Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО».

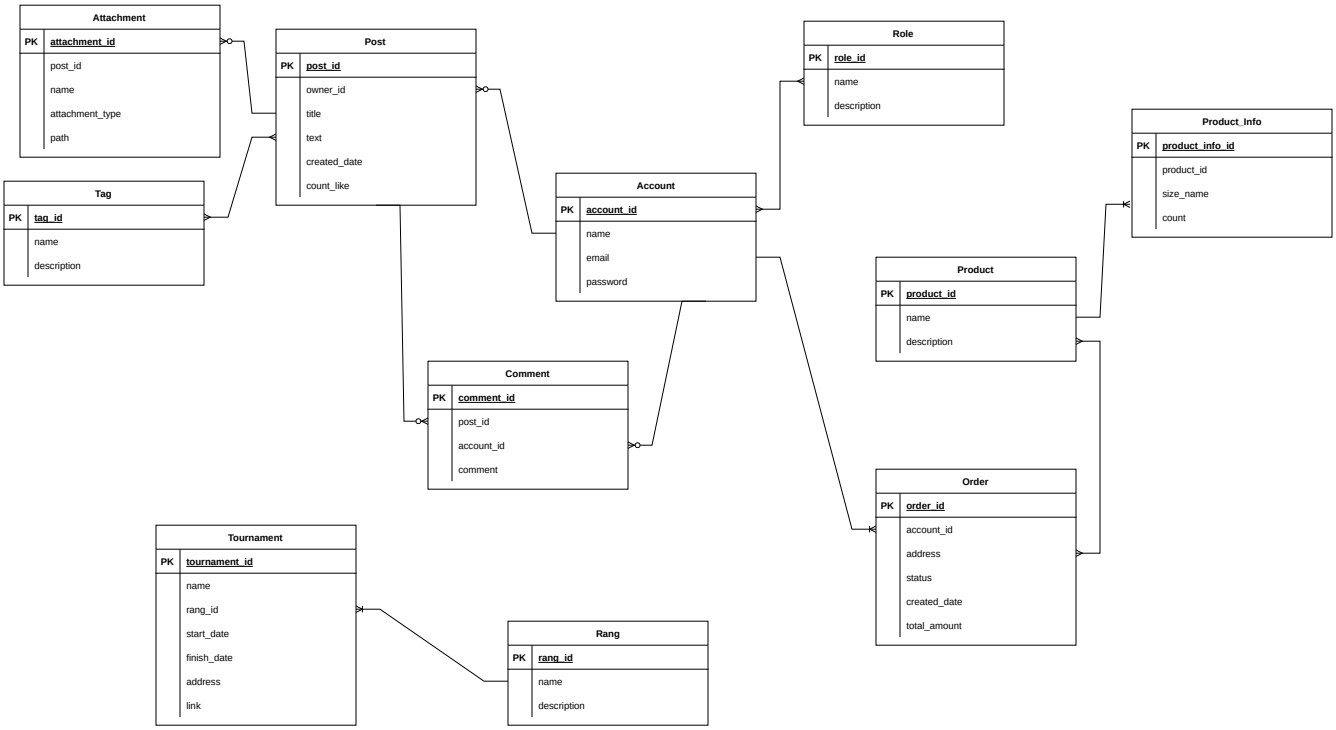
Факультет программной инженерии и компьютерной техники

