|  |  |  |
| --- | --- | --- |
| Лабораторная работа №5  Индексы, хранимые процедуры, представления, триггеры | Ф.И.О. | Попов А.В. |
| Группа | ПрИн 166 |
| Преподаватель |  |
| Дата сдачи |  |

**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

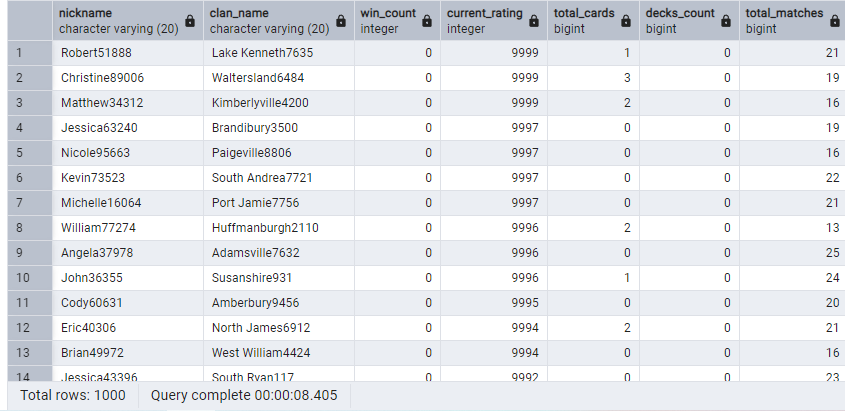
FROM users u

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

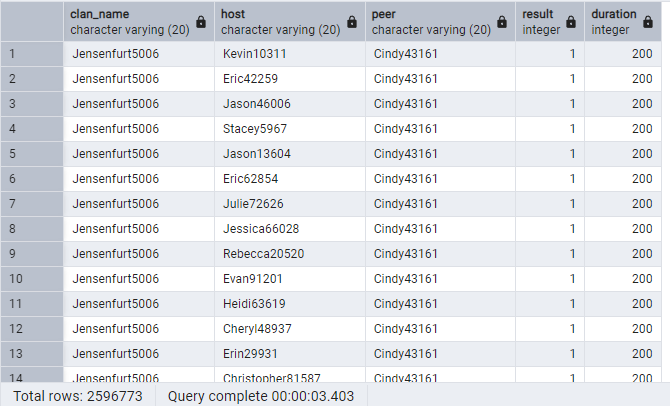
FROM match\_statistics m

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,

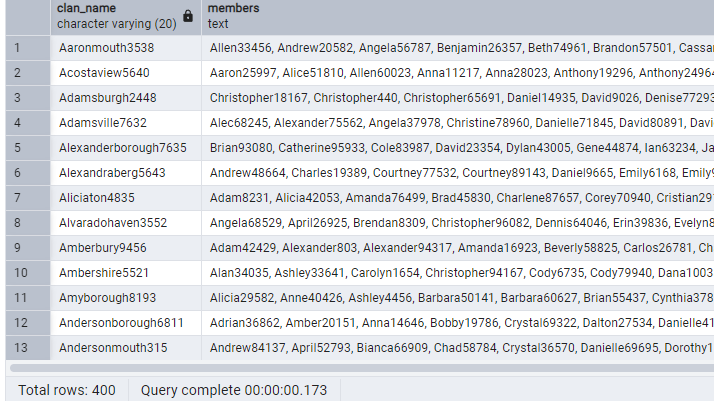
STRING\_AGG(u.nickname, ', ' ORDER BY u.nickname) AS members

FROM clans c

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

GROUP BY c.name

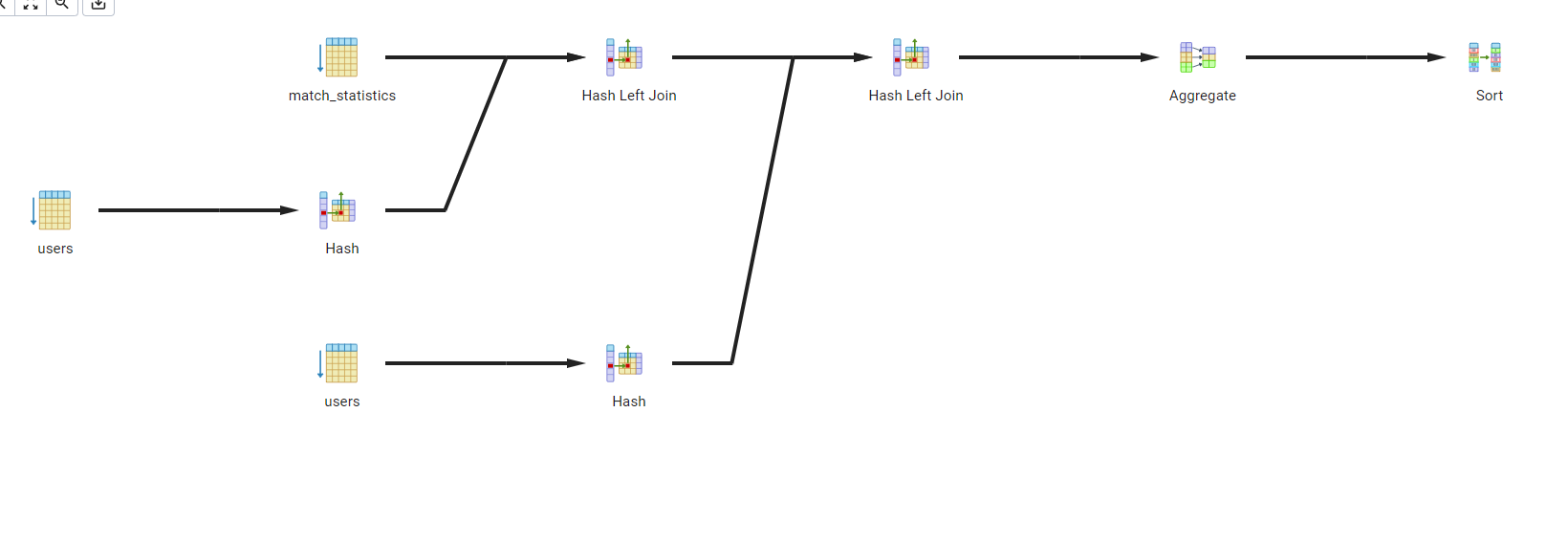
HAVING COUNT(u.id) > 0;

