|  |  |  |
| --- | --- | --- |
| Лабораторная работа №4  Реализация SQL-запросов базы данных | Ф.И.О. | Попов А.В. |
| Группа | ПрИн 166 |
| Преподаватель |  |
| Дата сдачи |  |

**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

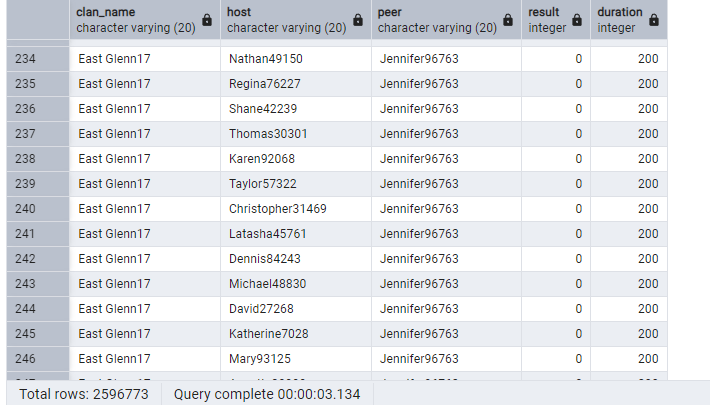
FROM match\_statistics m

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

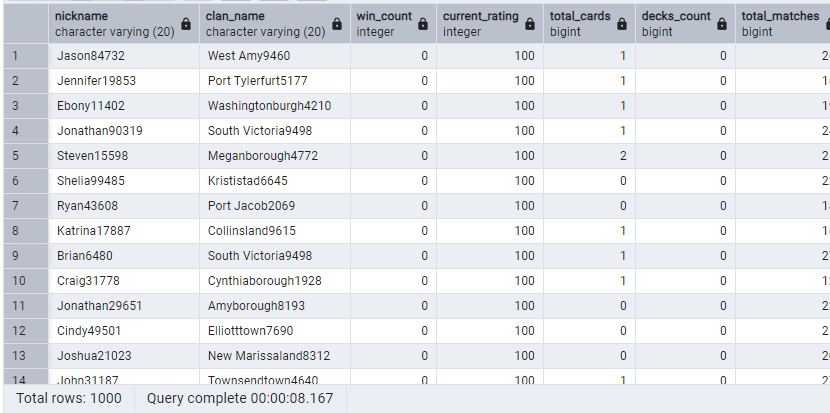
FROM users u

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

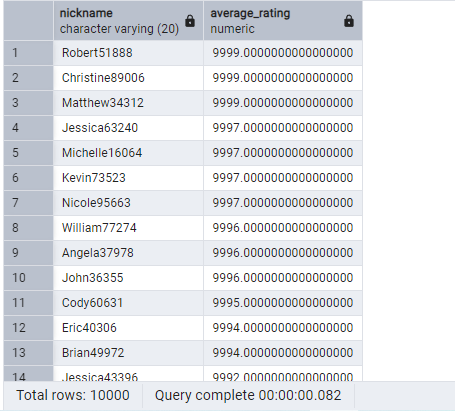
FROM users u

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;



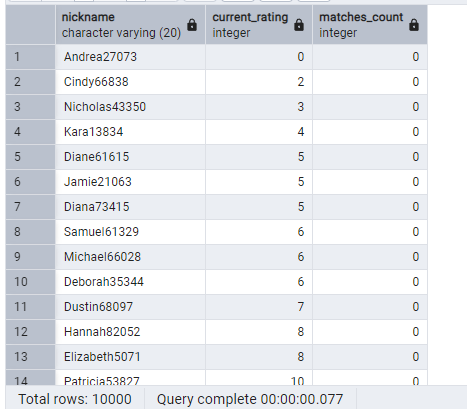
100. Получение топа игроков по наименьшему количеству матчей

