functions.sql

```sql
--1: Тех, кто не имеет воинских званий, нельзя отправлять на боевые миссии.
CREATE FUNCTION check_is_military_on_mission() RETURNS trigger AS $$
    DECLARE enemy TEXT;
    DECLARE emp_rank TEXT;
    BEGIN
        enemy = (SELECT enemies FROM mission WHERE miss_id = new.miss_id);
        emp_rank = (SELECT rank FROM position JOIN employee USING (pos_id) WHERE emp_id = new.emp_id);
        IF enemy IS NOT NULL AND emp_rank IS NULL THEN
            RAISE EXCEPTION 'Cannot set not military employee to a combat mission';
        ELSE RETURN new;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_is_military_on_mission BEFORE INSERT ON missions_emp
    FOR EACH ROW EXECUTE PROCEDURE check_is_military_on_mission();

/*
2: Информационная система должна учитывая какие сотрудники отправились на миссии (один и тот же сотрудник не
    может находиться на двух миссиях одновременно).
*/
CREATE FUNCTION check_periods_of_emp_missions() RETURNS trigger AS $$
    DECLARE start_time TIMESTAMP;
    DECLARE end_time TIMESTAMP;
    BEGIN
        SELECT start_time, end_time INTO start_time, end_time FROM mission WHERE miss_id = new.miss_id;
        IF (TRUE) IN (
            SELECT (start_time, end_time) OVERLAPS
                   (start_date_and_time, end_date_and_time) FROM mission
               WHERE miss_id IN (SELECT miss_id FROM missions_emp WHERE emp_id = new.emp_id)) THEN
            RAISE EXCEPTION 'This worker cannot be assigned to a mission as he was on another mission at the time';
        ELSE RETURN new;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_emp_mission_period BEFORE INSERT ON missions_emp
    FOR EACH ROW EXECUTE PROCEDURE check_periods_of_emp_missions();

CREATE INDEX mission_period ON mission USING btree(start_date_and_time, end_date_and_time);

--3: Работников неподходящих по физическим данным запрещено устраивать как военных сотрудников (рост < 150 см или вес < 45 кг).
CREATE FUNCTION check_physical_condition() RETURNS trigger AS $$
    DECLARE h SMALLINT;
    DECLARE w SMALLINT;
    BEGIN
        SELECT height_cm, weight_kg INTO h, w FROM medical_card JOIN employee USING (emp_id) WHERE emp_id = new.emp_id;
        IF h < 150 OR w < 45 THEN
            RAISE EXCEPTION 'Cannot hire this employee to military position because his physical data does not require the minimum';
        ELSE RETURN new;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_physical_condition BEFORE INSERT ON employee
    FOR EACH ROW EXECUTE PROCEDURE check_physical_condition();

/*
 4: Необходимо хранить историю инспекций транспорта (реализована отдельной таблицей),
    а транспорт со статусами «сломан» или «в ремонте» нельзя использовать в операциях.
 */
CREATE FUNCTION check_transport_condition() RETURNS trigger AS $$
    BEGIN
        IF (SELECT status FROM transport WHERE trans_id = new.trans_id AND status = 'available') IS NULL THEN
            RAISE EXCEPTION 'Cannot set not available transport to mission';
        ELSE RETURN new;
        END IF;
    END;
$$ LANGUAGE plpgsql;
```

functions.sql

```
70
71 CREATE TRIGGER check_transport_condition BEFORE INSERT ON missions_transport
72     FOR EACH ROW EXECUTE PROCEDURE check_transport_condition();
73
74 -- 5: Если за базой не закреплён ни один сотрудник, стоит закрыть её.
75 CREATE FUNCTION close_empty_bases() RETURNS SETOF void AS $$
76     BEGIN
77         DELETE FROM base WHERE base_id IN (SELECT * FROM base_count_emp);
78     END;
79 $$ LANGUAGE plpgsql;
80
81 CREATE MATERIALIZED VIEW base_count_emp AS
82     (SELECT base_id FROM base JOIN employee USING (base_id) GROUP BY base_id HAVING COUNT(emp_id) = 0);
83
84 CREATE FUNCTION update_base_count_emp() RETURNS trigger AS $$
85     BEGIN
86         REFRESH MATERIALIZED VIEW base_count_emp;
87         RETURN new;
88     END;
89 $$ LANGUAGE plpgsql;
90
91 CREATE TRIGGER update_base_count_emp AFTER INSERT OR UPDATE OR DELETE ON employee
92     FOR EACH ROW EXECUTE PROCEDURE update_base_count_emp();
93
94 /*
95 6: Стараться отправлять на боевые операции при прочих равных в первую очередь неженатых военных, давно
   не
96     участвовавших в миссиях, имеющих большой опыт работы.
97  */
98 CREATE FUNCTION get_combat_candidates(n int DEFAULT 1) RETURNS employee AS $$
99     BEGIN
100         SELECT emp_id FROM employee
101             JOIN position USING (pos_id)
102             JOIN missions_emp USING (emp_id)
103             JOIN mission USING (miss_id)
104         WHERE rank IS NOT NULL OR !~~ ''
105         ORDER BY is_married DESC, end_date_and_time DESC, hiring_date DESC
106         LIMIT n;
107     END;
108 $$ LANGUAGE plpgsql;
109
110 CREATE INDEX pos_rank ON position USING btree(rank);
111
112 /*
113 http://firststeps.ru/sql/oracle/r.php?43
114 https://postgrespro.ru/docs/postgresql/13/plpgsql-trigger
115 https://stackoverflow.com/questions/10335312/db-associative-entities-and-indexing
116 https://stackoverflow.com/questions/6015175/difference-between-view-and-table-in-sql
117 https://postgrespro.ru/docs/postgrespro/9.5/rules-materializedviews
118 */
```