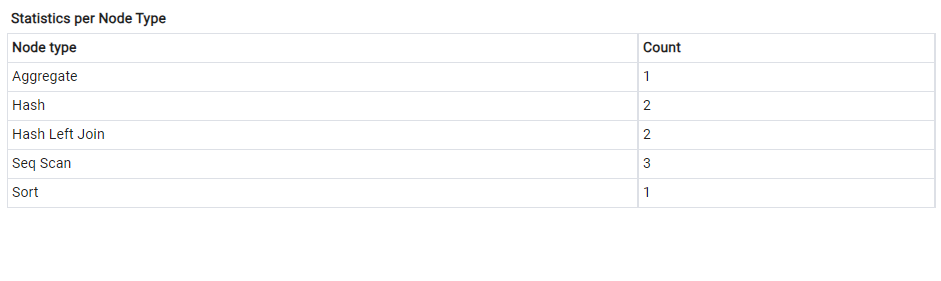


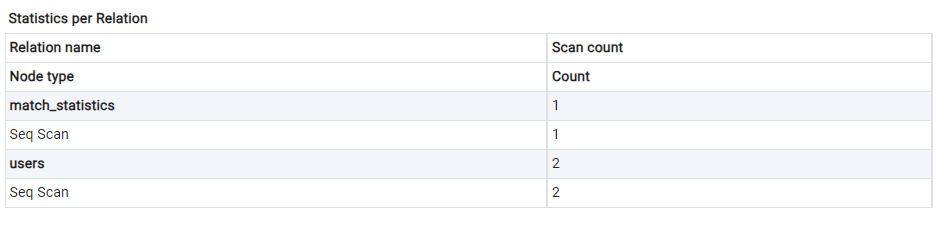
Время выполнения 173 мс.

3. Анализ скорости выполнения

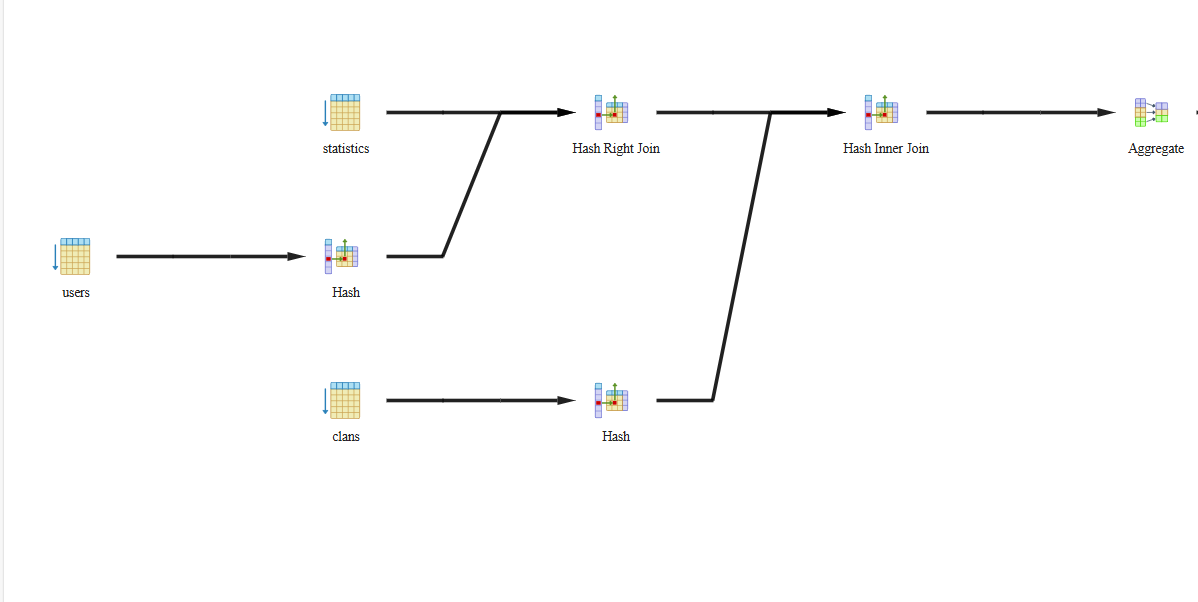
1)

