ER-модель

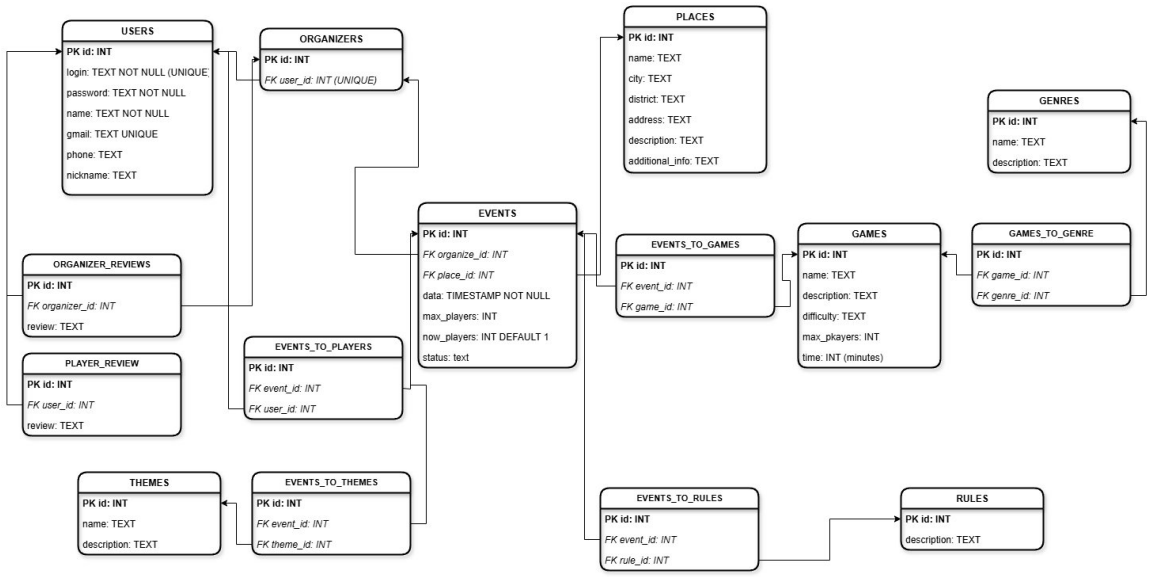
ОСНОВНЫЕ ТАБЛИЦЫ (16 таблиц):
• USERS - пользователи (name, gmail, phone, nickname)
• ORGANIZERS - организаторы (связь 1:1 с Users)

• ORGANIZER_REVIEWS - отзывы об организаторах
• PLAYER_REVIEW - отзывы о игроках
• EVENTS - мероприятия (organize_id, place_id, data, max_players, now_players, status)
• PLACES - места проведения (name, address, description, additional_info)
• GAMES - настольные игры (name, description, max_players, difficulty, time)
• GENRES - жанры игр (name, description)
• THEMES - темы мероприятий (name, description)
• RULES - правила мероприятий (description)

ТАБЛИЦЫ СВЯЗЕЙ M:M:
• EVENTS_TO_PLAYERS - связь между Events и Users
• EVENTS_TO_GAMES - связь между Events и Games
• EVENTS_TO_THEMES - связь между Events и Themes
• EVENTS_TO_RULES - связь между Events и Rules
• GAMES_TO_GENRE - связь между Games и Genres

КЛЮЧЕВЫЕ СВЯЗИ В МОДЕЛИ:
• Users ↔ Organizers: 1:1 (один пользователь может быть организатором)
• Organizers → Events: 1:N (один организатор создает много событий)
• Places → Events: 1:N (одно место для много событий)
• Events ↔ Users: M:M (через Events_to_Players - участники мероприятий)
• Events ↔ Games: M:M (через Events_to_Games - игры для каждого события)
• Events ↔ Themes: M:M (через Events_to_Themes - теми мероприятий)
• Events ↔ Rules: M:M (через Events_to_Rules - правила для каждого события)
• Games ↔ Genres: M:M (через Games_to_Genre - жанры игр)
• Organizers ↔ Reviews: 1:N (отзывы об организаторах)
• Users ↔ Reviews: 1:N (отзывы о игроках)