scheme.sql

```sql
-- https://stackoverflow.com/questions/7296846/how-to-implement-one-to-one-one-to-many-and-many-to-many-relationships-while-de
CREATE TABLE base
(
    base_id  SERIAL PRIMARY KEY,
    location TEXT NOT NULL,
    status   TEXT NOT NULL
);

CREATE TABLE mre
(
    mre_id         SERIAL PRIMARY KEY,
    breakfast      TEXT     NOT NULL,
    lunch          TEXT     NOT NULL,
    dinner         TEXT     NOT NULL,
    food_additives TEXT,
    kkal           SMALLINT NOT NULL CHECK (kkal >= 3000),
    proteins       SMALLINT NOT NULL CHECK (proteins > 0),
    fats           SMALLINT NOT NULL CHECK (fats > 0),
    carbohydrate   SMALLINT NOT NULL CHECK (carbohydrate > 0)
);

CREATE TABLE equipment
(
    equip_id      SERIAL PRIMARY KEY,
    camouflage    TEXT,
    communication TEXT,
    intelligence  TEXT,
    medical       TEXT,
    mre_id        INTEGER NOT NULL REFERENCES mre ON DELETE RESTRICT,
    extra         TEXT
);

CREATE TYPE force AS ENUM ('GF', 'NAVY', 'AF');

CREATE TABLE position
(
    pos_id   SERIAL PRIMARY KEY,
    name     TEXT           NOT NULL,
    salary   NUMERIC(11, 2) NOT NULL CHECK (salary >= 300),
    rank     TEXT,
    equip_id INTEGER        REFERENCES equipment ON DELETE SET NULL,
    forces   FORCE
);

CREATE TABLE employee
(
    emp_id        SERIAL PRIMARY KEY,
    name          TEXT    NOT NULL,
    surname       TEXT    NOT NULL,
    date_of_birth DATE    NOT NULL CHECK (DATE_PART('year', AGE(date_of_birth)) >= 18),
    education     TEXT,
    hiring_date   DATE NOT NULL DEFAULT CURRENT_DATE,
    pos_id        INTEGER NOT NULL REFERENCES position ON DELETE RESTRICT,
    is_married    BOOLEAN NOT NULL,
    base_id       INTEGER REFERENCES base ON DELETE SET NULL
);

CREATE TABLE medical_card
(
    med_id    SERIAL PRIMARY KEY,
    emp_id    INTEGER  NOT NULL REFERENCES employee ON DELETE CASCADE,
    height_cm SMALLINT NOT NULL,
    weight_kg SMALLINT NOT NULL,
    diseases  TEXT,
    blood     TEXT     NOT NULL,
    gender    BOOLEAN  NOT NULL
);

CREATE TABLE weapon
(
    weapon_id        SERIAL PRIMARY KEY,
    name             TEXT NOT NULL,
    type             TEXT NOT NULL,
    caliber          REAL CHECK (caliber > 0),
```

scheme.sql

```sql
    rate_of_fire     SMALLINT CHECK (rate_of_fire > 0),
    sighting_range_m SMALLINT CHECK (sighting_range_m > 0)
);

CREATE TABLE campaign
(
    camp_id          SERIAL PRIMARY KEY,
    name             TEXT           NOT NULL,
    customer         TEXT           NOT NULL,
    earning          NUMERIC(11, 2) NOT NULL CHECK (earning >= 0),
    spending         NUMERIC(11, 2) NOT NULL CHECK (spending >= 0),
    execution_status TEXT
);

CREATE TABLE mission
(
    miss_id             SERIAL PRIMARY KEY,
    camp_id             INTEGER NOT NULL REFERENCES campaign ON DELETE CASCADE,
    start_date_and_time TIMESTAMP,
    end_date_and_time   TIMESTAMP,
    legal_status        BOOLEAN NOT NULL,
    departure_location  TEXT,
    arrival_location    TEXT,
    enemies             TEXT
);

CREATE TABLE transport
(
    trans_id SERIAL PRIMARY KEY,
    name     TEXT NOT NULL,
    type     TEXT NOT NULL,
    status   TEXT NOT NULL
);

CREATE TABLE equip_weapon
(
    equip_id  INTEGER NOT NULL REFERENCES equipment,
    weapon_id INTEGER NOT NULL REFERENCES weapon
);

CREATE TABLE missions_transport
(
    miss_id  INTEGER NOT NULL REFERENCES mission,
    trans_id INTEGER NOT NULL REFERENCES transport
);

CREATE TABLE inspection
(
    emp_id       INTEGER NOT NULL REFERENCES employee,
    trans_id     INTEGER NOT NULL REFERENCES transport,
    service_date DATE    NOT NULL DEFAULT CURRENT_DATE
);

CREATE TABLE missions_emp
(
    miss_id INTEGER NOT NULL REFERENCES mission,
    emp_id  INTEGER NOT NULL REFERENCES employee
);
```