Курсовая работа
Часть №2

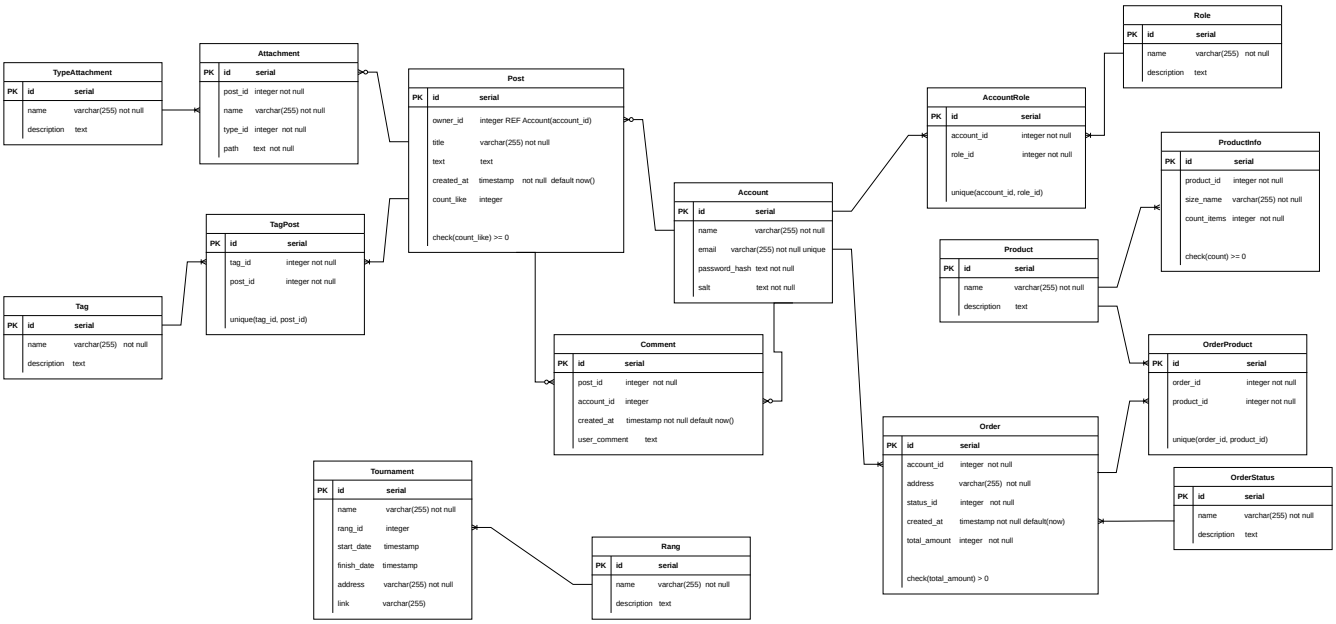
Выполнил
Путинцев Данил
Группа P3307
Проверил(а)
Преподаватель: Хумай Байрамова