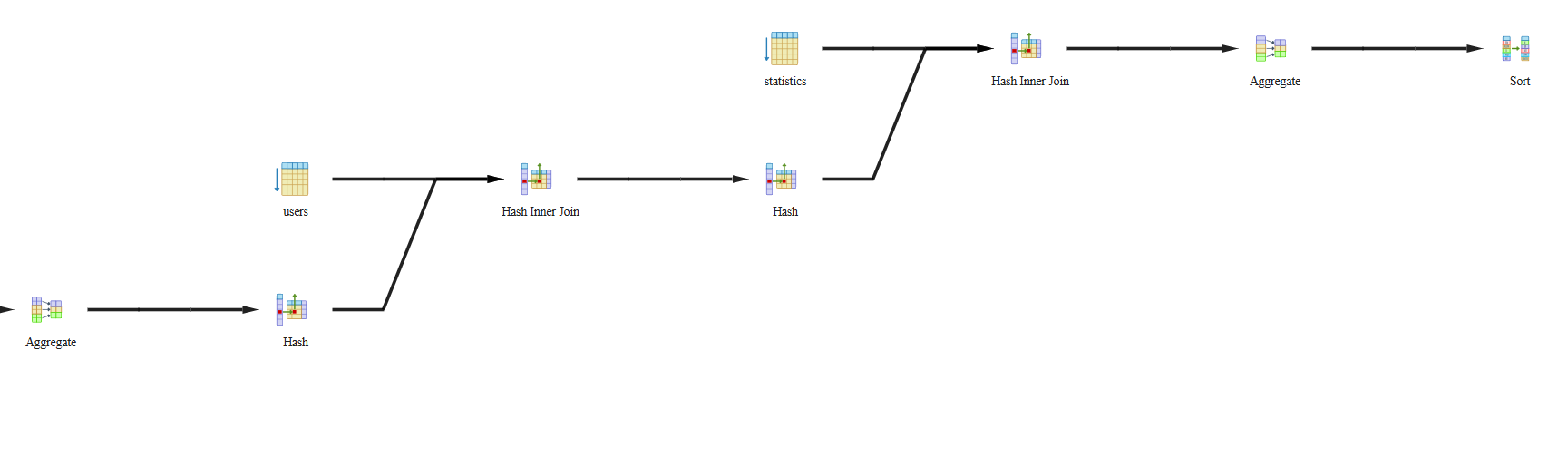




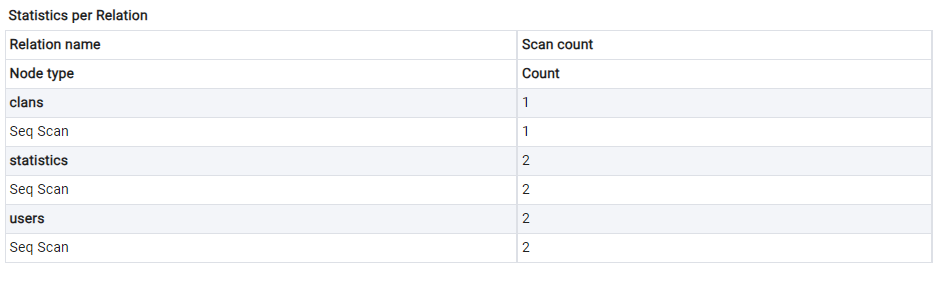


2)

