МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ИТМО»

ФАКУЛЬТЕТ ПИиКТ

КУРСОВАЯ РАБОТА

по дисциплине

«ИНФОРМАЦИОННЫЕ СИСТЕМЫ»

Этап 2

Выполнили:

Студенты группы Р3317

Самсонов Д. А.

Мищенко Р. А

Преподаватель:

Николаев Владимир Вячеславович

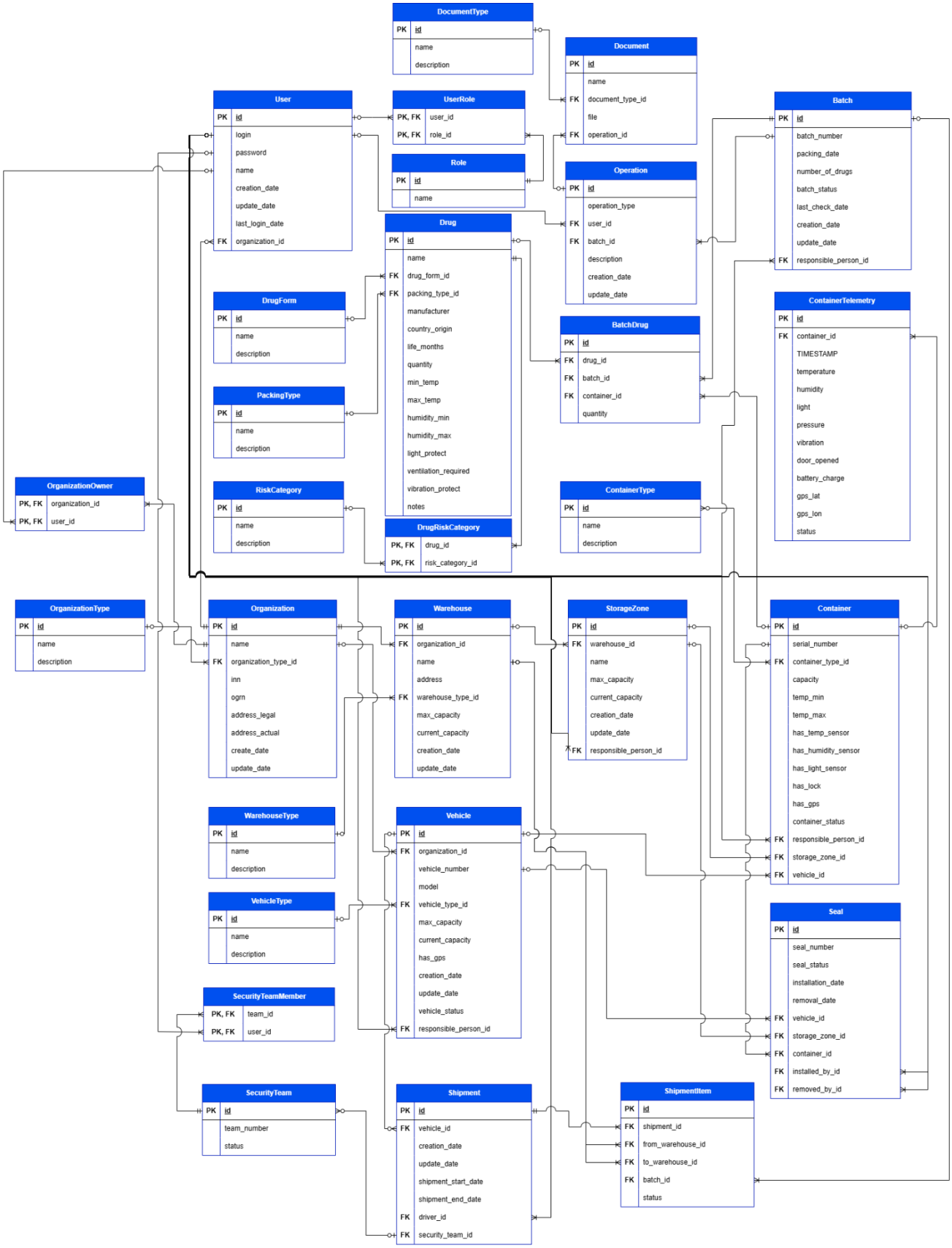
Санкт-Петербург, 2025

Оглавление

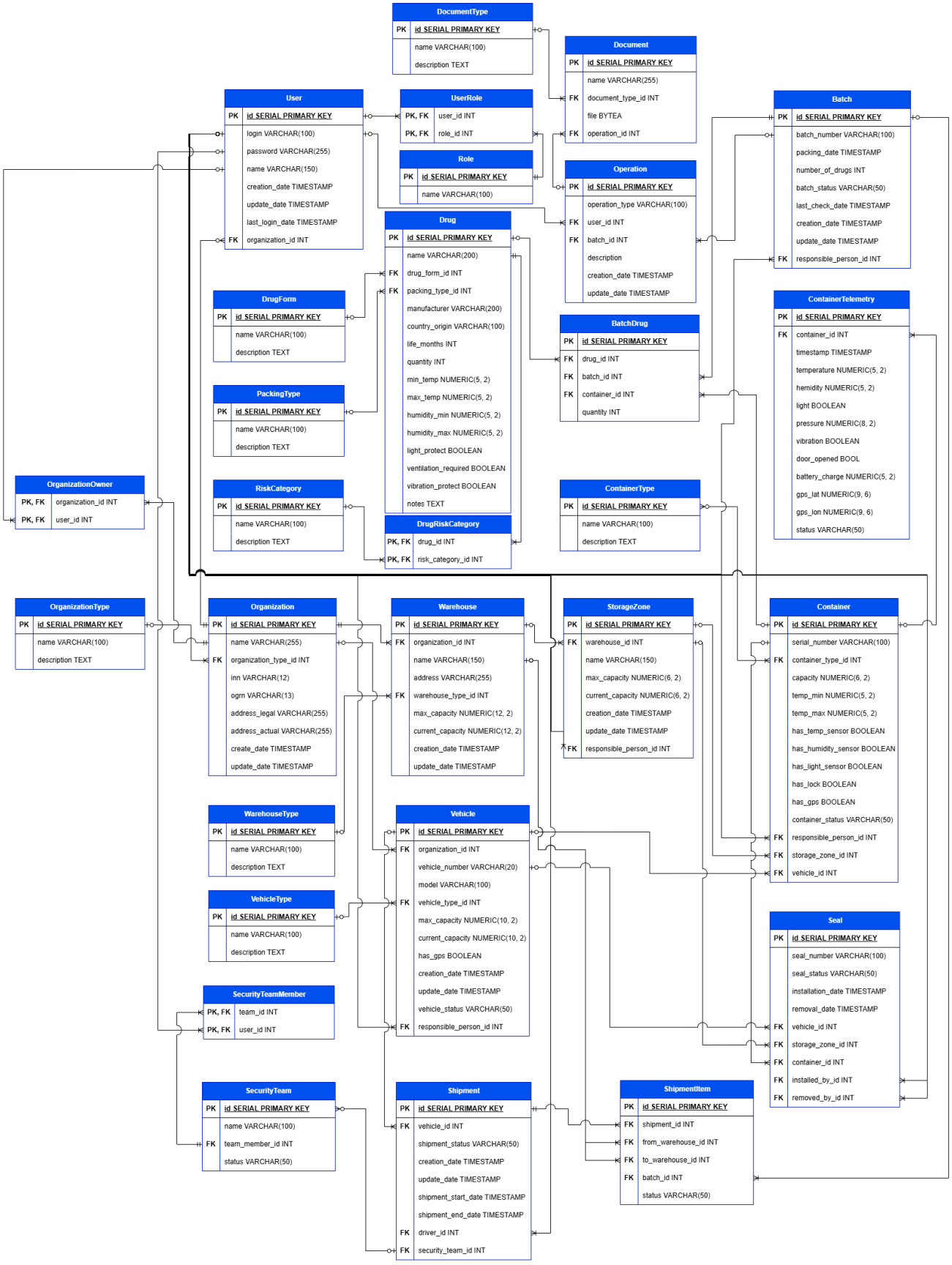
Задание

1. Сформировать ER-модель базы данных (на основе описаний предметной области и прецедентов из предыдущего этапа).
   ER-модель должна:
   a. включать в себя не менее 10 сущностей;
   b. содержать хотя бы одно отношение вида «многие-ко-многим».
2. Согласовать ER-модель с преподавателем. На основе ER-модели построить даталогическую модель.
3. Реализовать даталогическую модель в реляционной СУБД PostgreSQL.
4. Обеспечить целостность данных при помощи средств языка DDL и триггеров.
5. Реализовать скрипты для создания, удаления базы данных, заполнения базы тестовыми данными.
6. Предложить pl/pgsql-функции и процедуры, для выполнения критически важных запросов (которые потребуются при последующей реализации прецедентов).
7. Создать индексы на основе анализа использования базы данных в контексте описанных на первом этапе прецедентов. Обосновать полезность созданных индексов для реализации представленных на первом этапе бизнес-процессов.
8. Составить отчет.

