НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

# ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**БАЗЫ ДАННЫХ**

# **Информационная Система Военного округа**

Сироткин Михаил Александрович

Группа 21203

Новосибирск 2024

## Содержание

[Содержание 2](#_Toc168349207)

[Задание 3](#_Toc168349208)

[Схема базы данных 4](#_Toc168349209)

[Создание таблиц 8](#_Toc168349210)

[Ограничения по поддержанию целостности 21](#_Toc168349211)

[Запросы к базе данных 22](#_Toc168349212)

[Триггеры и хранимые процедуры 49](#_Toc168349213)

[Создание клиентского приложения 57](#_Toc168349214)

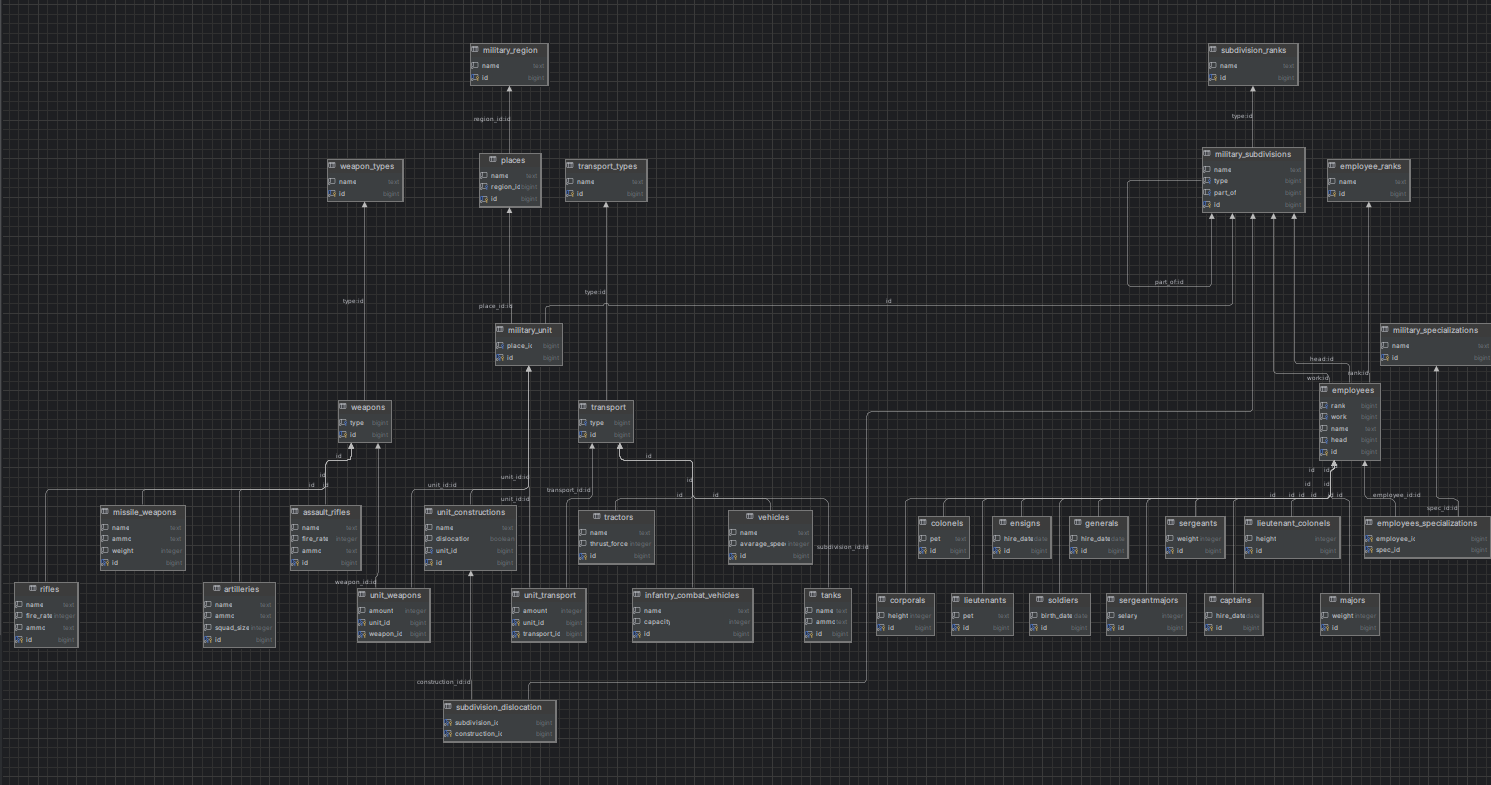
## Задание

Разработать структуру базы данных для информационной системы аптеки и реализовать приложение в архитектуре клиент-сервер, выполняющее операции внесения данных в базу данных, редактирование данных и запросы.