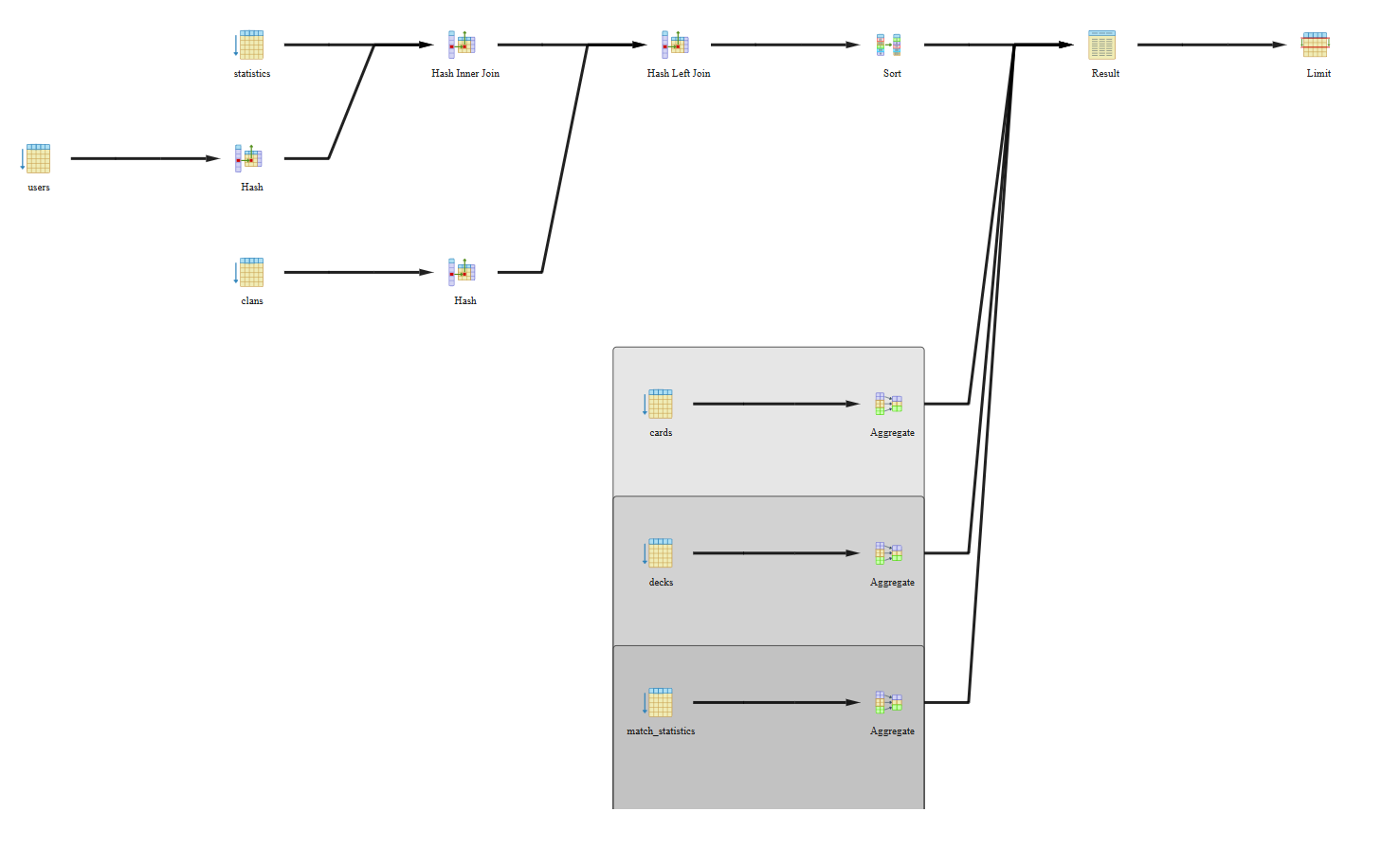


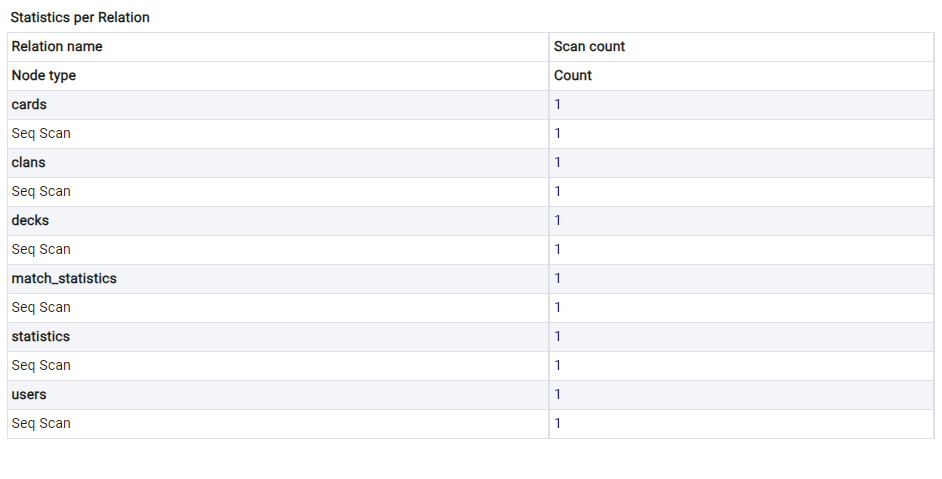




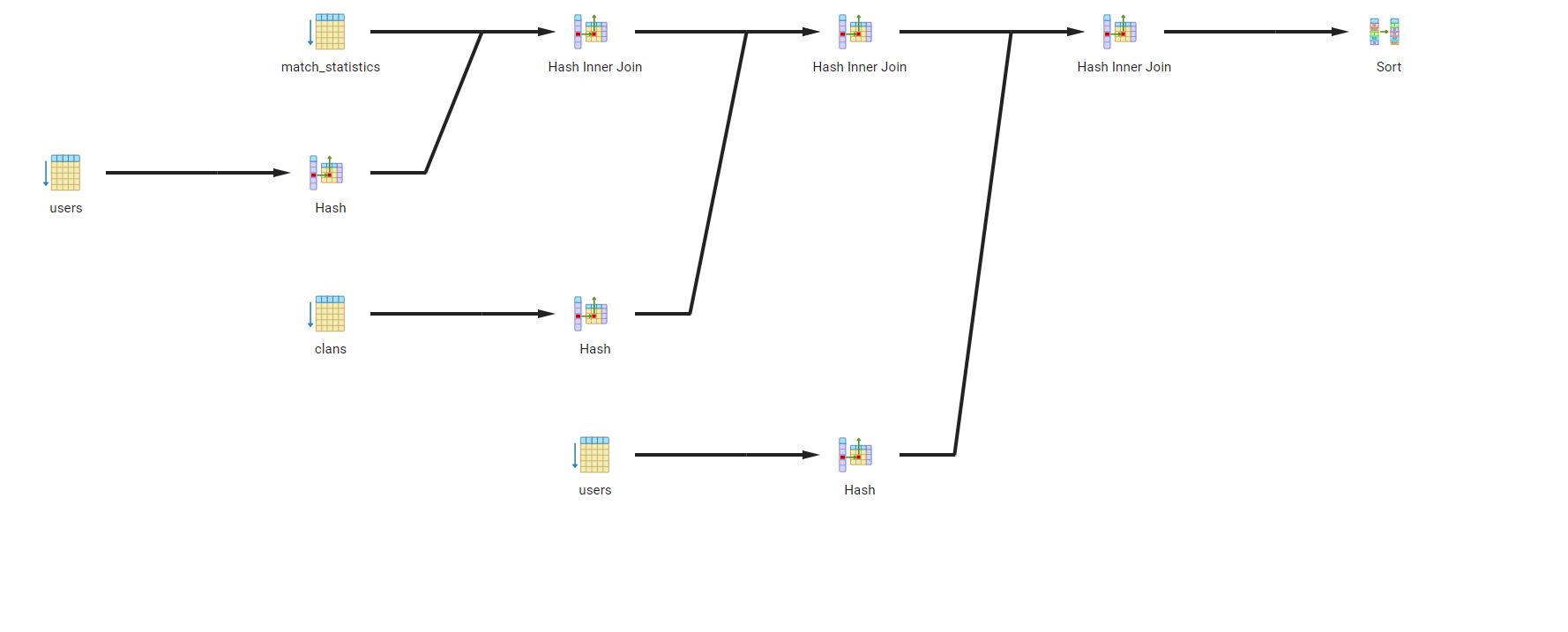


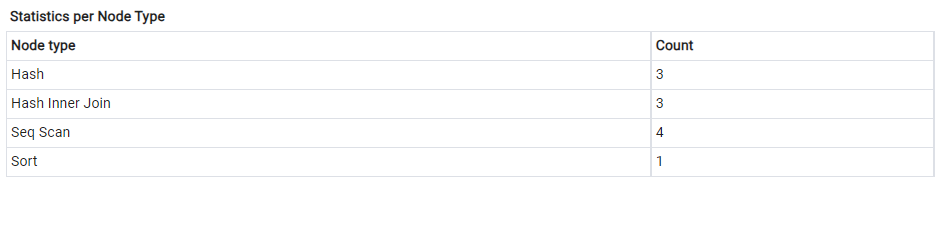
3)