ER-модель


DocumentType
PK id
name
description
Document
PK id
name
FK document_type_id
file
FK operation_id
User
PK id
login
password
name
creation_date
update_date
last_login_date
FK organization_id
UserRole
PK, FK user_id
PK, FK role_id
Role
PK id
name
Operation
PK id
operation_type
FK user_id
FK batch_id
description
creation_date
update_date
Batch
PK id
batch_number
packing_date
number_of_drugs
batch_status
last_check_date
creation_date
update_date
FK responsible_person_id
Drug
PK id
name
FK drug_form_id
FK packing_type_id
manufacturer
country_origin
life_months
quantity
min_temp
max_temp
humidity_min
humidity_max
light_protect
ventilation_required
vibration_protect
notes
DrugForm
PK id
name
description
BatchDrug
PK id
FK drug_id
FK batch_id
FK container_id
quantity
ContainerTelemetry
PK id
FK container_id
TIMESTAMP
temperature
humidity
light
pressure
vibration
door_opened
battery_charge
gps_lat
gps_lon
status
PackingType
PK id
name
description
OrganizationOwner
PK, FK organization_id
PK, FK user_id
RiskCategory
PK id
name
description
DrugRiskCategory
PK, FK drug_id
PK, FK risk_category_id
ContainerType
PK id
name
description
OrganizationType
PK id
name
description
Organization
PK id
name
FK organization_type_id
inn
ogrn
address_legal
address_actual
create_date
update_date
Warehouse
PK id
FK organization_id
name
address
FK warehouse_type_id
max_capacity
current_capacity
creation_date
update_date
StorageZone
PK id
FK warehouse_id
name
max_capacity
current_capacity
creation_date
update_date
FK responsible_person_id
Container
PK id
serial_number
FK container_type_id
capacity
temp_min
temp_max
has_temp_sensor
has_humidity_sensor
has_light_sensor
has_lock
has_gps
container_status
FK responsible_person_id
FK storage_zone_id
FK vehicle_id
WarehouseType
PK id
name
description
Vehicle
PK id
FK organization_id
vehicle_number
model
FK vehicle_type_id
max_capacity
current_capacity
has_gps
creation_date
update_date
vehicle_status
FK responsible_person_id
VehicleType
PK id
name
description
Seal
PK id
seal_number
seal_status
installation_date
removal_date
FK vehicle_id
FK storage_zone_id
FK container_id
FK installed_by_id
FK removed_by_id
SecurityTeamMember
PK, FK team_id
PK, FK user_id
SecurityTeam
PK id
team_number
status
Shipment
PK id
FK vehicle_id
creation_date
update_date
shipment_start_date
shipment_end_date
FK driver_id
FK security_team_id
ShipmentItem
PK id
FK shipment_id
FK from_warehouse_id
FK to_warehouse_id
FK batch_id
status