ЦЕЛОСТНОСТЬ ДАННЫХ:
• Все внешние ключи имеют ON DELETE CASCADE (каскадное удаление)
• Check ограничения на max_players > 0, now_players >= 0
• Статус события может быть: 'active', 'completed', 'cancelled'
• Сложность игры может быть: 'easy', 'medium', 'hard'



2. Создание базы данных DDL

```
-- Таблица пользователей
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    login TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL,
    name TEXT NOT NULL,
    gmail TEXT NOT NULL UNIQUE,
    phone TEXT,
    nickname TEXT
);

-- Таблица организаторов
CREATE TABLE organizers (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL UNIQUE REFERENCES users(id) ON DELETE CASCADE
);

-- Таблица мест проведения
CREATE TABLE places (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    city TEXT NOT NULL,
    district TEXT,
    address TEXT NOT NULL,
    description TEXT,
    additional_info TEXT
);

-- Таблица мероприятий
CREATE TABLE events (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    organizer_id INTEGER NOT NULL REFERENCES organizers(id) ON DELETE CASCADE,
```

```
    place_id INTEGER NOT NULL REFERENCES places(id) ON DELETE RESTRICT,
    data TIMESTAMP NOT NULL,
    max_players INTEGER NOT NULL CHECK (max_players > 0),
    now_players INTEGER DEFAULT 0 CHECK (now_players >= 0),
    status TEXT DEFAULT 'active' CHECK (status IN ('active', 'completed', 'cancelled'))
);

-- Таблица участников на мероприятиях (M:M)
CREATE TABLE events_to_players (
    id SERIAL PRIMARY KEY,
    event_id INTEGER NOT NULL REFERENCES events(id) ON DELETE CASCADE,
    user_id INTEGER NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    UNIQUE(event_id, user_id)
);

-- Таблица отзывов об организаторах
CREATE TABLE organizer_reviews (
    id SERIAL PRIMARY KEY,
    organizer_id INTEGER NOT NULL REFERENCES organizers(id) ON DELETE CASCADE,
    review TEXT NOT NULL
);

-- Таблица отзывов о игроках
CREATE TABLE player_review (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    review TEXT NOT NULL
);

-- Таблица тем мероприятий
CREATE TABLE themes (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT
);

-- Таблица связи мероприятий с темами (M:M)
CREATE TABLE events_to_themes (
    id SERIAL PRIMARY KEY,
    event_id INTEGER NOT NULL REFERENCES events(id) ON DELETE CASCADE,
    theme_id INTEGER NOT NULL REFERENCES themes(id) ON DELETE CASCADE,
    UNIQUE(event_id, theme_id)
);

-- Таблица настольных игр
CREATE TABLE games (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    max_players INTEGER CHECK (max_players > 0),
    difficulty TEXT CHECK (difficulty IN ('easy', 'medium', 'hard')),
    time INTEGER
);

-- Таблица связи мероприятий с играми (M:N)
CREATE TABLE events_to_games (
    id SERIAL PRIMARY KEY,
    event_id INTEGER NOT NULL REFERENCES events(id) ON DELETE CASCADE,
    game_id INTEGER NOT NULL REFERENCES games(id) ON DELETE CASCADE,
    UNIQUE(event_id, game_id)
);
```

```
-- Таблица жанров игр
CREATE TABLE genres (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT
);

-- Таблица связи игр с жанрами (M:N)
CREATE TABLE games_to_genre (
    id SERIAL PRIMARY KEY,
    game_id INTEGER NOT NULL REFERENCES games(id) ON DELETE CASCADE,
    genre_id INTEGER NOT NULL REFERENCES genres(id) ON DELETE CASCADE,
    UNIQUE(game_id, genre_id)
);

-- Таблица правил мероприятий
CREATE TABLE rules (
    id SERIAL PRIMARY KEY,
    description TEXT NOT NULL
);

-- Таблица связи мероприятий с правилами (M:N)
CREATE TABLE events_to_rules (
    id SERIAL PRIMARY KEY,
    event_id INTEGER NOT NULL REFERENCES events(id) ON DELETE CASCADE,
    rule_id INTEGER NOT NULL REFERENCES rules(id) ON DELETE CASCADE,
    UNIQUE(event_id, rule_id)
);
```