Санкт-Петербург 2025 год

Сформировать ER-модель базы данных



Даталогическая модель



Даталогическая модель в реляционной БД PostgreSQL

```
CREATE TABLE "Account" (
    account_id SERIAL PRIMARY KEY,
    name          VARCHAR(255) NOT NULL,
    email         VARCHAR(255) NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    salt          TEXT NOT NULL
);

CREATE TABLE "Post" (
    post_id    SERIAL PRIMARY KEY,
    owner_id   INTEGER REFERENCES "Account"(account_id) ON DELETE SET NULL,
    title      VARCHAR(255) NOT NULL,
    text       TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT now(),
    count_like INTEGER DEFAULT 0 CHECK(count_like >= 0)
```

```
);

CREATE TABLE "Comment" (
    comment_id SERIAL PRIMARY KEY,
    post_id    INTEGER NOT NULL REFERENCES "Post"(post_id) ON DELETE CASCADE,
    account_id INTEGER REFERENCES "Account"(account_id) ON DELETE SET NULL,
    created_at TIMESTAMP NOT NULL DEFAULT now(),
    user_comment TEXT
);

CREATE TABLE "TypeAttachment" (
    type_id     SERIAL PRIMARY KEY,
    name        VARCHAR(255) NOT NULL,
    description TEXT
);


CREATE TABLE "Attachment" (
    attachment_id SERIAL PRIMARY KEY,
    post_id INTEGER NOT NULL REFERENCES "Post"(post_id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    type_id INTEGER REFERENCES "TypeAttachment"(type_id) ON DELETE SET NULL,
    path TEXT NOT NULL
);

CREATE TABLE "Tag" (
    tag_id  SERIAL PRIMARY KEY,
    name    VARCHAR(255) NOT NULL,
    description TEXT
);
```

```
CREATE TABLE "TagPost" (
    tag_post_id SERIAL PRIMARY KEY,
    tag_id      INTEGER NOT NULL REFERENCES "Tag"(tag_id) ON DELETE CASCADE,
    post_id     INTEGER NOT NULL REFERENCES "Post"(post_id) ON DELETE CASCADE,
    UNIQUE(tag_id, post_id)
);

CREATE TABLE "Rang" (
    rang_id     SERIAL PRIMARY KEY,
    name        VARCHAR(255) NOT NULL,
    description TEXT
);

CREATE TABLE "Tournament" (
    tournament_id  SERIAL PRIMARY KEY,
    name           VARCHAR(255) NOT NULL,
    rang_id        INTEGER REFERENCES "Rang"(rang_id) ON DELETE SET NULL,
    start_date     TIMESTAMP,
    finish_date    TIMESTAMP,
    address        VARCHAR(255) NOT NULL,
    link           VARCHAR(255)
);

CREATE TABLE "Role" (
    role_id     SERIAL  PRIMARY KEY,
    name        VARCHAR(255) NOT NULL,
    description TEXT
);
```

```
CREATE TABLE "AccountRole" (
    account_role_id  SERIAL PRIMARY KEY,
    account_id       INTEGER NOT NULL REFERENCES "Account"(account_id) ON DELETE CASCADE,
    role_id          INTEGER NOT NULL REFERENCES "Role"(role_id) ON DELETE CASCADE,
    UNIQUE(account_id, role_id)
);



CREATE TABLE "OrderStatus" (
    status_id   SERIAL PRIMARY KEY,
    name        VARCHAR(255) NOT NULL,
    description TEXT
);

CREATE TABLE "Order" (
    order_id        SERIAL PRIMARY KEY,
    account_id      INTEGER NOT NULL REFERENCES "Account"(account_id) ON DELETE CASCADE,
    address         VARCHAR(255) NOT NULL,
    status_id       INTEGER REFERENCES "OrderStatus"(status_id) ON DELETE SET NULL,
    created_at      TIMESTAMP NOT NULL DEFAULT NOW(),
    total_amount    INTEGER NOT NULL CHECK(total_amount > 0)
);

CREATE TABLE "Product" (
    product_id  SERIAL PRIMARY KEY,
    name        VARCHAR(255) NOT NULL,
    description TEXT
);
```

```
CREATE TABLE "ProductInfo" (
    product_info_id   SERIAL PRIMARY KEY,
    product_id        INTEGER NOT NULL REFERENCES "Product"(product_id) ON DELETE CASCADE,
    size_name         VARCHAR(255) NOT NULL,
    count_items       INTEGER NOT NULL CHECK(count_items >= 0),
    UNIQUE(product_id, size_name)
);


CREATE TABLE "OrderProduct" (
    order_product_id    SERIAL PRIMARY KEY,
    order_id            INTEGER NOT NULL REFERENCES "Order"(order_id) ON DELETE CASCADE,
    product_id          INTEGER NOT NULL REFERENCES "Product"(product_id) ON DELETE CASCADE,
    UNIQUE(order_id, product_id)
);
```

Обеспечение целостности данных при помощи языка DDL и триггеров

```
CREATE OR REPLACE FUNCTION check_tournament_dates()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.finish_date < NEW.start_date THEN
        RAISE EXCEPTION 'Дата завершения не может быть меньше даты начала соревнований';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_check_tournament_dates
    BEFORE INSERT OR UPDATE ON "Tournament"
    FOR EACH ROW EXECUTE FUNCTION check_tournament_dates();

CREATE OR REPLACE FUNCTION update_product_count()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "ProductInfo"
    SET count_items = count_items - 1
    WHERE product_id = NEW.product_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_product_count
    AFTER INSERT ON "OrderProduct"
    FOR EACH ROW EXECUTE FUNCTION update_product_count();

CREATE OR REPLACE FUNCTION check_product_availability()
RETURNS TRIGGER AS $$
DECLARE
    available_count INTEGER;
BEGIN
    SELECT count_items INTO available_count
    FROM "ProductInfo"
    WHERE product_id = NEW.product_id;

    IF available_count <= 0 THEN
        RAISE EXCEPTION 'Товар не найден';
```

```
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_check_product_availability
    BEFORE INSERT ON "OrderProduct"
    FOR EACH ROW EXECUTE FUNCTION check_product_availability();
```