Даталогическая модель



Реализация модели в PostgreSQL

```
CREATE TABLE organization_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE organization (
    id SERIAL PRIMARY KEY,
    organization_type_id INT NOT NULL REFERENCES organization_type(id) ON
UPDATE CASCADE ON DELETE RESTRICT,
    name VARCHAR(150) NOT NULL,
    address VARCHAR(255),
    address_legal VARCHAR(255),
    inn VARCHAR(12) NOT NULL,
    ogrn VARCHAR(13) NOT NULL,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP
);
CREATE TABLE role (
    id SERIAL PRIMARY KEY,
    name varchar(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE "user" (
    id SERIAL PRIMARY KEY,
    login VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    last_login_date TIMESTAMP,
    organization_id INT REFERENCES organization(id) ON UPDATE CASCADE ON
DELETE SET NULL
);
CREATE TABLE user_role (
    user_id INT NOT NULL REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE
CASCADE,
    role_id INT NOT NULL REFERENCES role(id) ON UPDATE  CASCADE ON DELETE
RESTRICT,
    PRIMARY KEY (user_id, role_id)
);
CREATE TABLE organization_owner (
    organization_id INT NOT NULL REFERENCES organization(id) ON UPDATE
CASCADE ON DELETE CASCADE,
    user_id INT NOT NULL REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE
CASCADE,
    PRIMARY KEY (organization_id, user_id)
);
CREATE TABLE drug_form (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE packing_type (
    id SERIAL PRIMARY KEY,
    name varchar(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE risk_category (
    id SERIAL PRIMARY KEY,
    name varchar(100) UNIQUE NOT NULL,
    description TEXT
);
```

```
CREATE TABLE drug (
    id SERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    drug_form_id INT NOT NULL REFERENCES drug_form(id) ON UPDATE CASCADE ON
DELETE RESTRICT,
    packing_type_id INT NOT NULL REFERENCES packing_type(id) ON UPDATE
CASCADE ON DELETE RESTRICT,
    manufacturer VARCHAR(200),
    country VARCHAR(100),
    packing_date TIMESTAMP,
    life_month INT CHECK ( life_month IS NULL OR life_month >= 0),
    quantity INT NOT NULL CHECK(quantity >= 0),
    min_temp NUMERIC(5, 2),
    max_temp NUMERIC(5, 2),
    humidity_min NUMERIC(5, 2),
    humidity_max NUMERIC(5, 2),
    light_protect BOOLEAN,
    ventilation_required BOOLEAN,
    vibration_protect BOOLEAN,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    notes TEXT,
    CONSTRAINT ck_drug_temp_range CHECK (
        (min_temp IS NULL OR max_temp IS NULL OR min_temp <= max_temp) AND
        (min_temp IS NULL OR min_temp >= -80) AND
        (max_temp IS NULL OR max_temp <= 80)
    ),
    CONSTRAINT ck_drug_humidity_range CHECK (
        (humidity_min IS NULL OR humidity_max IS NULL OR humidity_min <=
humidity_max) AND
        (humidity_min IS NULL OR humidity_min >= 0) AND
        (humidity_max IS NULL OR humidity_max <= 100)
    ),
    CONSTRAINT uq_drug UNIQUE(name, drug_form_id, packing_type_id)
);
CREATE TABLE drug_risk_category (
    id SERIAL PRIMARY KEY,
    drug_id INT NOT NULL REFERENCES drug(id) ON UPDATE CASCADE ON DELETE
CASCADE,
    risk_category_id INT NOT NULL REFERENCES risk_category(id) ON UPDATE
CASCADE ON DELETE RESTRICT,
    CONSTRAINT uq_drug_risk UNIQUE(drug_id, risk_category_id)
);
CREATE TABLE warehouse_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE warehouse (
    id SERIAL PRIMARY KEY,
    organization_id INT NOT NULL REFERENCES organization(id) ON UPDATE
CASCADE ON DELETE CASCADE,
    warehouse_type_id INT NOT NULL REFERENCES warehouse_type(id) ON UPDATE
CASCADE ON DELETE RESTRICT,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255),
    max_capacity NUMERIC (12, 2) CHECK(max_capacity >= 0),
    current_capacity NUMERIC(12, 2) CHECK(current_capacity >= 0),
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    CONSTRAINT uq_warehouse UNIQUE(organization_id, name),
    CONSTRAINT ck_capacity CHECK(
        current_capacity IS NULL AND max_capacity IS NULL OR current_capacity
```

```
<= max_capacity
    )
);
CREATE TABLE storage_zone (
    id SERIAL PRIMARY KEY,
    warehouse_id INT NOT NULL REFERENCES warehouse(id) ON UPDATE CASCADE ON
DELETE CASCADE,
    name VARCHAR(150),
    max_capacity NUMERIC(6, 2) CHECK(max_capacity >= 0),
    current_capacity NUMERIC(6, 2) CHECK(current_capacity >= 0),
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    responsible_person_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    CONSTRAINT uq_storage_zone UNIQUE(warehouse_id, name),