Примечания к DDL:

- Все таблицы используют SERIAL для автоинкремента первичных ключей
- Внешние ключи установлены с ON DELETE CASCADE для каскадного удаления
- Добавлены CHECK ограничения для валидации данных

3. Удаление базы данных

```
DROP TABLE IF EXISTS events_to_rules CASCADE;
DROP TABLE IF EXISTS rules CASCADE;
DROP TABLE IF EXISTS games_to_genre CASCADE;
DROP TABLE IF EXISTS genres CASCADE;
DROP TABLE IF EXISTS events_to_games CASCADE;
DROP TABLE IF EXISTS games CASCADE;
DROP TABLE IF EXISTS events_to_themes CASCADE;
DROP TABLE IF EXISTS themes CASCADE;
DROP TABLE IF EXISTS player_review CASCADE;
DROP TABLE IF EXISTS organizer_reviews CASCADE;
DROP TABLE IF EXISTS events_to_players CASCADE;
DROP TABLE IF EXISTS events CASCADE;
DROP TABLE IF EXISTS organizers CASCADE;
DROP TABLE IF EXISTS places CASCADE;
DROP TABLE IF EXISTS users CASCADE;

-- Удаление функций и триггеров
DROP FUNCTION IF EXISTS check_event_player_limit() CASCADE;
DROP FUNCTION IF EXISTS update_event_player_count_insert() CASCADE;
DROP FUNCTION IF EXISTS update_event_player_count_delete() CASCADE;
DROP FUNCTION IF EXISTS check_event_date() CASCADE;
DROP FUNCTION IF EXISTS add_organizer_as_participant() CASCADE;
DROP FUNCTION IF EXISTS get_upcoming_events(INT) CASCADE;
DROP FUNCTION IF EXISTS get_event_participants(INT) CASCADE;
DROP PROCEDURE IF EXISTS register_player_on_event(INT, INT) CASCADE;
DROP PROCEDURE IF EXISTS unregister_player_from_event(INT, INT) CASCADE;
DROP FUNCTION IF EXISTS search_events_by_game(TEXT) CASCADE;
DROP FUNCTION IF EXISTS get_organizer_statistics(INT) CASCADE;
```

