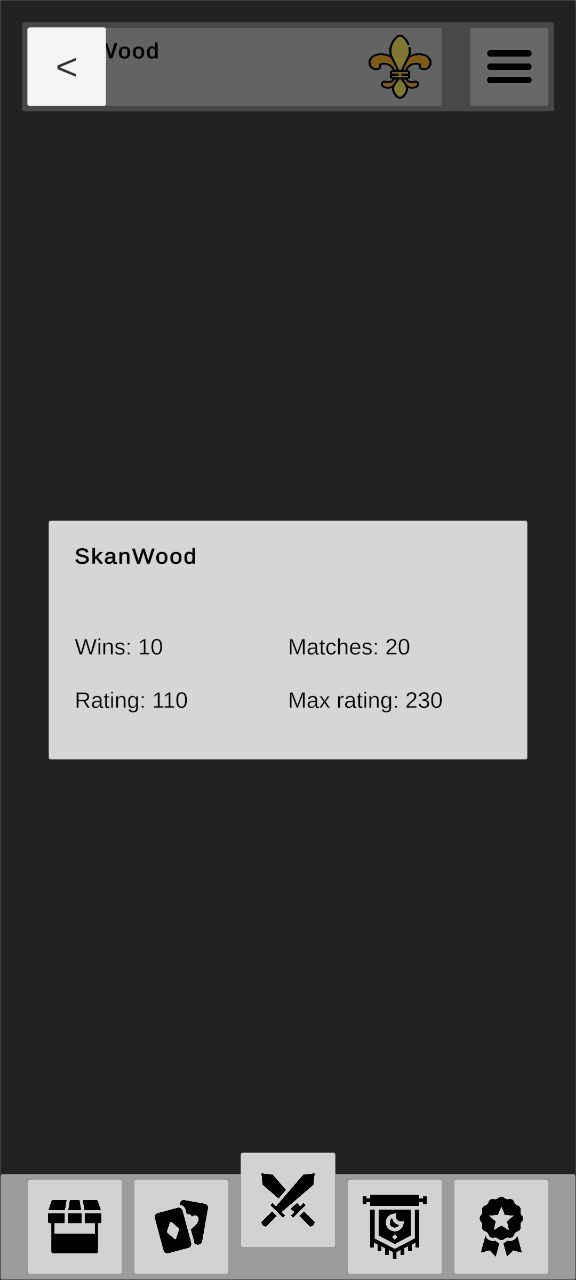
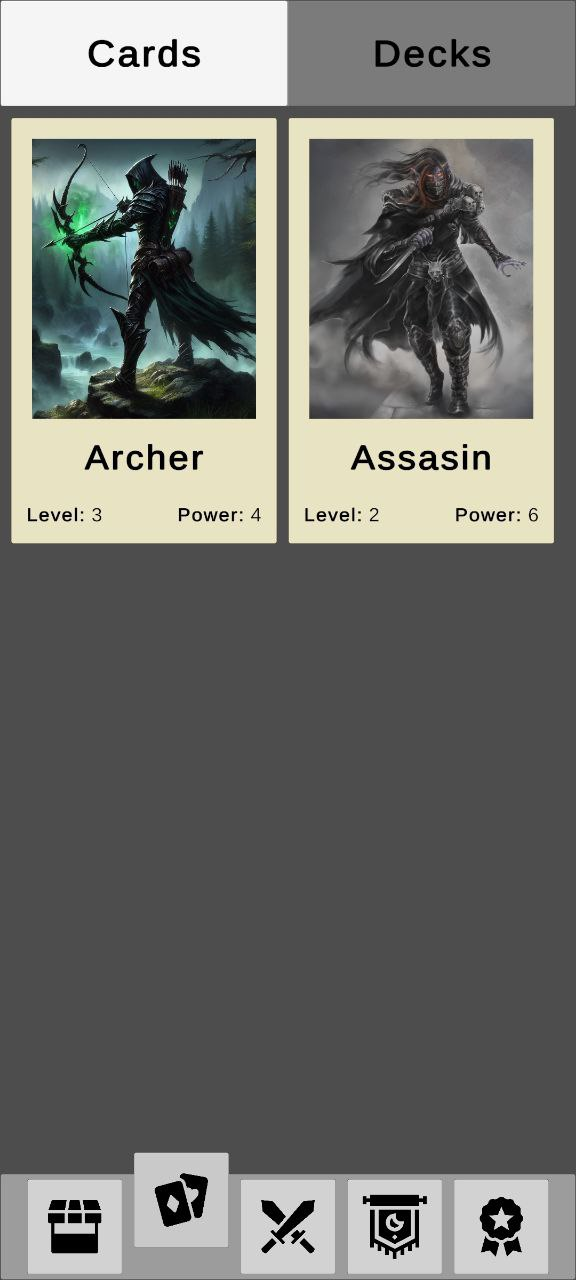
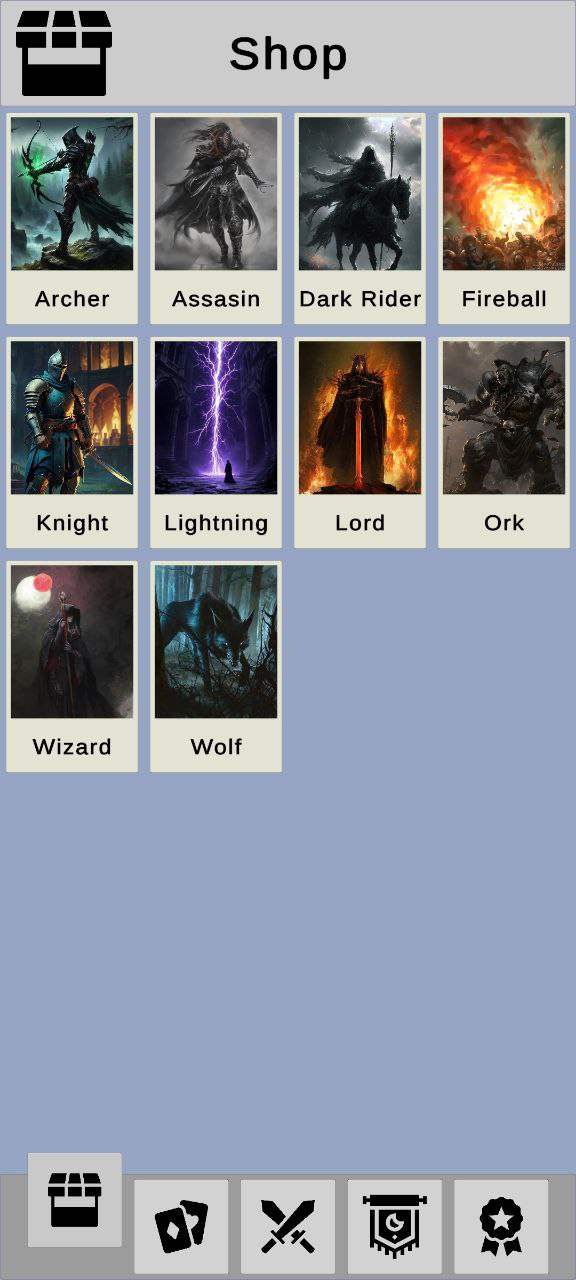
Билд собирался под Android. Собрать под IOS не было возможности.

В клиенте реализована большая часть возможностей бэкенда, но не вся. Из основного: не реализованными остались матчи и их статистика, т.к. тащить в зависимость Net-фреймворк – слишком сложно. Условная реализация не предусмотрена.

CRUD реализован для пользователей и кланов.

Скриншоты билда проекта:





Билд под Android прилагается в zip файле с семестровой работой.

Ссылка на репозиторий проекта клиента: <https://github.com/TWTwoFace/card-game-client>

**Вывод:** за период выполнения семестровой работы я научился многому: принципы реляционных баз данных; работе с СУБД; написанию запросов к базам данных; оптимизации запросов; многим инструментам баз данных; работе с базами данных из других ЯП; репликации данных из БД в клиентских приложениях.