Скрипты для создания, удаления базы данных, заполнение базы тестовыми данными

Скрипт для создания БД:

```
CREATE DATABASE platform_db;
```

Скрипт для удаления БД:

```
DROP DATABASE platform_db;
```

Скрипт для заполнения базы тестовыми данными:

```
INSERT INTO "Account" (name, email, password_hash, salt) VALUES
('Данил Путинцев', 'admin@mail.ru', 'hash21', 'salt21'),
('Алина Кабаева', 'alina.kabaeva@rg.ru', 'hash1', 'salt1'),
('Маргарита Мамун', 'rita.mamun@rg.ru', 'hash2', 'salt2'),
('Яна Кудрявцева', 'yana.kudryavtseva@rg.ru', 'hash3', 'salt3'),
('Дина Аверина', 'dina.averina@rg.ru', 'hash4', 'salt4'),
('Арина Аверина', 'arina.averina@rg.ru', 'hash5', 'salt5'),
('Ирина Винер', 'irina.viner@rg.ru', 'hash6', 'salt6'),
('Амина Зарипова', 'amina.zaripova@rg.ru', 'hash7', 'salt7');

INSERT INTO "Role" (name, description) VALUES
('OAPI:ROLE:PublishPost', 'Роль для публикации постов'),
('OAPI:ROLE:DeletePost', 'Роль для удаления постов'),
```

('OAPI:ROLE:EditPost', 'Роль для редактирования постов'),

('OAPI:ROLE:PublishTournament', 'Роль для публикации соревнований'),

('OAPI:ROLE:EditTournament', 'Роль для редактирования соревнования'),

('OAPI:ROLE:DeleteTournament', 'Роль для удаления соревнования'),

('OAPI:ROLE:PublishProduct', 'Роль для публикации карточки товара'),

('OAPI:ROLE:EditProduct', 'Роль для редактирования карточки товара'),

('OAPI:ROLE:DeleteProduct', 'Роль для удаления карточки товара'),

('OAPI:ROLE:UpdateProductInfo', 'Роль для обновления информации о количестве товаров'),

('OAPI:ROLE:GetProductInfo', 'Роль для получения информации о количестве товаров'),

('OAPI:ROLE:BlockAccount', 'Роль для блокирования пользователей')

;

INSERT INTO "AccountRole" (account_id, role_id) VALUES

(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11), (1, 12),

(2, 2),

(3, 2),

(4, 2),

(5, 2),

(6, 3), (6, 5),

(7, 3), (7, 4);

INSERT INTO "Rang" (name, description) VALUES

('клубный', 'Клубные соревнования'),

('городской', 'Городские соревнования'),

('региональный', 'Региональные соревнования'),

('всероссийский', 'Всероссийские соревнования'),

('международный', 'Международные соревнования'),

('Чемпионат мира', 'Чемпионат мира по художественной гимнастике'),

('Олимпийские игры', 'Олимпийские игры');

INSERT INTO "Tournament" (name, rang_id, start_date, finish_date, address, link) VALUES

('Чемпионат Москвы', 2, '2026-03-15', '2026-03-17', 'Москва, Лужники', 'http://mosgymnastics.ru'),

('Кубок России', 4, '2026-04-10', '2026-04-15', 'Казань, Дворец гимнастики', 'http://rusgymnastics.ru/cup'),