4. Заполнение тестовыми данными

```
-- Вставка пользователей
INSERT INTO users (login, password, name, gmail, phone, nickname) VALUES
('ivan_user', '1234', 'Иван Иванов', 'ivan@gmail.com', '+79160001122', 'ivan_gamer'),
('maria_user', '1234', 'Мария Петрова', 'maria@gmail.com', '+79260001123', 'mary_plays'),
('petr_user', '1234', 'Петр Сидоров', 'petr@gmail.com', '+79360001124', 'petr_fan'),
('alex_user', '1234', 'Алексей Смирнов', 'alex@gmail.com', '+79460001125', 'alex_pro'),
('svetlana_user', '1234', 'Светлана Морозова', 'svetlana@gmail.com', '+79560001126', 'lana_games');

-- Вставка организаторов
INSERT INTO organizers (user_id) VALUES (1), (2), (3);

-- Вставка мест проведения
INSERT INTO places (name, city, district, address, description, additional_info) VALUES
('Game Cafe', 'Санкт-Петербург', 'Центральный', 'ул. Советская, 5', 'Основная площадка для игр',
'WiFi, чай, кофе'),
('Клуб на Невском', 'Санкт-Петербург', 'Адмиралтейский', 'пр. Невский, 42', 'Центральная локация',
'Вход с торца здания'),
('Galaxy games', 'Санкт-Петербург', 'Красносельский', 'ул. Маршала Казакова, 26', 'Анти-кафе', 'Диван-
зона, проектор, настолки, PS5');

-- Вставка игр
INSERT INTO games (name, description, max_players, difficulty, time) VALUES
    ('Мафия', 'Психологическая игра на выживание', 12, 'medium', 10),
    ('Колонизаторы', 'Экономическая стратегия', 6, 'hard', 90),
    ('Эволюция', 'Стратегическая игра на выживание созданного животного', 6, 'hard', 45),
    ('Code Names', 'Словесная игра в команде', 8, 'medium', 30),
    ('Имаджинариум', 'Игра на ассоциации', 8, 'easy', 45);

-- Вставка жанров
INSERT INTO genres (name, description) VALUES
    ('Стратегия', 'Игры на планирование и управление ресурсами'),
    ('Психология', 'Игры на дедукцию и блеф'),
    ('Словесные', 'Игры со словами и коммуникацией'),
    ('Все против друг друга', 'Игры, где все играют против друг друга');

-- Вставка тем
INSERT INTO themes (name, description) VALUES
    ('Вечер стратегий', 'Серьезные стратегические игры'),
    ('Вечер легких игр', 'Легкие партийные игры'),
    ('Турнир', 'Соревновательный формат');

-- Вставка правил
INSERT INTO rules (description) VALUES
    ('Приносить свои  еду и напитки запрещено'),
    ('Соблюдать тишину'),
    ('18+');

-- Вставка мероприятий
INSERT INTO events (name, description, organizer_id, place_id, data, max_players, now_players, status)
VALUES
('Битва за Катан', 'Играем в Колонизаторов с дополнениями. Только для опытных игроков.', 1, 1, '2026-
02-15 18:00:00', 6, 2, 'active'),
('Старая добрая мафия', 'Классическая мафия с профессиональным ведущим.', 2, 2, '2026-02-16 19:00:00',
12, 1, 'active'),
('Имаджинариум с карточками 18+', 'Имаджинариум с карточками 18+',3, 3, '2026-02-17 17:30:00', 8, 3,
'active');

-- Вставка связи игр с жанрами
INSERT INTO games_to_genre (game_id, genre_id) VALUES
    (1, 2), (2, 1), (3, 1), (4, 3), (5, 4);

-- Вставка связи мероприятий с играми
INSERT INTO events_to_games (event_id, game_id) VALUES
    (1, 2), (1, 3), (2, 1), (2, 4), (3, 5);

-- Вставка связи мероприятий с темами
INSERT INTO events_to_themes (event_id, theme_id) VALUES
    (1, 1), (2, 2), (3, 1);
```

```
-- Вставка связи мероприятий с правилами
INSERT INTO events_to_rules (event_id, rule_id) VALUES
    (1, 1), (1, 2), (2, 2), (3, 1);

-- Вставка участников на мероприятия
INSERT INTO events_to_players (event_id, user_id) VALUES
    (1, 1), (1, 4), (2, 2), (3, 3), (3, 5);

-- Вставка отзывов об организаторах
INSERT INTO organizer_reviews (organizer_id, review) VALUES
    (1, 'Отличный организатор, все прошло гладко'),
    (2, 'Ужасный организатор, сам опоздал, так еще и в правилах не было сказано, что со совей едой и
напитками нельзя, а на месте оказалось, что все наоборот');

-- Вставка отзывов о игроках
INSERT INTO player_review (user_id, review) VALUES
    (4, 'Постоянно шумит и не соблюдает правила проведения мероприятия'),
    (5, 'Отличный  и грамотный игрок в стратегии, мот помочь новичкам разобраться');
```