Военные части округа расквартированы по различным местам дислокации, причем в одном месте  могут располагаться несколько частей. Каждая воинская часть состоит из рот, роты из взводов,  взводы из отделений, в свою очередь воинские части объединяются в дивизии, корпуса или  бригады, а те в армии. Военный округ представлен офицерским составом (генералы, полковники,  подполковники, майоры, капитаны, лейтенанты) и рядовым и сержантским составом (старшины,  сержанты, прапорщики, ефрейторы, рядовые). Каждая из перечисленных категорий  военнослужащих может иметь характеристики, присущие только этой категории: для генералов  это может быть дата окончания академии, дата присвоения генеральского звания и т.д. Каждое из  подразделений имеет командира, причем военнослужащие офицерского состава могут  командовать любым из вышеперечисленных подразделений, а военнослужащие рядового и  сержантского состава только взводом и отделением. Все военнослужащие имеют одну или  несколько воинских специальностей.

Каждой воинской части придана боевая и транспортная техника: БМП, тягачи, автотранспорт и пр.  и вооружение: карабины, автоматическое оружие, артиллерия, ракетное вооружение и т.д. Каждая  из перечисленных категорий боевой техники и вооружения также имеет специфические, присущие  только ей атрибуты и по каждой категории может быть несколько видов техники и вооружения.  Инфраструктура военной части представлена набором сооружений (сооружение ©1, сооружение  ©2 . . .), некоторые из которых предназначены для дислокации подразделений части.