    CONSTRAINT ck_capacity CHECK(
        current_capacity IS NULL AND max_capacity IS NULL OR current_capacity
<= max_capacity
    )
);
CREATE TABLE vehicle_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE vehicle (
    id SERIAL PRIMARY KEY,
    organization_id INT NOT NULL REFERENCES organization(id) ON UPDATE
CASCADE ON DELETE CASCADE,
    vehicle_number VARCHAR(20) NOT NULL,
    model VARCHAR(100),
    vehicle_type INT NOT NULL REFERENCES vehicle_type(id) ON UPDATE CASCADE
ON DELETE RESTRICT,
    max_capacity NUMERIC(10, 2) CHECK(max_capacity >=0),
    current_capacity NUMERIC(10, 2) CHECK (current_capacity >=0),
    has_gps BOOLEAN,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    vehicle_status VARCHAR(50),
    responsible_person_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    CONSTRAINT uq_vehicle UNIQUE(organization_id, vehicle_number),
    CONSTRAINT ck_vehicle_status CHECK(COALESCE(vehicle_status, '') IN
        ('IDLE',
        'PREPARING',
        'LOADED',
        'IN_TRANSIT',
        'STOPPED',
        'ARRIVED',
        'UNLOADED',
        'RETURNING',
        'REPAIR')
    ),
    CONSTRAINT ck_capacity CHECK(
        current_capacity IS NULL AND max_capacity IS NULL OR current_capacity
<= max_capacity
    )
);
CREATE TABLE container_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
```

```
CREATE TABLE container (
    id SERIAL PRIMARY KEY,
    serial_number VARCHAR(100) UNIQUE NOT NULL,
    container_type_id INT NOT NULL REFERENCES container_type(id) ON UPDATE
CASCADE ON DELETE RESTRICT,
    max_capacity NUMERIC(10, 2) CHECK(max_capacity >=0),
    current_capacity NUMERIC(10, 2) CHECK (current_capacity >=0),
    temp_min NUMERIC(5, 2),
    temp_max NUMERIC(5, 2),
    has_temp_sensor BOOLEAN,
    has_humidity_sensor BOOLEAN,
    has_light_sensor BOOLEAN,
    has_lock BOOLEAN,
    has_gps BOOLEAN,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    responsible_person_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    storage_zone_id INT REFERENCES storage_zone(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    vehicle_id INT REFERENCES vehicle(id) ON UPDATE CASCADE ON DELETE SET
NULL,
    CONSTRAINT ck_locate_container CHECK(
        (storage_zone_id IS NOT NULL AND vehicle_id IS NULL ) OR
        (storage_zone_id IS NULL AND vehicle_id IS NOT NULL ) OR
        (storage_zone_id IS NULL AND vehicle_id IS NULL)
    ),
    CONSTRAINT ck_capacity CHECK(
        current_capacity IS NULL AND max_capacity IS NULL OR current_capacity
<= max_capacity
    )
);
CREATE TABLE batch (
    id SERIAL PRIMARY KEY,
    batch_number VARCHAR(100) UNIQUE NOT NULL,
    packing_date TIMESTAMP NOT NULL,
    number_of_drugs INT NOT NULL,
    batch_status VARCHAR(50) NOT NULL,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    responsible_person_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    CONSTRAINT ck_batch_status CHECK(COALESCE(batch_status, '') IN
        ('REGISTERED',
        'VERIFIED',
        'IN_STORAGE',
        'IN_QUARANTINE',
        'PREPARED_FOR_TRANSIT',
        'IN_TRANSIT',
        'DELIVERED',
        'ACCEPTED',
        'REJECTED',
        'DISPOSED')
    )
);
CREATE TABLE operation (
    id SERIAL PRIMARY KEY,
    operation_type VARCHAR(100) NOT NULL,
    batch_id INT REFERENCES batch(id) ON UPDATE CASCADE ON DELETE SET NULL,
    user_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE SET NULL,
    description TEXT,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP
```

```
);
CREATE TABLE document_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
CREATE TABLE document (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    document_type_id INT NOT NULL REFERENCES document_type(id) ON UPDATE
CASCADE ON DELETE RESTRICT,
    file BYTEA,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    operation_id INT NOT NULL REFERENCES operation(id) ON UPDATE CASCADE ON
DELETE CASCADE

);
CREATE TABLE batch_drug (
    id SERIAL PRIMARY KEY,
    drug_id INT NOT NULL REFERENCES drug(id) ON UPDATE CASCADE ON DELETE
RESTRICT,
    batch_id INT NOT NULL REFERENCES batch(id) ON UPDATE CASCADE ON DELETE
CASCADE,
    container_id INT NOT NULL REFERENCES container(id) ON UPDATE CASCADE ON
DELETE RESTRICT,
    quantity INT NOT NULL CHECK(quantity >= 0),
    CONSTRAINT uq_batch_drug UNIQUE(drug_id, batch_id, container_id)
);