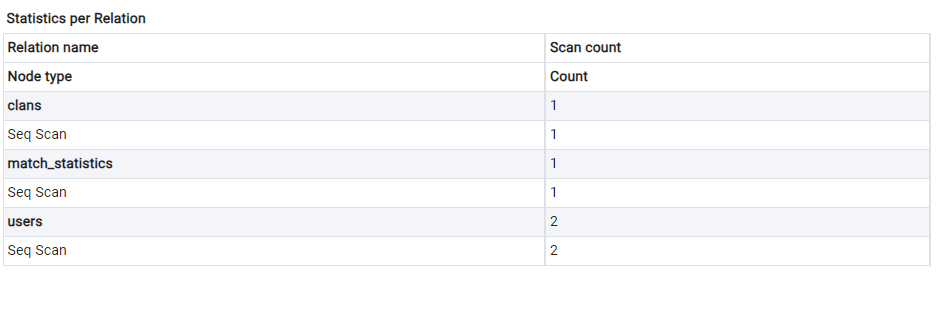


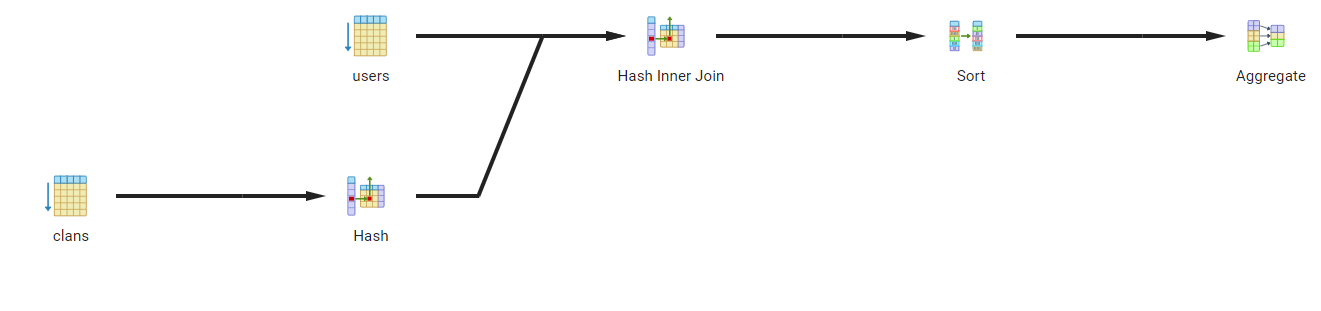
4)

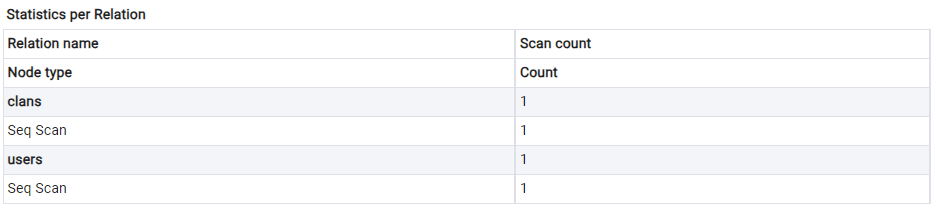
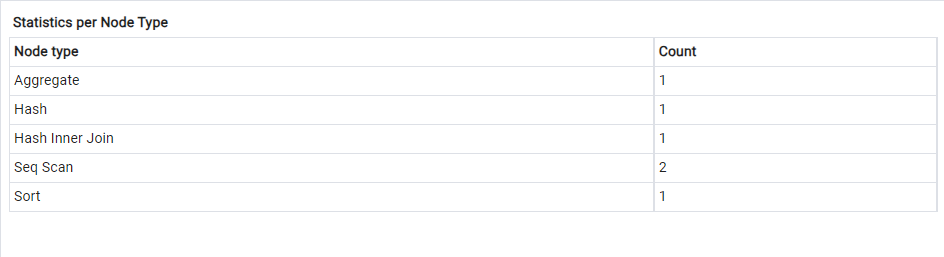






5)



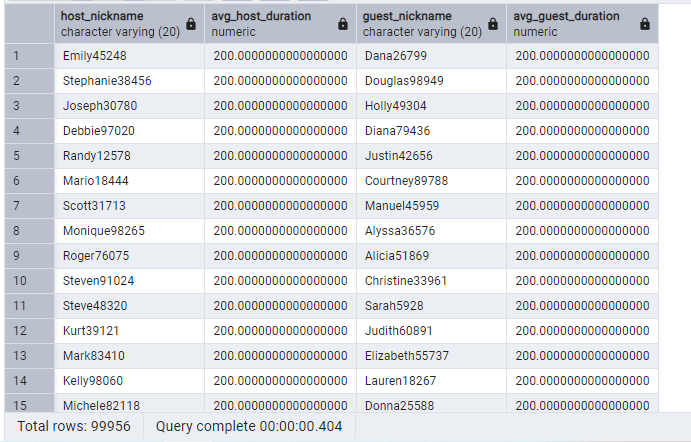


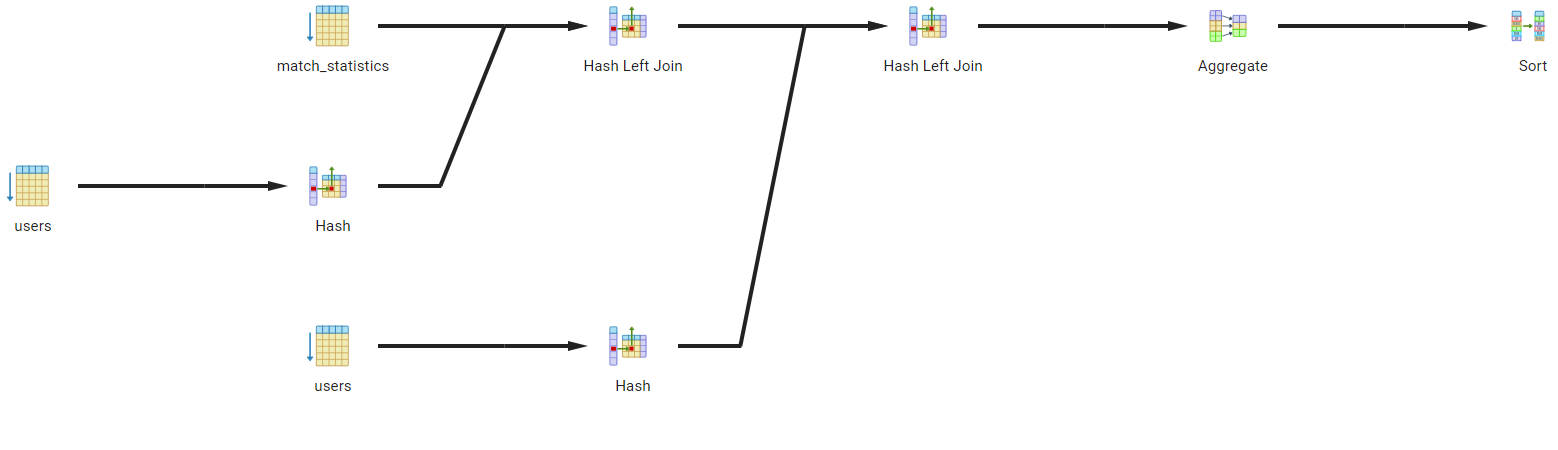
4. Добавление индексов

1)