5. Обеспечение целостности данных (Триггеры)

```
-- Триггер 1: Проверка лимита участников
CREATE OR REPLACE FUNCTION check_event_player_limit()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT now_players FROM events WHERE id = NEW.event_id) >=
       (SELECT max_players FROM events WHERE id = NEW.event_id) THEN
        RAISE EXCEPTION 'Достигнут максимальный лимит участников для мероприятия';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_event_player_limit
BEFORE INSERT ON events_to_players
FOR EACH ROW
EXECUTE FUNCTION check_event_player_limit();

-- Триггер 2: Обновление счетчика при добавлении участника
CREATE OR REPLACE FUNCTION update_event_player_count_insert()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE events SET now_players = now_players + 1 WHERE id = NEW.event_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_player_count_insert
AFTER INSERT ON events_to_players
FOR EACH ROW
EXECUTE FUNCTION update_event_player_count_insert();

-- Триггер 3: Обновление счетчика при удалении участника
CREATE OR REPLACE FUNCTION update_event_player_count_delete()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE events SET now_players = now_players - 1 WHERE id = OLD.event_id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_player_count_delete
AFTER DELETE ON events_to_players
FOR EACH ROW
EXECUTE FUNCTION update_event_player_count_delete();

-- Триггер 4: Проверка датировки мероприятия (не в прошлом)
CREATE OR REPLACE FUNCTION check_event_date()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.data < NOW() THEN
        RAISE EXCEPTION 'Дата мероприятия не может быть в прошлом';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_event_date
BEFORE INSERT OR UPDATE ON events
FOR EACH ROW
EXECUTE FUNCTION check_event_date();

-- Триггер 5: Организатор автоматически становится участником
CREATE OR REPLACE FUNCTION add_organizer_as_participant()
RETURNS TRIGGER AS $$
DECLARE
    organizer_user_id INTEGER;
BEGIN
    SELECT user_id INTO organizer_user_id FROM organizers WHERE id = NEW.organizer_id;
    INSERT INTO events_to_players (event_id, user_id)
```

```
    VALUES (NEW.id, organizer_user_id)
    ON CONFLICT DO NOTHING;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_add_organizer_as_participant
AFTER INSERT ON events
FOR EACH ROW
EXECUTE FUNCTION add_organizer_as_participant();
```

Описание триггеров:

- trg_check_event_player_limit - проверяет перед добавлением, не превышен ли лимит
- trg_update_player_count_insert - увеличивает счетчик при добавлении
- trg_update_player_count_delete - уменьшает счетчик при удалении
- trg_check_event_date - проверяет, что дата не в прошлом
- trg_add_organizer_as_participant - организатор автоматически участник