## Схема базы данных



* **places –** места дислокации военных частей (идентификатор места, название места, идентификатор округа из **military\_region**).
* **military\_subdivisions** – военные подразделения (идентификатор подразделения, название подразделения, идентификатор типа подразделения из таблицы **subdivision\_ranks**, идентификатор подразделения в котором содержится это подразделение).
* **subdivision\_ranks –** типы военных подразделений (идентификатор типа, название подразделения).
* **employees –** военнослужащие (идентификатор военнослужащего, идентификатор звания военнослужащего из **employee\_ranks**, идентификатор подразделения, где работает военнослужащий, из таблицы **military\_subdivisions**, ФИО военнослужащего, идентификатор подразделения, начальником которого является данный работник из таблицы **military\_subdivisions**).
* **employee\_ranks –** звания военнослужащих (идентификатор звания, название звания).
* **generals –** специальная информация о генералах (идентификатор генерала из таблицы **employees,** дата приёма на работу генерала).
* **colonels –** специальная информация о полковниках (идентификатор полковника из таблицы **employees,** животное, которым владеет полковник).
* **lieutenant\_colonels –** специальная информация о подполковниках (идентификатор подполковника из таблицы **employees,** рост подполковника).
* **majors –** специальная информация о майорах (идентификатор майора из таблицы **employees,** вес майора).
* **captains –** специальная информация о капитанах (идентификатор капитана из таблицы **employees,** дата приёма на работу капитана).
* **lieutenants –** специальная информация о лейтенантах (идентификатор лейтенанта из таблицы **employees,** животное, которым владеет лейтенант).
* **sergeantmajors –** специальная информация о старшинах (идентификатор старшины из таблицы **employees,** зарплата старшины).
* **sergeants –** специальная информация о сержантах (идентификатор сержанта из таблицы **employees,** вес сержанта).
* **ensigns –** специальная информация о прапорщиках (идентификатор ефрейтора из таблицы **employees,** дата приёма на работу прапорщика).
* **corporals –** специальная информация о ефрейторах (идентификатор ефрейтора из таблицы **employees,** рост ефрейтора).
* **soldiers –** специальная информация о солдатах (идентификатор солдата из таблицы **employees,** дата рождения солдата).
* **military\_specializations –** военные специализации (идентификатор специализации, название специализации).
* **employees\_specializations –** связь специализации и военнослужащего (идентификатор военнослужащего из таблицы **employees,** идентификатор специальности из таблицы **military\_specializations**).
* **military\_unit –** связь между военным подразделением (военной части) и местом её расположения (идентификатор военной части из таблицы **military\_subdivisions**, идентификатор из таблицы **places**).
* **military\_region –** военные круга (идентификатор военного округа, название военного округа).
* **transport\_types –** типы транспортных средств (идентификатор типа транспортного средства, название типа транспортного средства).
* **infantry\_combat\_vehicles –** специальная информация о БМП (идентификатор из таблицы **transport**, вместимость человек).
* **tanks –** специальная информация о танках (идентификатор из таблицы **transport**, тип боеприпасов, которыми стреляет танк).
* **tractors –** специальная информация о тягочах (идентификатор из таблицы **transport**, сила тяги)
* **vehicles –** специальная информация о автомобилях (идентификатор из таблицы **transport**, название автомобиля, средняя скорость).
* **transport –** связь между транспортом и его идентификатором (идентификатор транспорта, тип транспорта из таблицы **transport\_types,** название транспортного средства).
* **unit\_transport –** информация о том, какой транспорт и сколько его единиц состоит на счету воинской части (идентификатор воинской части из таблички **military\_subdivisions**, идентификатор транспортного средства из таблицы **transport,** количество транспортных средств).
* **weapon\_types –** типы оружия (идентификатор типа оружия, название типа оружия).
* **rifles –** специальная информация о оружии типа винтовка (идентификатор винтовки из таблицы **weapons**, количество выстрелов в минуту, тип боеприпасов).
* **assault\_rifles -** специальная информация о оружии типа автомат (идентификатор автомата из таблицы **weapons**, количество выстрелов в минуту, тип боеприпасов).
* **artilleries -** специальная информация о оружии типа артиллерия (идентификатор артиллерии из таблицы **weapons**, тип боеприпасов, размер группы для управления артиллерией).
* **missile\_weapons -** специальная информация о оружии типа ракетное вооружение (идентификатор ракетного вооружения из таблицы **weapons**, тип боеприпасов, вес ракетного вооружения).
* **weapons –** связь между оружием и его идентификатором (идентификатор оружия, идентификатор типа оружия из таблицы **weapon\_types**, название вооружения).
* **unit\_weapons –** информация о том, какое оружие и сколько его единиц состоит на счету воинской части (идентификатор воинской части из таблицы **military\_subdivisions**, идентификатор оружия из таблицы **weapons,** количество оружия).
* **unit\_constructions –** сооружения состоящие на счету у воинской части (идентификатор сооружения, название сооружения, можно ли в сооружении дислоцировать подразделение, идентификатор воинской части из таблицы **military\_subdivisions**).
* **subdivision\_dislocation –** связь между воинскими подразделениями и сооружениями военной части (идентификатор воинского подразделения из таблицы **military\_subdivisions**, идентификатор сооружения из таблицы **unit\_constructions**).

## Создание таблиц

CREATE TABLE IF NOT EXISTS "places" (

"id" bigserial,

"name" text NOT NULL,

"region\_id" bigint NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "military\_subdivisions" (

"id" bigserial,

"name" text NOT NULL,

"type" bigint NOT NULL,

"part\_of" bigint ,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "subdivision\_ranks" (

"id" bigserial,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

INSERT INTO "subdivision\_ranks" ("name")

VALUES('army'),

('brigade'),

('corp'),

('division'),

('unit'),

('company'),

('platoon'),

('squad');

CREATE TABLE IF NOT EXISTS "employees" (

"id" bigserial,

"rank" bigint NOT NULL,

"work" bigint NOT NULL,

"name" text NOT NULL,

"head" bigint,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "employee\_ranks" (

"id" bigserial,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

INSERT INTO "employee\_ranks"("name")