('Гран-При Москва', 5, '2026-05-20', '2026-05-25', 'Москва, Олимпийский', 'http://grandprix-moscow.ru'),

('Чемпионат Европы', 5, '2026-06-01', '2026-06-05', 'Будапешт, Будапешт Арена', 'http://ueg.org'),

('Олимпийские игры 2024', 7, '2026-07-27', '2026-08-05', 'Париж, Берси Арена', 'http://olympics.com');

INSERT INTO "Post" (owner_id, title, text, count_like) VALUES

(6, 'Новая программа подготовки', 'Представляем новую методику подготовки гимнасток к олимпийскому сезону...', 2),

(1, 'Мои воспоминания об Афинах 2004', 'Олимпиада в Афинах стала важнейшим событием в моей карьере...', 3),

(4, 'Тренировка с булавами - советы', 'Хочу поделиться секретами работы с булавами...', 0),

(7, 'Изменения в правилах на 2024 год', 'FIG внесла изменения в правила судейства...', 1),

(2, 'Возвращение в большой спорт', 'После перерыва возвращаюсь к соревнованиям...', 4);

INSERT INTO "Tag" (name, description) VALUES

('художественная гимнастика', 'Все о художественной гимнастике'),

('соревнования', 'Соревнования и турниры'),

('тренировки', 'Методики тренировок'),

('инвентарь', 'Снаряды и оборудование'),

('булавы', 'Работа с булавами'),

('лента', 'Упражнения с лентой'),

('мяч', 'Упражнения с мячом'),

('обруч', 'Упражнения с обручем'),

('олимпиада', 'Олимпийские игры'),

('чемпионат мира', 'Чемпионаты мира'),

('тренерские советы', 'Советы от тренеров'),

('выступления', 'Видео выступлений'),

('здоровье', 'Здоровье и восстановление'),

('питание', 'Питание гимнасток'),

('растяжка', 'Упражнения на гибкость');

INSERT INTO "TagPost" (tag_id, post_id) VALUES

(1, 1), (3, 1), (11, 1),

(1, 2), (9, 2), (2, 2),

(1, 3), (5, 3), (3, 3),

(1, 4), (2, 4), (11, 4),

(1, 5), (2, 5), (13, 5);

INSERT INTO "Comment" (post_id, account_id, user_comment) VALUES

(1, 2, 'Отличная методика! Уже применяю на тренировках.'),

(1, 3, 'Спасибо за полезные советы, Ирина Александровна!'),

(2, 4, 'Алина, вы вдохновляете новое поколение гимнасток!'),

(3, 5, 'Диночка, покажи еще упражнения с булавами!'),

(4, 1, 'Важные изменения, нужно изучать всем гимнасткам.'),

(5, 6, 'Желаю успешного возвращения, Маргарита!');

INSERT INTO "TypeAttachment" (name, description) VALUES

('image', 'Изображение'),

('video', 'Видео'),

('file', 'Файлы'),

('music', 'Музыка');

INSERT INTO "Attachment" (post_id, name, type_id, path) VALUES

(1, 'training_program.pdf', 3, '/attachments/training_program.pdf'),

(2, 'athens_2004.jpg', 1, '/attachments/athens_2004.jpg'),

(3, 'clubs_training.mp4', 2, '/attachments/clubs_training.mp4'),

(4, 'new_rules_2024.pdf', 3, '/attachments/new_rules_2024.pdf'),

(5, 'comeback_interview.mp4', 2, '/attachments/comeback_interview.mp4');

INSERT INTO "OrderStatus" (name, description) VALUES

('pending', 'Ожидает обработки'),

('processing', 'В обработке'),

('shipped', 'Отправлен'),

('delivered', 'Доставлен'),

('cancelled', 'Отменен');