6. PL/pgSQL функции и процедуры

```
-- Функция 1: Получить список предстоящих мероприятий
CREATE OR REPLACE FUNCTION get_upcoming_events()
RETURNS TABLE(
    event_id INTEGER,
    event_title TEXT,
    organizer_name TEXT,
    event_date TIMESTAMP,
    event_description TEXT,
    topics_list TEXT,
    rules_list TEXT,
    city TEXT,
    district TEXT,
    address TEXT,
    games_list TEXT,
    current_players INTEGER,
    max_players INTEGER
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        e.id AS event_id,
        e.name AS event_title,
        u.name AS organizer_name,
        e.data AS event_date,
        e.description AS event_description,
        COALESCE(STRING_AGG(DISTINCT t.name, ', '), 'Нет тем') AS topics_list,
        COALESCE(STRING_AGG(DISTINCT r.description, '; '), 'Нет особых правил') AS rules_list,
        p.city,
        p.district,
        p.address,
        COALESCE(STRING_AGG(DISTINCT g.name, ', '), 'Игры не указаны') AS games_list,
        e.now_players,
        e.max_players
    FROM events e
    JOIN organizers o ON e.organizer_id = o.id
    JOIN users u ON o.user_id = u.id
    JOIN places p ON e.place_id = p.id
    LEFT JOIN events_to_themes ett ON e.id = ett.event_id
    LEFT JOIN themes t ON ett.theme_id = t.id
    LEFT JOIN events_to_rules etr ON e.id = etr.event_id
    LEFT JOIN rules r ON etr.rule_id = r.id
    LEFT JOIN events_to_games etg ON e.id = etg.event_id
    LEFT JOIN games g ON etg.game_id = g.id
    WHERE
        e.data > NOW()
        AND e.status = 'active'
    GROUP BY
        e.id, e.name, u.name, e.data, e.description,
        p.city, p.district, p.address, e.now_players, e.max_players
    ORDER BY
        e.data ASC;
END;
$$ LANGUAGE plpgsql;

-- Процедура 1: Записать пользователя на мероприятие
CREATE OR REPLACE PROCEDURE register_player_on_event(
    p_event_id INTEGER, p_user_id INTEGER)
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO events_to_players (event_id, user_id)
    VALUES (p_event_id, p_user_id);
EXCEPTION
    WHEN UNIQUE_VIOLATION THEN
        RAISE EXCEPTION 'Пользователь уже записан на это мероприятие';
    WHEN OTHERS THEN
        RAISE EXCEPTION 'Ошибка при регистрации: %', SQLERRM;
END;
$$;
```

```
-- Процедура 2: Отменить запись пользователя на мероприятие
CREATE OR REPLACE PROCEDURE unregister_player_from_event(
    p_event_id INTEGER, p_user_id INTEGER)
LANGUAGE plpgsql AS $$
BEGIN
    DELETE FROM events_to_players
    WHERE event_id = p_event_id AND user_id = p_user_id;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Запись не найдена';
    END IF;
END;
$$;

-- Функция 2: Поиск мероприятий по названию игры
CREATE OR REPLACE FUNCTION search_events_by_game(p_game_name TEXT)
RETURNS TABLE(
    event_id INTEGER, event_name TEXT, event_description TEXT, event_date TIMESTAMP, place_name TEXT,
    game_name TEXT, free_slots INTEGER
) AS $$
BEGIN
    RETURN QUERY
    SELECT e.id, e.name, e.description, e.data, p.name, g.name, (e.max_players - e.now_players)
    FROM events e
    JOIN places p ON e.place_id = p.id
    JOIN events_to_games etg ON e.id = etg.event_id
    JOIN games g ON etg.game_id = g.id
    WHERE LOWER(g.name) LIKE LOWER('%' || p_game_name || '%')
        AND e.data > NOW() AND e.status = 'active';
END;
$$ LANGUAGE plpgsql;

-- Функция 3: Получить статистику по организатору
CREATE OR REPLACE FUNCTION get_organizer_statistics(p_organizer_id INTEGER)
RETURNS TABLE(
    total_events INTEGER, completed_events INTEGER, cancelled_events INTEGER,
    total_participants INTEGER, avg_rating NUMERIC
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        COUNT(DISTINCT e.id)::INTEGER,
        COUNT(DISTINCT CASE WHEN e.status = 'completed' THEN e.id END)::INTEGER,
        COUNT(DISTINCT CASE WHEN e.status = 'cancelled' THEN e.id END)::INTEGER,
        COUNT(DISTINCT ep.user_id)::INTEGER,
        AVG(CAST(SUBSTRING(orv.review, 1, 1) AS NUMERIC))
    FROM organizers o
    LEFT JOIN events e ON o.id = e.organizer_id
    LEFT JOIN events_to_players ep ON e.id = ep.event_id
    LEFT JOIN organizer_reviews orv ON o.id = orv.organizer_id
    WHERE o.id = p_organizer_id
    GROUP BY o.id;
END;
$$ LANGUAGE plpgsql;
```

Описание функций и процедур:

- get_upcoming_events() - получает список ближайших активных мероприятий(название, его организатора, дата, описание, темы(список тем), правила(список правил), город, район, адрес, игры(список, только название игр) текущее и максимальное кол-во игроков. Списки – строки с перечислением через запятую)
- get_event_participants() - возвращает список участников мероприятия
- register_player_on_event() - процедура регистрации на мероприятие

- unregister_player_from_event() - процедура отмены регистрации
- search_events_by_game() - поиск мероприятий по названию игры
- get_organizer_statistics() - сбор статистики организатора

7. Индексы

Новые индексы

```
CREATE INDEX idx_games_name ON games(name);
CREATE INDEX idx_places_name ON places(name);
CREATE INDEX idx_events_data ON events(data) WHERE status = 'active';
CREATE INDEX idx_users_gmail ON users(gmail);
CREATE INDEX idx_games_to_genre_game ON games_to_genre(game_id);

-- Индексы для авторизации
CREATE INDEX idx_users_login ON users(login);
-- Индексы для поиска по локации
CREATE INDEX idx_places_city ON places(city);
CREATE INDEX idx_places_district ON places(district);
-- Остальные полезные индексы из вашего файла
CREATE INDEX idx_events_data_active ON events(data) WHERE status = 'active';

CREATE INDEX idx_places_city_district ON places(city, district);
```