CREATE TABLE container_telemetry (
    id SERIAL PRIMARY KEY,
    container_id INT NOT NULL REFERENCES container(id) ON UPDATE CASCADE ON
DELETE CASCADE,
    timestamp TIMESTAMP NOT NULL DEFAULT now(),
    temperature NUMERIC(6, 2),
    humidify NUMERIC(6, 2),
    light BOOLEAN,
    vibration BOOLEAN,
    pressure NUMERIC(8, 2),
    door_opened BOOLEAN,
    battery_charge NUMERIC(5, 2),
    gps_latitude NUMERIC(9, 6),
    gps_longitude NUMERIC(9, 6),
    status VARCHAR(50) NOT NULL,
    CONSTRAINT ck_container_telemetry CHECK(COALESCE(status, '') IN
        ('NORMAL',
        'WARNING',
        'ALARM',
        'SENSOR_ERROR',
        'POWER_OFF',
        'INACTIVE')
    ),
    CONSTRAINT ck_telemetry_ranges CHECK(
        (temperature IS NULL OR temperature BETWEEN -80 AND 80)
        AND (humidify IS NULL OR humidify BETWEEN 0 AND 100)
        AND (pressure IS NULL OR pressure BETWEEN 700 AND 1200)
        AND (battery_charge IS NULL OR battery_charge BETWEEN 0 AND 100)
        AND (gps_latitude IS NULL OR gps_latitude BETWEEN -90 AND 90)
        AND (gps_longitude IS NULL OR gps_longitude BETWEEN -180 AND 180)
    )
);
CREATE TABLE seal (
    id SERIAL PRIMARY KEY,
```

```
    seal_number VARCHAR(100) NOT NULL,
    seal_status VARCHAR(50) NOT NULL,
    installation_date TIMESTAMP,
    removal_date TIMESTAMP,
    vehicle_id INT REFERENCES vehicle(id) ON UPDATE CASCADE ON DELETE SET
NULL,
    storage_zone_id INT REFERENCES storage_zone(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    container_id INT REFERENCES container(id) ON UPDATE CASCADE ON DELETE SET
NULL,
    installed_by_id INT NOT NULL REFERENCES "user"(id) ON UPDATE CASCADE ON
DELETE SET NULL,
    removed_by_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE SET
NULL,
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    CONSTRAINT uq_seal_number UNIQUE(seal_number),
    CONSTRAINT ck_seal_status CHECK(COALESCE(seal_status, '') IN
        ('SEALED',
        'INTACT',
        'DAMAGED',
        'REMOVED',
        'LOST',
        'DEFECTIVE',
        'PENDING_INSTALLATION',
        'UNDER_INSPECTION',
        'ARCHIVED')
    ),
    CONSTRAINT ck_locate_seal CHECK (
        (vehicle_id IS NULL AND storage_zone_id IS NULL AND container_id IS
NULL) OR
        (vehicle_id IS NOT NULL AND storage_zone_id IS NULL AND container_id
IS NULL) OR
        (vehicle_id IS NULL AND storage_zone_id IS NOT NULL AND container_id
IS NULL ) OR
        (vehicle_id IS NULL AND storage_zone_id IS NULL AND container_id IS
NOT NULL )
    )
);
CREATE TABLE security_team (
    id SERIAL PRIMARY KEY,
    team_number VARCHAR(100) UNIQUE NOT NULL,
    status VARCHAR(50) NOT NULL,
    CONSTRAINT ch_security_team_status CHECK(COALESCE(status, '') IN
        ('INACTIVE',
        'READY',
        'ASSIGNED',
        'IN_ROUTE',
        'RETURNING',
        'INCIDENT_DETECTED')
    )
);
CREATE TABLE security_team_member (
    team_id INT NOT NULL REFERENCES security_team(id) ON UPDATE CASCADE ON
DELETE CASCADE,
    user_id INT NOT NULL REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE
RESTRICT,
    PRIMARY KEY (team_id, user_id)
);
CREATE TABLE shipment (
    id SERIAL PRIMARY KEY,
    vehicle_id INT REFERENCES vehicle(id) ON UPDATE CASCADE ON DELETE SET
NULL,
```

```
    creation_date TIMESTAMP DEFAULT now(),
    update_date TIMESTAMP,
    shipment_start_date TIMESTAMP,
    shipment_end_date TIMESTAMP,
    driver_id INT REFERENCES "user"(id) ON UPDATE CASCADE ON DELETE SET NULL,
    security_team_id INT REFERENCES security_team(id) ON UPDATE CASCADE ON
DELETE SET NULL
);
CREATE TABLE shipment_item (
    id SERIAL PRIMARY KEY,
    shipment_id INT NOT NULL REFERENCES shipment(id) ON UPDATE CASCADE ON
DELETE RESTRICT,
    from_warehouse_id INT NOT NULL REFERENCES warehouse(id) ON UPDATE CASCADE
ON DELETE RESTRICT,
    to_warehouse_id INT NOT NULL REFERENCES warehouse(id) ON UPDATE CASCADE
ON DELETE RESTRICT,
    batch_id INT NOT NULL REFERENCES batch(id) ON UPDATE CASCADE ON DELETE
RESTRICT,
    status VARCHAR(50) NOT NULL,
    CONSTRAINT uq_shipment_item UNIQUE(shipment_id, from_warehouse_id,
to_warehouse_id, batch_id),
    CONSTRAINT ck_shipment_status CHECK(COALESCE(status, '') IN
        ('REGISTERED',
        'AWAITING_VERIFICATION',
        'VERIFIED',
        'PACKED',
        'LOADED',
        'IN_TRANSIT',
        'ARRIVED',
        'ACCEPTED',
        'REJECTED',
        'IN_QUARANTINE',
        'DISPOSAL')
    )
);
```