INSERT INTO "Product" (name, description) VALUES

('Булавы Sasaki Pro', 'Профессиональные булавы для соревнований'),

('Гимнастическая лента Silk', 'Шелковая лента с палкой'),

('Мяч для художественной гимнастики', 'Профессиональный мяч 18см'),

('Обруч профессиональный', 'Обруч 85см для взрослых гимнасток'),

('Купальник для выступлений', 'Кристалл-купальник ручной работы'),

('Гимнастические полутапочки', 'Кожаные полутапочки для тренировок'),

('Чехол для снарядов', 'Чехол для переноски снарядов'),

('Массажный ролл', 'Ролл для восстановления после тренировок');

INSERT INTO "ProductInfo" (product_id, size_name, count_items) VALUES

-- Булавы

(1, 'пара', 25),

-- Ленты

(2, '5м', 30), (2, '6м', 25),

-- Мячи

(3, '18см', 40),

-- Обручи

(4, '85см', 35),

-- Купальники

(5, 'XS', 10), (5, 'S', 15), (5, 'M', 12), (5, 'L', 8),

-- Полутапочки

(6, '32', 20), (6, '34', 25), (6, '36', 30), (6, '38', 15),

-- Чехлы

(7, 'универсальный', 50),

-- Массажные роллы

(8, 'стандартный', 20);

INSERT INTO "Order" (account_id, address, status_id, total_amount) VALUES

(2, 'Москва, ул. Спортивная, 15', 4, 12500),

(3, 'Москва, пр. Мира, 28', 3, 8700),

(4, 'Нижний Новгород, ул. Гимнастическая, 5', 2, 15600),

(5, 'Москва, ул. Тренерская, 12', 1, 9800);

INSERT INTO "OrderProduct" (order_id, product_id) VALUES

(1, 1), (1, 5),

(2, 2), (2, 6),

(3, 3), (3, 4), (3, 5),

(4, 7), (4, 8);

Скрипт для удаление таблиц:

DROP TABLE IF EXISTS "OrderProduct" CASCADE;

DROP TABLE IF EXISTS "Order" CASCADE;

DROP TABLE IF EXISTS "ProductInfo" CASCADE;

DROP TABLE IF EXISTS "Product" CASCADE;

DROP TABLE IF EXISTS "OrderStatus" CASCADE;

DROP TABLE IF EXISTS "AccountRole" CASCADE;

DROP TABLE IF EXISTS "Role" CASCADE;

DROP TABLE IF EXISTS "Tournament" CASCADE;

```
DROP TABLE IF EXISTS "Rang" CASCADE;

DROP TABLE IF EXISTS "Attachment" CASCADE;
DROP TABLE IF EXISTS "TypeAttachment" CASCADE;

DROP TABLE IF EXISTS "TagPost" CASCADE;
DROP TABLE IF EXISTS "Tag" CASCADE;

DROP TABLE IF EXISTS "Comment" CASCADE;
DROP TABLE IF EXISTS "Post" CASCADE;

DROP TABLE IF EXISTS "Account" CASCADE;

DROP SEQUENCE IF EXISTS account_account_id_seq CASCADE;
DROP SEQUENCE IF EXISTS post_post_id_seq CASCADE;
DROP SEQUENCE IF EXISTS comment_comment_id_seq CASCADE;
DROP SEQUENCE IF EXISTS typeattachment_type_id_seq CASCADE;
DROP SEQUENCE IF EXISTS attachment_attachment_id_seq CASCADE;
DROP SEQUENCE IF EXISTS tag_tag_id_seq CASCADE;
DROP SEQUENCE IF EXISTS tagpost_tag_post_id_seq CASCADE;
DROP SEQUENCE IF EXISTS rang_rang_id_seq CASCADE;
DROP SEQUENCE IF EXISTS tournament_tournament_id_seq CASCADE;
DROP SEQUENCE IF EXISTS role_role_id_seq CASCADE;
DROP SEQUENCE IF EXISTS accountrole_account_role_id_seq CASCADE;
DROP SEQUENCE IF EXISTS orderstatus_status_id_seq CASCADE;
DROP SEQUENCE IF EXISTS order_order_id_seq CASCADE;
DROP SEQUENCE IF EXISTS product_product_id_seq CASCADE;
DROP SEQUENCE IF EXISTS productinfo_product_info_id_seq CASCADE;
DROP SEQUENCE IF EXISTS orderproduct_order_product_id_seq CASCADE;
```