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

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





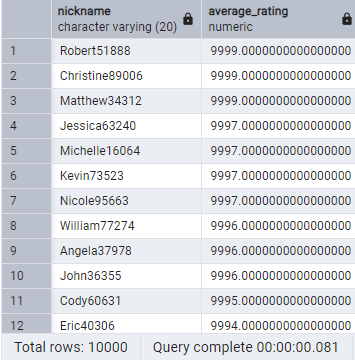
Выбран, т.к. часто используется в условиях (WHERE, ON)

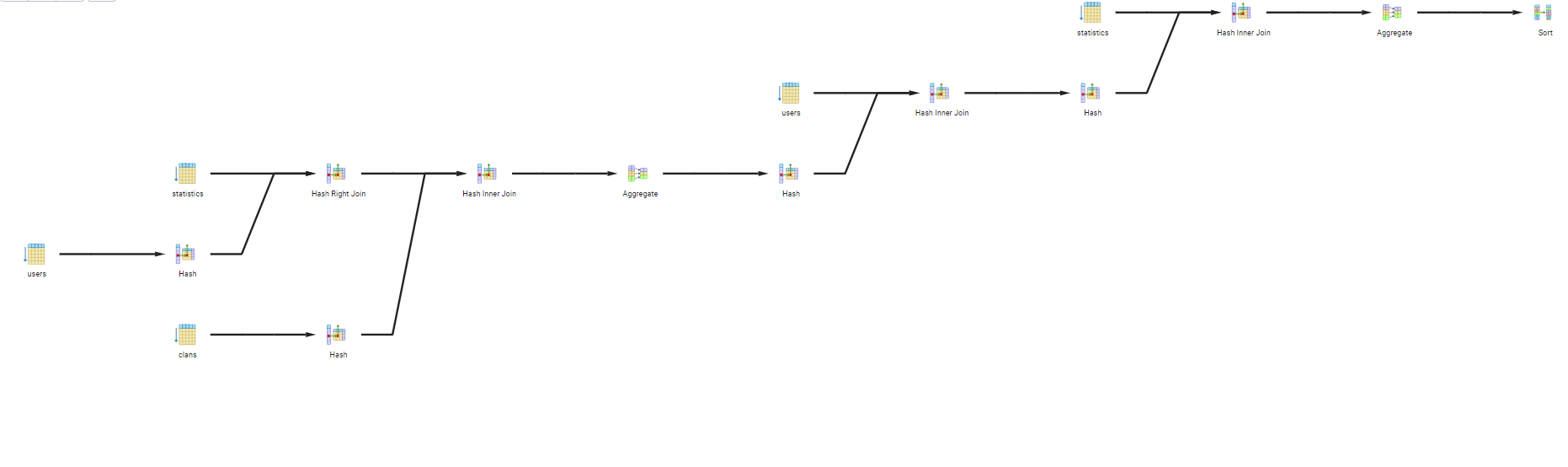
2)

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)