Обеспечение целостности данных

Большая часть обеспечения целостности уже была выполнена с помощью встроенного функционала языка DDL. Но нашей модели не хватает ещё некоторых ограничений, которые можно реализовать с помощью триггеров.

```
CREATE OR REPLACE FUNCTION update_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.update_date = now();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_timestamp_organization
BEFORE UPDATE ON organization
FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_user
BEFORE UPDATE ON "user"
FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_drug
    BEFORE UPDATE ON drug
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();
```

```
CREATE TRIGGER trg_update_timestamp_warehouse
    BEFORE UPDATE ON warehouse
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_storage_zone
    BEFORE UPDATE ON storage_zone
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_vehicle
    BEFORE UPDATE ON vehicle
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_container
    BEFORE UPDATE ON container
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_batch
    BEFORE UPDATE ON batch
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_operation
    BEFORE UPDATE ON operation
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_document
    BEFORE UPDATE ON document
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_seal
    BEFORE UPDATE ON seal
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE TRIGGER trg_update_timestamp_shipment
    BEFORE UPDATE ON shipment
    FOR EACH ROW EXECUTE FUNCTION update_timestamp();

CREATE OR REPLACE FUNCTION check_container_capacity()
    RETURNS TRIGGER AS $$
DECLARE
    total NUMERIC;
    capacity NUMERIC;
BEGIN
    SELECT COALESCE(SUM(quantity), 0) INTO total FROM batch_drug WHERE
container_id = NEW.container_id;
    SELECT max_capacity INTO capacity FROM container WHERE id =
NEW.container_id;
    IF capacity IS NOT NULL AND total > capacity THEN
        RAISE EXCEPTION 'Ёмкость контейнера превышена', total, capacity;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_container_capacity
    BEFORE INSERT OR UPDATE ON batch_drug
    FOR EACH ROW EXECUTE FUNCTION check_container_capacity();

CREATE OR REPLACE FUNCTION set_batch_status_on_shipment()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE batch
    SET batch_status = 'IN_TRANSIT', update_date = now()
```

```
    WHERE id = NEW.batch_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_set_batch_status_on_shipment
    AFTER INSERT ON shipment_item
    FOR EACH ROW EXECUTE FUNCTION set_batch_status_on_shipment();

CREATE OR REPLACE FUNCTION prevent_batch_edit_in_transit()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM batch
        WHERE id = NEW.batch_id AND batch_status = 'IN_TRANSIT'
    ) THEN
        RAISE EXCEPTION 'Нельзя изменять данные партии, находящейся в пути';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_prevent_batch_edit_in_transit
    BEFORE UPDATE OR DELETE ON batch_drug
    FOR EACH ROW EXECUTE FUNCTION prevent_batch_edit_in_transit();

CREATE TRIGGER trg_prevent_batch_edit_in_transit
    BEFORE UPDATE OR DELETE ON batch
    FOR EACH ROW EXECUTE FUNCTION prevent_batch_edit_in_transit();

CREATE OR REPLACE FUNCTION check_container_location()
    RETURNS TRIGGER AS $$
BEGIN
    IF NEW.storage_zone_id IS NOT NULL AND NEW.vehicle_id IS NOT NULL THEN
        RAISE EXCEPTION 'Контейнер не может находиться одновременно на складе
и в транспорте';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_container_location
    BEFORE INSERT OR UPDATE ON container
    FOR EACH ROW EXECUTE FUNCTION check_container_location();
```