VALUES('general'),

('colonel'),

('lieutenant colonel'),

('major'),

('captain'),

('lieutenant'),

('sergeantmajor'),

('sergeant'),

('ensign'),

('corporal'),

('soldier');

CREATE TABLE IF NOT EXISTS "generals" (

"id" bigint NOT NULL,

"hire\_date" date NOT NULL CHECK("hire\_date" <= CURRENT\_DATE),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "colonels" (

"id" bigint NOT NULL,

"pet" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "lieutenant\_colonels" (

"id" bigint NOT NULL,

"height" integer NOT NULL CHECK("height" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "majors" (

"id" bigint NOT NULL,

"weight" integer NOT NULL CHECK("weight" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "captains" (

"id" bigint NOT NULL,

"hire\_date" date NOT NULL CHECK("hire\_date" <= CURRENT\_DATE),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "lieutenants" (

"id" bigint NOT NULL,

"pet" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "sergeantmajors" (

"id" bigint NOT NULL,

"selary" integer NOT NULL CHECK("selary" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "sergeants" (

"id" bigint NOT NULL,

"weight" integer NOT NULL CHECK("weight" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "ensigns"(

"id" bigint NOT NULL,

"hire\_date" date NOT NULL CHECK("hire\_date" <= CURRENT\_DATE),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "corporals" (

"id" bigint NOT NULL,

"height" integer NOT NULL CHECK(height > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "soldiers" (

"id" bigint NOT NULL,

"birth\_date" date NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "military\_specializations" (

"id" bigserial,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "employees\_specializations" (

"employee\_id" bigint NOT NULL,

"spec\_id" bigint NOT NULL,

PRIMARY KEY("employee\_id", "spec\_id")

);

CREATE TABLE IF NOT EXISTS "military\_unit" (

"id" bigint NOT NULL UNIQUE,

"place\_id" bigint NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "military\_region" (

"name" text NOT NULL,

"id" bigserial,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "transport\_types"(

"id" bigserial,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

INSERT INTO "transport\_types"("name")

VALUES('infantry combat vehicle'),

('tank'),

('tractor'),

('vehicle');

CREATE TABLE IF NOT EXISTS "infantry\_combat\_vehicles"(

"id" bigint NOT NULL,

"capacity" integer NOT NULL CHECK("capacity" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "tanks"(

"id" bigint NOT NULL,

"ammo" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "tractors"(

"id" bigint NOT NULL,

"thrust\_force" integer NOT NULL CHECK("thrust\_force" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "vehicles"(

"id" bigint NOT NULL,

"avarage\_speed" integer NOT NULL CHECK("avarage\_speed" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "transport"(

"id" bigserial,

"type" bigint NOT NULL,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "unit\_transport"(

"id" bigserial NOT NULL,

"unit\_id" bigint NOT NULL,

"transport\_id" bigint NOT NULL,

"amount" integer NOT NULL CHECK("amount" > 0),

PRIMARY KEY("id")

);

CREATE TABLE IF NOT EXISTS "weapon\_types"(

"id" bigserial,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

INSERT INTO "weapon\_types"("name")

VALUES('rifle'),

('assault rifle'),

('artillery'),

('missile weapons');

CREATE TABLE IF NOT EXISTS "rifles"(

"id" bigint NOT NULL,

"fire\_rate" integer NOT NULL CHECK("fire\_rate" > 0),

"ammo" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "assault\_rifles"(

"id" bigint NOT NULL,

"fire\_rate" integer NOT NULL CHECK("fire\_rate" > 0),

"ammo" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "artilleries"(

"id" bigint NOT NULL,

"ammo" text NOT NULL,

"squad\_size" integer NOT NULL CHECK ("squad\_size" > 0),

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "missile\_weapons"(

"id" bigint NOT NULL,

"ammo" text NOT NULL,

"weight" integer NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "weapons"(

"id" bigserial,

"type" bigint NOT NULL,

"name" text NOT NULL,

PRIMARY KEY ("id")

);

CREATE TABLE IF NOT EXISTS "unit\_weapons"(

"id" bigserial NOT NULL,

"unit\_id" bigint NOT NULL,

"weapon\_id" bigint NOT NULL,

"amount" integer NOT NULL CHECK("amount" > 0),

PRIMARY KEY("id")