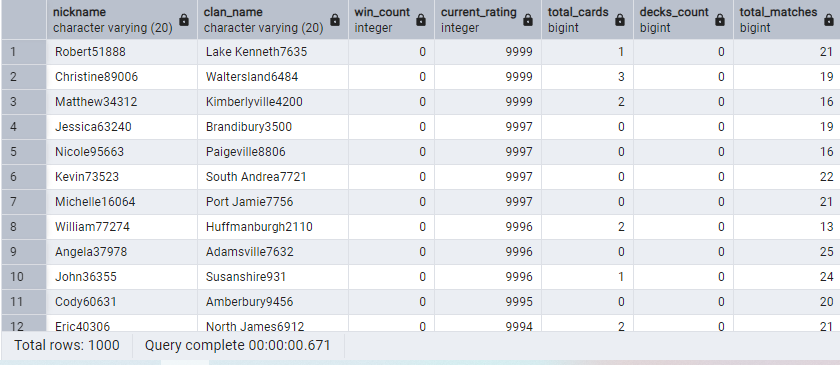


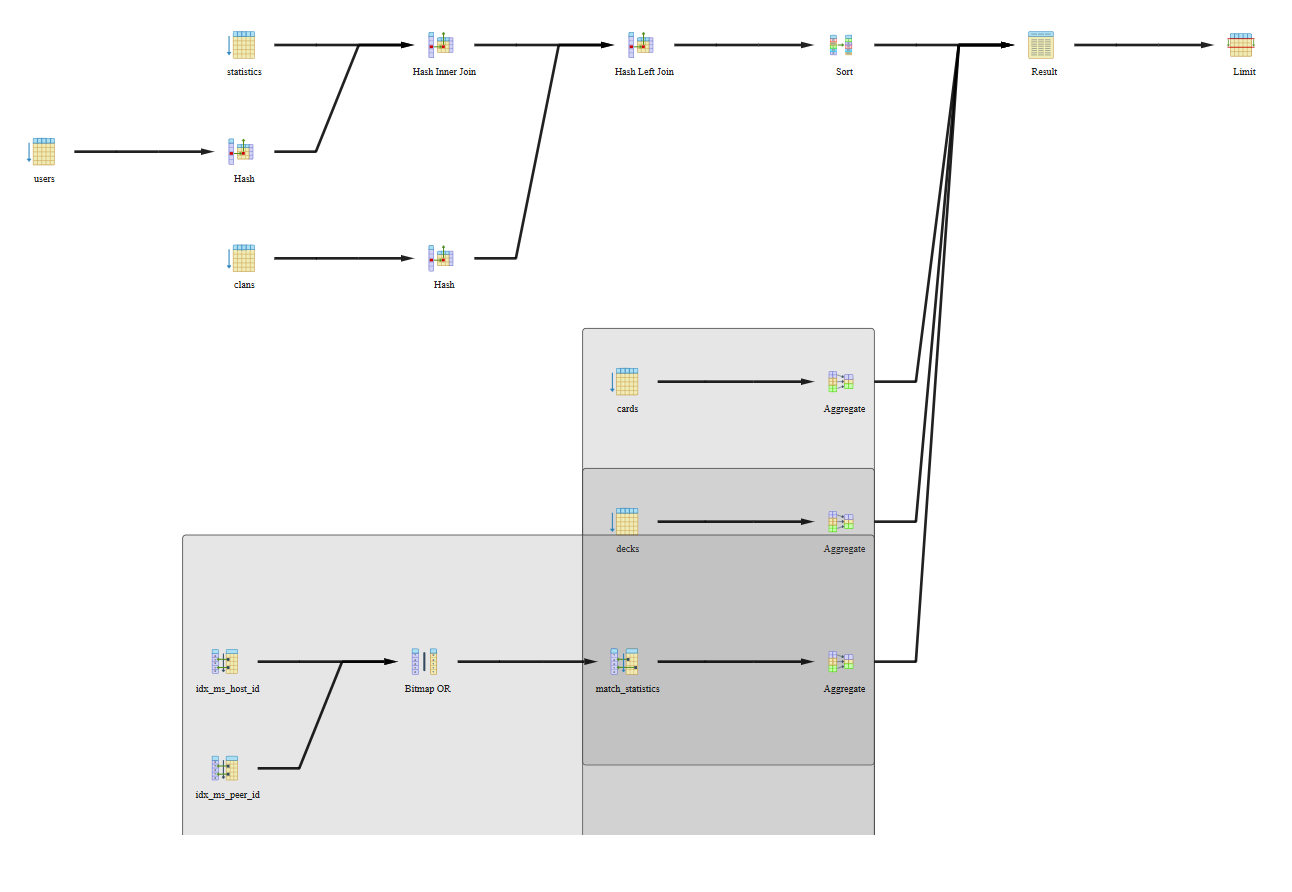


Выбраны, т.к. используются в условиях (WHERE, JOIN ON)

3)

После добавления предыдущих индексов время выполнения сильно уменьшилась:

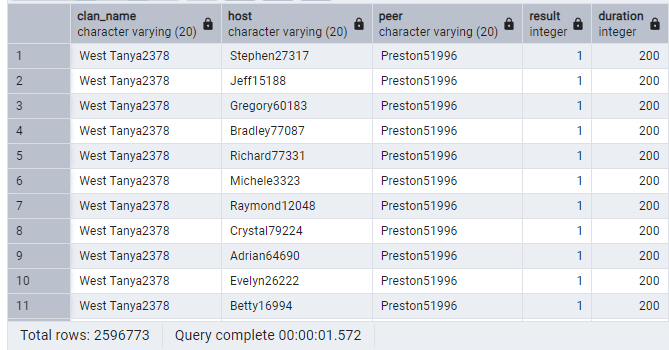


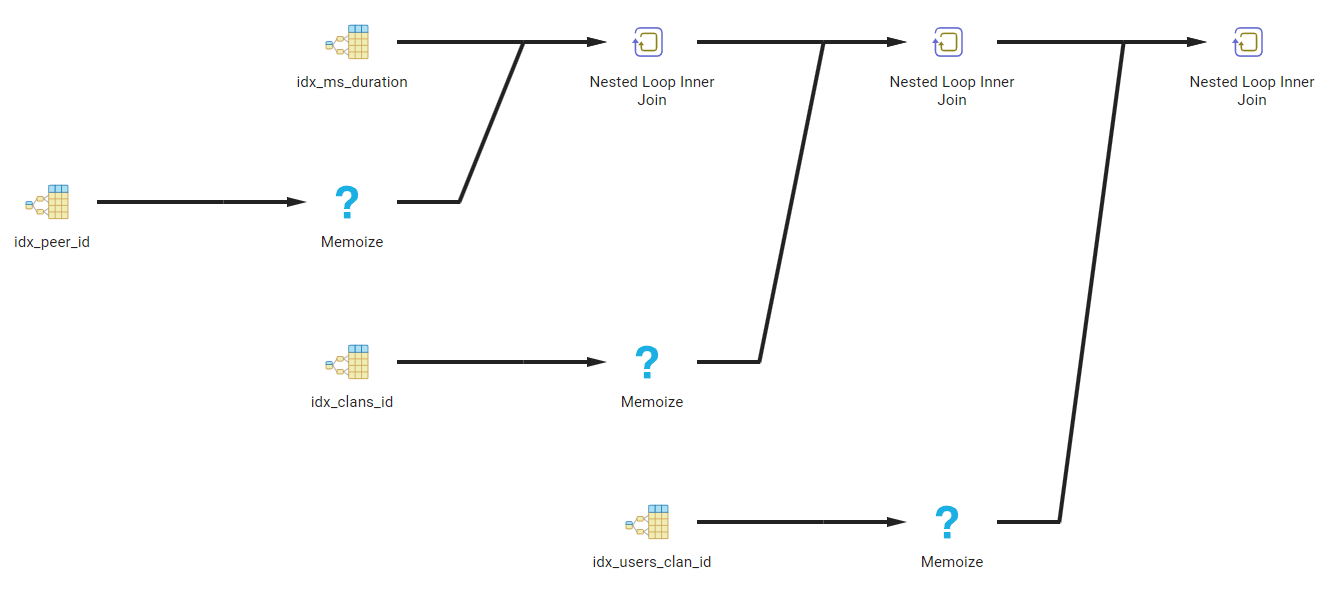


Связано с тем, что много условий (WHERE, JOIN ON)

4)