PL/PGSQL функции и процедуры для выполнения критически важных запросов

```
create procedure ban_user(IN p_account_id integer)
    language plpgsql
as
$$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM "Account" WHERE id = p_account_id) THEN
        RAISE EXCEPTION 'Пользователь с ID % не найден', p_account_id;
    END IF;

    DELETE FROM "AccountRole" WHERE account_id = p_account_id;
    DELETE FROM "Post" WHERE owner_id = p_account_id;
    DELETE FROM "Comment" WHERE account_id = p_account_id;

    RAISE NOTICE 'Пользователь с ID % заблокирован. Все роли, посты и комментарии удалены.', p_account_id;

EXCEPTION
    WHEN OTHERS THEN
        RAISE EXCEPTION 'Ошибка при блокировке пользователя: %', SQLERRM;
END;
$$;



create function create_comment(p_user_comment text, p_post_id integer, p_account_id integer) returns integer
    language plpgsql
as
```

```
$$
DECLARE
    v_comment_id INTEGER;
    v_account_name varchar;
BEGIN
    IF p_user_comment IS NULL OR TRIM(p_user_comment) = '' THEN
        RAISE EXCEPTION 'Комментарий не может быть пустым';
    END IF;

    IF p_post_id IS NULL OR NOT EXISTS (
        SELECT 1 FROM "Post" p WHERE p.id = p_post_id
    ) THEN
        RAISE EXCEPTION 'Указанный пост не существует';
    END IF;

    IF p_account_id IS NULL OR NOT EXISTS (
        SELECT 1 FROM "Account" a WHERE a.id = p_account_id
    ) THEN
        RAISE EXCEPTION 'Указанный аккаунт не существует';
    END IF;

    INSERT INTO "Comment" (created_at, user_comment, account_id, post_id)
    VALUES (NOW(), p_user_comment, p_account_id, p_post_id)
    RETURNING "Comment".id INTO v_comment_id;

    RETURN v_comment_id;

END;
$$;
```

```
create function insert_rang(p_name character varying, p_description character varying DEFAULT NULL::character varying) returns integer
    language plpgsql
as
$$
DECLARE
    v_id INTEGER;
    v_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM "Rang" WHERE name = p_name;

    IF v_count > 0 THEN
        RAISE EXCEPTION 'Rang with name % already exists', p_name;
    END IF;

    INSERT INTO "Rang" (name, description)
    VALUES (p_name, p_description)
    RETURNING id INTO v_id;

    RETURN v_id;
END;
$$;
```

Создание индексов на основе анализа использования базы данных в контексте описанных прецедентов

```
-- Для быстрого поиска по email при регистрации и входе
```

```
CREATE UNIQUE INDEX idx_account_email_lower_hash ON "Account" USING HASH
(LOWER(email));


-- Для быстрого получения постов с сортировкой по дате

CREATE INDEX idx_post_created_at_desc ON "Post" (created_at DESC);


-- Для сортировки соревнований по дате начала

CREATE INDEX idx_tournament_start_date ON "Tournament" (start_date);

CREATE INDEX idx_tournament_start_date_desc ON "Tournament" (start_date DESC);


-- Для поиска товаров по названию

CREATE INDEX idx_product_name ON "Product" (name);
```

Вывод

В результате выполнения второго этапа курсовой работы была разработана полноценная реляционная база данных на основе предварительно сформулированной ER- и даталогической модели базы данных для будущей информационной системы. При реализации функции и создания индексов в реляционной БД PostgreSQL, я несколько раз пересмотрел даталогическую модель, добавив недостающие поля и ограничения. На основе проделанной работы была создана структура базы данных, которая отвечает поставленным требованиям, гарантирует необходимую целостность хранения данных