);

CREATE TABLE IF NOT EXISTS "unit\_constructions"(

"id" bigserial,

"name" text NOT NULL,

"dislocation" boolean NOT NULL,

"unit\_id" bigint NOT NULL,

PRIMARY KEY("id")

);

CREATE TABLE IF NOT EXISTS "subdivision\_dislocation"(

"subdivision\_id" bigint NOT NULL,

"construction\_id" bigint NOT NULL,

PRIMARY KEY ("subdivision\_id", "construction\_id")

);

ALTER TABLE "unit\_constructions" ADD CONSTRAINT "unit\_constructions\_fk1" FOREIGN KEY ("unit\_id") REFERENCES "military\_unit"("id") ON DELETE CASCADE;

ALTER TABLE "subdivision\_dislocation" ADD CONSTRAINT "subdivision\_dislocation\_fk1" FOREIGN KEY ("construction\_id") REFERENCES "unit\_constructions"("id") ON DELETE CASCADE;

ALTER TABLE "subdivision\_dislocation" ADD CONSTRAINT "subdivision\_dislocation\_fk2" FOREIGN KEY ("subdivision\_id") REFERENCES "military\_subdivisions"("id") ON DELETE CASCADE;

ALTER TABLE "unit\_weapons" ADD CONSTRAINT "unit\_weapon\_fk1" FOREIGN KEY ("weapon\_id") REFERENCES "weapons"("id") ON DELETE CASCADE;

ALTER TABLE "unit\_weapons" ADD CONSTRAINT "unit\_weapon\_fk2" FOREIGN KEY ("unit\_id") REFERENCES "military\_unit"("id") ON DELETE CASCADE;

ALTER TABLE "weapons" ADD CONSTRAINT "weapon\_fk1" FOREIGN KEY ("type") REFERENCES "weapon\_types"("id") ON DELETE CASCADE;

ALTER TABLE "unit\_transport" ADD CONSTRAINT "unit\_transport\_fk1" FOREIGN KEY ("transport\_id") REFERENCES "transport"("id") ON DELETE CASCADE;

ALTER TABLE "unit\_transport" ADD CONSTRAINT "unit\_transport\_fk2" FOREIGN KEY ("unit\_id") REFERENCES "military\_unit"("id") ON DELETE CASCADE;

ALTER TABLE "transport" ADD CONSTRAINT "transport\_fk1" FOREIGN KEY ("type") REFERENCES "transport\_types"("id") ON DELETE CASCADE;

ALTER TABLE "military\_subdivisions" ADD CONSTRAINT "military\_subdivisions\_fk2" FOREIGN KEY ("type") REFERENCES "subdivision\_ranks"("id") ON DELETE CASCADE;

ALTER TABLE "military\_subdivisions" ADD CONSTRAINT "military\_subdivisions\_fk3" FOREIGN KEY ("part\_of") REFERENCES "military\_subdivisions"("id") ON DELETE SET NULL;

ALTER TABLE "employees" ADD CONSTRAINT "employees\_fk1" FOREIGN KEY ("rank") REFERENCES "employee\_ranks"("id") ON DELETE CASCADE;

ALTER TABLE "employees" ADD CONSTRAINT "employees\_fk2" FOREIGN KEY ("work") REFERENCES "military\_subdivisions"("id") ON DELETE CASCADE;

ALTER TABLE "employees" ADD CONSTRAINT "employees\_fk3" FOREIGN KEY ("head") REFERENCES "military\_subdivisions"("id") ON DELETE SET NULL;

ALTER TABLE "generals" ADD CONSTRAINT "generals\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "colonels" ADD CONSTRAINT "colonels\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "lieutenant\_colonels" ADD CONSTRAINT "lieutenant colonels\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "majors" ADD CONSTRAINT "majors\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "captains" ADD CONSTRAINT "captains\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "lieutenants" ADD CONSTRAINT "lieutenants\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "sergeantmajors" ADD CONSTRAINT "sergeantmajors\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "sergeants" ADD CONSTRAINT "sergeants\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "corporals" ADD CONSTRAINT "corporals\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "soldiers" ADD CONSTRAINT "soldiers\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "ensigns" ADD CONSTRAINT "ensigns\_fk0" FOREIGN KEY ("id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "infantry\_combat\_vehicles" ADD CONSTRAINT "infantry combat vehicles\_fk0" FOREIGN KEY ("id") REFERENCES "transport"("id") ON DELETE CASCADE;