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

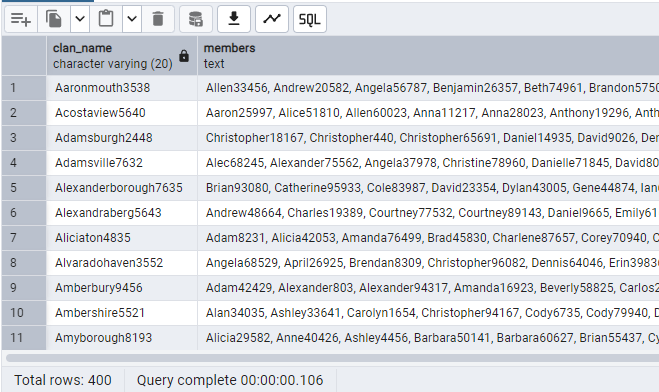


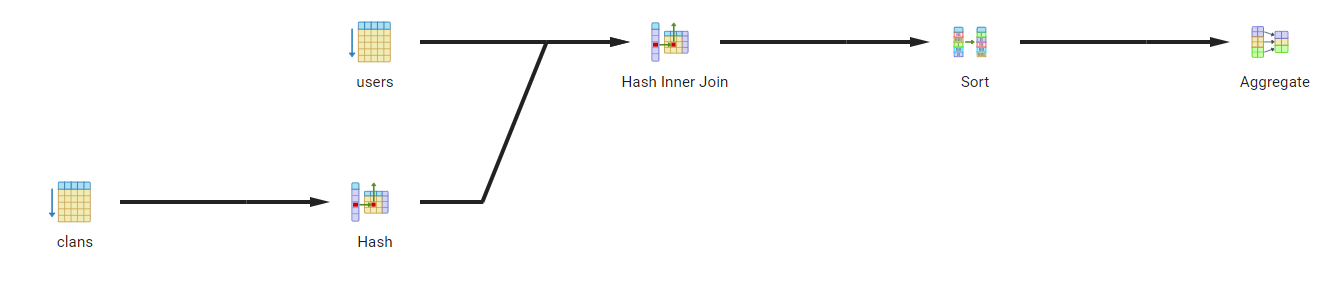


Был выбран, т.к. столбец участвует в сортировке

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;

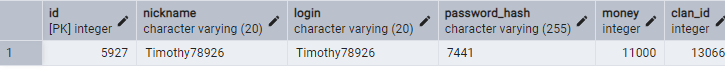
COMMIT;

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;

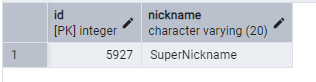
END;

$$;

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;

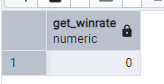
END;

$$;

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;

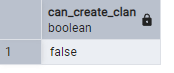
END;

$$;

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;

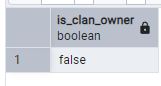
END;

$$;

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

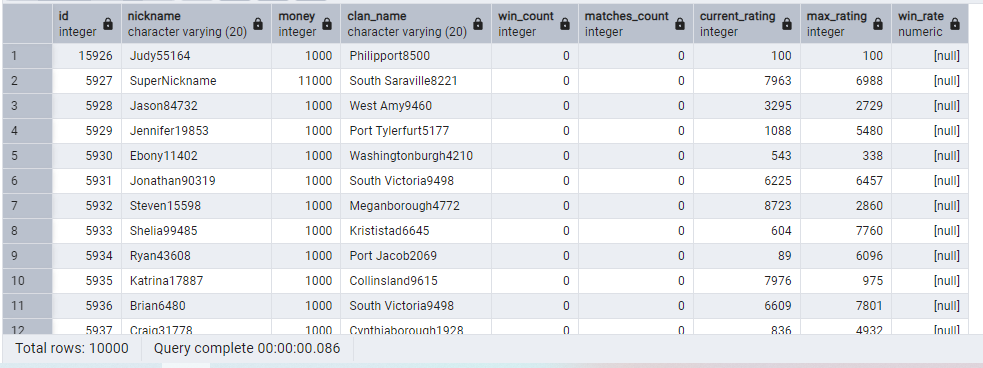
clans c ON u.clan\_id = c.id

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

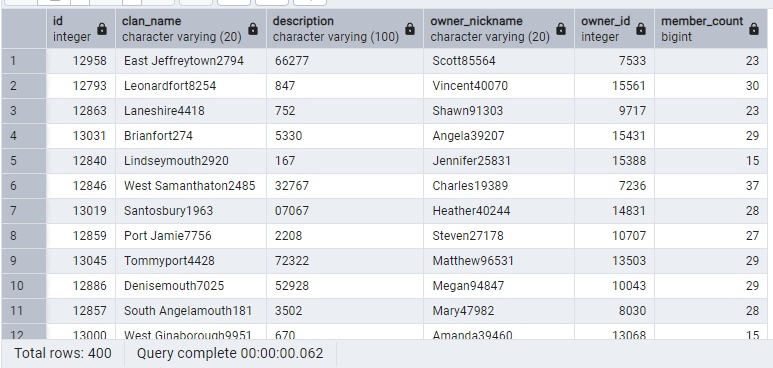
users m ON c.id = m.clan\_id

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

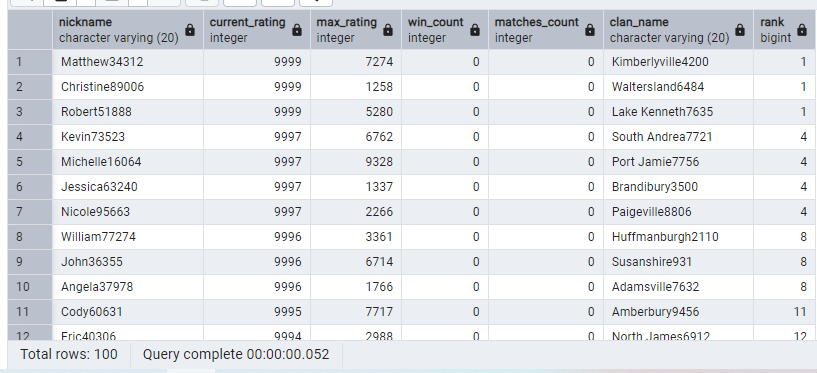
ORDER BY

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

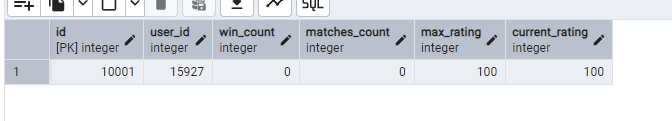
INSERT INTO users (nickname, login, password\_hash)

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.