Скрипты для удаления базы данных, заполнения базы тестовыми данными

Скрипт для удаления таблиц:

```
DO $$
DECLARE
    r RECORD;
BEGIN
    FOR r IN (
        SELECT tablename
        FROM pg_tables
        WHERE schemaname = 's368759'
    )
    LOOP
        EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(r.tablename) || '
CASCADE';
    END LOOP;
END $$;
```

Пример заполнения базы данных значениями:

```
INSERT INTO organization_type (name, description)
VALUES ('Складская организация', 'Осуществляет хранение препаратов'),
       ('Транспортная компания', 'Осуществляет перевозку лекарственных
препаратов');

INSERT INTO organization (organization_type_id, name, address, address_legal,
inn, ogrn)
VALUES (1, 'ФармаСклад №1', 'г. Санкт-Петербург, ул. Тестовая, 1', 'г. Санкт-
Петербург, ул. Тестовая, 1', '780123456789', '1027800000001'),
       (2, 'ТрансФарм', 'г. Москва, ул. Промышленная, 45', 'г. Москва, ул.
Промышленная, 45', '770987654321', '1027700000002');

INSERT INTO role (name, description)
VALUES ('ADMIN', 'Администратор системы'),
       ('WAREHOUSE_EMPLOYEE', 'Сотрудник склада'),
       ('SPECIALIST', 'Фармацевт-контролёр'),
       ('DRIVER', 'Перевозчик'),
       ('SECURITY', 'Охрана');

INSERT INTO "user" (login, password, organization_id)
VALUES ('admin', 'admin123', 1),
       ('sklad1', 'password', 1),
       ('spec1', 'password', 1),
       ('driver1', 'password', 2),
       ('security1', 'password', 2);

INSERT INTO user_role (user_id, role_id)
VALUES (1, 1),
       (2, 2),
       (3, 3),
       (4, 4),
       (5, 5);

INSERT INTO drug_form (name) VALUES ('Таблетки'), ('Раствор для инъекций');
INSERT INTO packing_type (name) VALUES ('Блистер'), ('Флакон');
INSERT INTO risk_category (name) VALUES ('Высокий риск'), ('Контролируемое
вещество'), ('Температурный режим');

INSERT INTO drug (name, drug_form_id, packing_type_id, manufacturer, country,
packing_date, life_month, quantity, min_temp, max_temp, humidity_min,
humidity_max, light_protect, ventilation_required, vibration_protect)
```

```
VALUES ('Инсулин', 2, 2, 'Novo Nordisk', 'Дания', '2025-01-01', 12, 1000, 2,
8, 30, 70, TRUE, FALSE, FALSE),
       ('Амоксициллин', 1, 1, 'KRKA', 'Словения', '2024-07-01', 24, 5000, 15,
25, 30, 65, FALSE, FALSE, FALSE);

INSERT INTO drug_risk_category (drug_id, risk_category_id)
VALUES (1, 2),
       (1, 3),
       (2, 1);

INSERT INTO warehouse_type (name) VALUES ('Обычный склад'), ('Холодильный
склад');
INSERT INTO vehicle_type (name) VALUES ('Рефрижератор'), ('Грузовой фургон');

INSERT INTO warehouse (organization_id, warehouse_type_id, name, address,
max_capacity, current_capacity)
VALUES (1, 2, 'Склад холодильного хранения', 'г. Санкт-Петербург, ул.
Медицинская, 5', 10000, 2000);

INSERT INTO storage_zone (warehouse_id, name, max_capacity, current_capacity)
VALUES (1, 'Холодильная зона', 5000, 1000),
       (1, 'Обычная зона', 5000, 1000);

INSERT INTO container_type (name) VALUES ('Термоконтейнер'), ('Сейф');
INSERT INTO container (serial_number, container_type_id, max_capacity,
current_capacity, temp_min, temp_max, has_temp_sensor, has_humidity_sensor,
has_lock, storage_zone_id)
VALUES ('CNT-001', 1, 500, 100, 2, 8, TRUE, TRUE, TRUE, 1),
       ('CNT-002', 2, 200, 50, 15, 25, FALSE, FALSE, TRUE, 2);

INSERT INTO batch (batch_number, packing_date, number_of_drugs, batch_status,
responsible_person_id)
VALUES ('BATCH-001', '2025-02-01', 1, 'IN_STORAGE', 2),
       ('BATCH-002', '2025-03-01', 1, 'REGISTERED', 3);

INSERT INTO batch_drug (drug_id, batch_id, container_id, quantity)
VALUES (1, 1, 1, 200),
       (2, 2, 2, 100);

INSERT INTO vehicle (organization_id, vehicle_number, model, vehicle_type,
max_capacity, current_capacity, has_gps, vehicle_status,
responsible_person_id)
VALUES (2, 'A123BC178', 'Mercedes Sprinter', 1, 2000, 0, TRUE, 'IDLE', 4);

INSERT INTO security_team (team_number, status)
VALUES ('SEC-001', 'READY');

INSERT INTO security_team_member (team_id, user_id)
VALUES (1, 5);

INSERT INTO shipment (vehicle_id, driver_id, security_team_id,
shipment_start_date)
VALUES (1, 4, 1, now());

INSERT INTO shipment_item (shipment_id, from_warehouse_id, to_warehouse_id,
batch_id, status)
VALUES (1, 1, 1, 1, 'REGISTERED');

INSERT INTO document_type (name, description)
VALUES ('Акт приёмки', 'Документ подтверждения поступления препаратов'),
       ('Акт пломбирования', 'Документ, фиксирующий установку пломб'),
       ('Маршрутный лист', 'Документ, описывающий путь транспортировки'),
       ('Отчёт мониторинга', 'Данные с датчиков за период перевозки');
```

```
INSERT INTO operation (operation_type, batch_id, user_id, description)
VALUES ('RECEIVE_BATCH', 1, 2, 'Приёмка партии BATCH-001 на склад'),
       ('VERIFY_BATCH', 1, 3, 'Проверка условий хранения и документации'),
       ('LOAD_BATCH', 1, 4, 'Погрузка партии в транспортное средство'),
       ('SHIPMENT_START', 1, 4, 'Отправка партии в пункт назначения'),
       ('SHIPMENT_END', 1, 4, 'Доставка партии в пункт назначения');

INSERT INTO document (name, document_type_id, file, operation_id)
VALUES ('Акт приёмки BATCH-001', 1, NULL, 1),
       ('Акт пломбирования CNT-001', 2, NULL, 2),
       ('Маршрутный лист №1', 3, NULL, 4),
       ('Отчёт мониторинга по контейнеру CNT-001', 4, NULL, 5);

INSERT INTO seal (seal_number, seal_status, installation_date, vehicle_id,
installed_by_id)
VALUES ('SEAL-0001', 'SEALED', now(), 1, 5),
       ('SEAL-0002', 'SEALED', now(), NULL, 5);

INSERT INTO container_telemetry (
    container_id, temperature, humidify, light, vibration, pressure,
    door_opened, battery_charge, gps_latitude, gps_longitude, status
)
VALUES
    (1, 4.2, 55.3, FALSE, FALSE, 1013.2, FALSE, 97.5, 59.9390, 30.3158,
'NORMAL'),
    (1, 5.1, 60.2, FALSE, FALSE, 1012.8, FALSE, 95.4, 59.9401, 30.3162,
'NORMAL'),
    (2, 21.0, 40.5, TRUE, FALSE, 1011.9, FALSE, 90.0, 59.9350, 30.3100,
'WARNING');

INSERT INTO shipment_item (shipment_id, from_warehouse_id, to_warehouse_id,
batch_id, status)
VALUES
    (1, 1, 1, 2, 'IN_TRANSIT'),
    (1, 1, 1, 1, 'ARRIVED');
```