ALTER TABLE "tanks" ADD CONSTRAINT "tanks\_fk0" FOREIGN KEY ("id") REFERENCES "transport"("id") ON DELETE CASCADE;

ALTER TABLE "tractors" ADD CONSTRAINT "tractors\_fk0" FOREIGN KEY ("id") REFERENCES "transport"("id") ON DELETE CASCADE;

ALTER TABLE "vehicles" ADD CONSTRAINT "vehicles\_fk0" FOREIGN KEY ("id") REFERENCES "transport"("id") ON DELETE CASCADE;

ALTER TABLE "rifles" ADD CONSTRAINT "rifles\_fk0" FOREIGN KEY ("id") REFERENCES "weapons"("id") ON DELETE CASCADE;

ALTER TABLE "assault\_rifles" ADD CONSTRAINT "assault rifles\_fk0" FOREIGN KEY ("id") REFERENCES "weapons"("id") ON DELETE CASCADE;

ALTER TABLE "artilleries" ADD CONSTRAINT "artilleries\_fk0" FOREIGN KEY ("id") REFERENCES "weapons"("id") ON DELETE CASCADE;

ALTER TABLE "missile\_weapons" ADD CONSTRAINT "missile\_weapons\_fk0" FOREIGN KEY ("id") REFERENCES "weapons"("id") ON DELETE CASCADE;

ALTER TABLE "employees\_specializations" ADD CONSTRAINT "employees\_specializations\_fk0" FOREIGN KEY ("employee\_id") REFERENCES "employees"("id") ON DELETE CASCADE;

ALTER TABLE "employees\_specializations" ADD CONSTRAINT "employees\_specializations\_fk1" FOREIGN KEY ("spec\_id") REFERENCES "military\_specializations"("id") ON DELETE CASCADE;

ALTER TABLE "military\_unit" ADD CONSTRAINT "military\_unit\_fk0" FOREIGN KEY ("id") REFERENCES "military\_subdivisions"("id") ON DELETE CASCADE;

ALTER TABLE "military\_unit" ADD CONSTRAINT "military\_unit\_fk1" FOREIGN KEY ("place\_id") REFERENCES "places"("id") ON DELETE CASCADE;

ALTER TABLE "places" ADD CONSTRAINT "places\_fk0" FOREIGN KEY ("region\_id") REFERENCES "military\_region"("id") ON DELETE CASCADE;

## Ограничения по поддержанию целостности

При добавлении нового воинского подразделения нужно учитывать, что у воинских подразделений есть чёткая иерархия, описанная в задании.

При добавлении нового военнослужащего нужно учитывать, что он не может управлять всеми воинскими подразделениями в зависимости от званий, это описано в задании.

Так же нужно поддерживать правильную связь идентификаторов в таблице **employees** и в таблицах для каждого воинского звания.

Аналогичная ситуация и с таблицей **transport** и **weapons**.

При добавлении нового воинского звания, типа транспорта и типа вооружения нужно создать новую таблицу для соответствующей разновидности.

Для таблицы **subdivision\_dislocation** нужно учитывать возможность разместиться в соответствующем сооружении.

## Запросы к базе данных

1. **Получить перечень всех частей военного округа, указанной армии, дивизии, корпуса и их командиров.**

Парметр: $region\_id$

WITH region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$)

SELECT ms1.name AS subdivision\_name, employees.name AS head\_name, employee\_ranks.name AS rank

FROM region\_units INNER JOIN military\_subdivisions ms1 ON region\_units.id = ms1.id

INNER JOIN military\_subdivisions ms2 ON ms1.part\_of = ms2.id

INNER JOIN military\_subdivisions ms3 ON ms2.part\_of = ms3.id

LEFT JOIN employees ON employees.head = region\_units.id

INNER JOIN employee\_ranks ON employee\_ranks.id = employees.rank;

Параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$ ),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of)