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

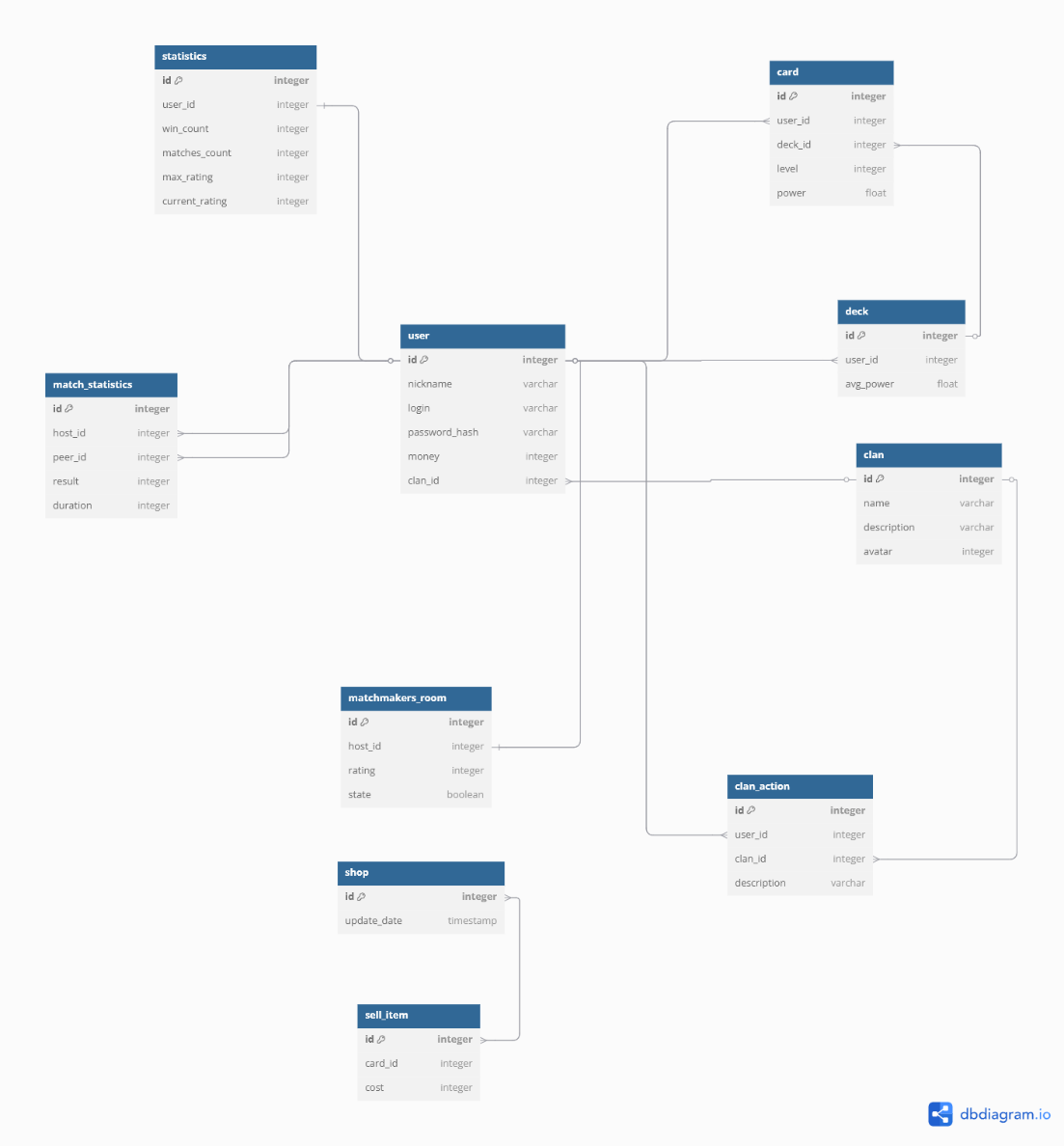
FROM users u

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.