Функции и процедуры, для выполнения критически важных запросов.

```
CREATE OR REPLACE FUNCTION verify_batch(p_batch_id INT, p_user_id INT)
    RETURNS VOID AS $$
BEGIN
    UPDATE batch
    SET batch_status = 'VERIFIED', update_date = now()
    WHERE id = p_batch_id;

    INSERT INTO operation(operation_type, batch_id, user_id, description)
    VALUES ('BATCH_VERIFIED', p_batch_id, p_user_id,
            'Партия проверена специалистом');
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION verify_batch(p_batch_id INT, p_user_id INT)
    RETURNS VOID AS $$
BEGIN
    UPDATE batch
    SET batch_status = 'VERIFIED', update_date = now()
    WHERE id = p_batch_id;

    INSERT INTO operation(operation_type, batch_id, user_id, description)
```

```
    VALUES ('BATCH_VERIFIED', p_batch_id, p_user_id,
            'Партия проверена специалистом');
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION archive_expired_batches()
    RETURNS VOID AS $$
BEGIN
    UPDATE batch
    SET batch_status = 'DISPOSED',
        update_date = now()
    WHERE id IN (
        SELECT DISTINCT b.id
        FROM batch b
                 JOIN batch_drug bd ON bd.batch_id = b.id
                 JOIN drug d ON d.id = bd.drug_id
        WHERE b.packing_date + (d.life_month || ' month')::interval < now()
          AND b.batch_status NOT IN ('DISPOSED', 'REJECTED')
    );

    INSERT INTO operation(operation_type, description, creation_date)
    VALUES ('BATCH_DISPOSED', 'Автоматическая утилизация просроченных
партий', now());
END;
$$ LANGUAGE plpgsql;
```

Индексы

```
CREATE INDEX idx_batch_number ON batch(batch_number);
CREATE INDEX idx_batch_status ON batch(batch_status);
CREATE INDEX idx_batch_drug_batch ON batch_drug(batch_id);
CREATE INDEX idx_batch_drug_container ON batch_drug(container_id);

CREATE INDEX idx_container_telemetry_container_time ON
container_telemetry(container_id, timestamp DESC);
CREATE INDEX idx_container_telemetry_status ON container_telemetry(status);

CREATE INDEX idx_container_vehicle ON container(vehicle_id);
CREATE INDEX idx_container_storage_zone ON container(storage_zone_id);

CREATE INDEX idx_shipment_item_shipment ON shipment_item(shipment_id);
CREATE INDEX idx_shipment_item_status ON shipment_item(status);

CREATE INDEX idx_user_login ON "user"(login);

CREATE INDEX idx_user_organization ON "user"(organization_id);
```