SELECT ms1.name AS subdivision\_name, employees.name AS head\_name, employee\_ranks.name AS rank

FROM specific\_units INNER JOIN military\_subdivisions ms1 ON specific\_units.id = ms1.id

LEFT JOIN employees ON employees.head = specific\_units.id

INNER JOIN employee\_ranks ON employee\_ranks.id = employees.rank;

параметр: $under\_army\_id$

WITH

specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of)

SELECT ms1.name AS subdivision\_name, employees.name AS head\_name, employee\_ranks.name AS rank

FROM specific\_units INNER JOIN military\_subdivisions ms1 ON specific\_units.id = ms1.id

LEFT JOIN employees ON employees.head = specific\_units.id

INNER JOIN employee\_ranks ON employee\_ranks.id = employees.rank;

2. **Получить данные по офицерскому составу в целом и по офицерскому составу указанного звания всех частей военного округа, отдельной армии, дивизии, корпуса, военной части.**

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank < 7;

параметр: $region\_id$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

specific\_companies as (SELECT ms.id as id FROM region\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = region\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from region\_units UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank < 7;

параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from specific\_army UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

параметр: $under\_army\_id$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank < 7;

параметр: $unit\_id$

WITH specific\_units as (SELECT ms.id FROM military\_subdivisions ms WHERE id = $unit\_id$),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank < 7;

3. **Получить данные по рядовому и сержантскому составу в целом и с учетом указанного звания всех частей военного округа, отдельной армии, дивизии, корпуса, военной части.**

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank >= 7;

параметр: $region\_id$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

specific\_companies as (SELECT ms.id as id FROM region\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = region\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from region\_units UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank >= 7;

параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from specific\_army UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank >= 7;

параметр: $under\_army\_id$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank >= 7;

параметр: $unit\_id$

WITH specific\_units as (SELECT ms.id FROM military\_subdivisions ms WHERE id = $unit\_id$),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_units)

SELECT employees.name as "name", employee\_ranks.name as "rank", ms.name as "work\_id", ms1.name as "head\_of" FROM employees

INNER JOIN subdivisions ON subdivisions.id = employees.work

INNER JOIN employee\_ranks ON employees.rank = employee\_ranks.id

LEFT JOIN public.military\_subdivisions ms on ms.id = employees.work

LEFT JOIN public.military\_subdivisions ms1 on ms1.id = employees.head

WHERE employees.rank >= 7;

4. **Получить цепочку подчиненности снизу доверху для указанного военнослужащего.**

Параметр: $employee\_id$

WITH RECURSIVE r AS (

SELECT

id, head FROM employees WHERE id = $employee\_id$

UNION

SELECT

employees.id as id, employees.head FROM employees LEFT JOIN military\_subdivisions ms on employees.head = ms.id INNER JOIN r on part\_of = r.head

)

SELECT employees.name as name, military\_subdivisions.name as work, employee\_ranks.name as rank

FROM r INNER JOIN military\_subdivisions on military\_subdivisions.id = r.head

INNER JOIN employees on employees.work = military\_subdivisions.id

INNER JOIN employee\_ranks on employees.rank = employee\_ranks.id;

5. **Получить перечень мест дислокации всех частей военного округа, отдельной армии, дивизии, корпуса, военной части.**

Параметр: $region\_id$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

specific\_companies as (SELECT ms.id as id FROM region\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = region\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT DISTINCT unit\_constructions.name FROM subdivision\_dislocation

INNER JOIN subdivisions on subdivision\_id = subdivisions.id

INNER JOIN unit\_constructions on subdivision\_dislocation.subdivision\_id = unit\_constructions.id;

параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$ ),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT DISTINCT unit\_constructions.name FROM subdivision\_dislocation

INNER JOIN subdivisions on subdivision\_id = subdivisions.id

INNER JOIN unit\_constructions on subdivision\_dislocation.subdivision\_id = unit\_constructions.id;

параметр: $under\_army\_id$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_units)

SELECT DISTINCT unit\_constructions.name FROM subdivision\_dislocation

INNER JOIN subdivisions on subdivision\_id = subdivisions.id

INNER JOIN unit\_constructions on subdivision\_dislocation.subdivision\_id = unit\_constructions.id;

параметр: $unit\_id$

WITH specific\_units as (SELECT ms.id FROM military\_subdivisions ms WHERE id = $unit\_id$),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT DISTINCT unit\_constructions.name FROM subdivision\_dislocation

INNER JOIN subdivisions on subdivision\_id = subdivisions.id

INNER JOIN unit\_constructions on subdivision\_dislocation.subdivision\_id = unit\_constructions.id;

6. **Получить данные о наличии боевой технике в целом и с учетом указанной категории или вида во всех частях военного округа, в отдельной армии, дивизии, корпусе, военной части.**

WITH t as (SELECT transport\_id, sum(unit\_transport.amount) as total\_amout FROM unit\_transport

INNER JOIN transport on unit\_transport.transport\_id = transport.id

GROUP BY transport\_id)

SELECT transport.name as name, t.total\_amout as total\_amout FROM t INNER JOIN transport on transport.id = t.transport\_i;

Параметры: $region\_id$, $transport\_type$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

t as (SELECT transport\_id, sum(unit\_transport.amount) FROM unit\_transport

INNER JOIN region\_units on region\_units.id = unit\_transport.unit\_id

INNER JOIN transport on unit\_transport.transport\_id = transport.id

GROUP BY transport\_id)

SELECT transport.name as transport, t.sum as total\_amout, transport\_types.name as type FROM t INNER JOIN transport on t.transport\_id = transport.id

INNER JOIN transport\_types on transport.type = transport\_types.id

WHERE transport.type = $transport\_type$ ;

Параметры: $army\_id$, $transport\_type$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$ ),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

t as (SELECT transport\_id, sum(unit\_transport.amount) FROM unit\_transport

INNER JOIN specific\_units on specific\_units.id = unit\_transport.unit\_id

INNER JOIN transport on unit\_transport.transport\_id = transport.id

GROUP BY transport\_id)

SELECT transport.name as transport, t.sum as total\_amout, transport\_types.name as type FROM t INNER JOIN transport on t.transport\_id = transport.id

INNER JOIN transport\_types on transport.type = transport\_types.id

WHERE transport.type = $transport\_type$ ;

Параметры: $under\_army\_id$, $transport\_type$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

t as (SELECT transport\_id, sum(unit\_transport.amount) FROM unit\_transport

INNER JOIN specific\_units on specific\_units.id = unit\_transport.unit\_id

INNER JOIN transport on unit\_transport.transport\_id = transport.id

GROUP BY transport\_id)

SELECT transport.name as transport, t.sum as total\_amout, transport\_types.name as type FROM t INNER JOIN transport on t.transport\_id = transport.id

INNER JOIN transport\_types on transport.type = transport\_types.id

WHERE transport.type = $transport\_type$;

Параметры: $unit\_id$, $transport\_type$

WITH t as (SELECT transport\_id, sum(unit\_transport.amount) FROM unit\_transport

INNER JOIN transport on unit\_transport.transport\_id = transport.id

WHERE unit\_id =$unit\_id$

GROUP BY transport\_id)

SELECT transport.name as transport, t.sum as total\_amout, transport\_types.name as type FROM t INNER JOIN transport on t.transport\_id = transport.id

INNER JOIN transport\_types on transport.type = transport\_types.id

WHERE transport.type = $transport\_type$ ;

**7. Получить перечень сооружений указанной военной части, перечень сооружений, где дислоцировано более одного подразделения, где недислоцировано ни одного подразделения.**

WITH c as (SELECT unit\_constructions.id, count(sd.construction\_id) FROM unit\_constructions LEFT JOIN public.subdivision\_dislocation sd on unit\_constructions.id = sd.construction\_id

GROUP BY unit\_constructions.id)

SELECT name, c.count as number\_of\_dislocations from c JOIN unit\_constructions on c.id = unit\_constructions.id

WHERE c.count > 1;

-- менее одного

WITH c as (SELECT unit\_constructions.id, count(sd.construction\_id) FROM unit\_constructions LEFT JOIN public.subdivision\_dislocation sd on unit\_constructions.id = sd.construction\_id

GROUP BY unit\_constructions.id)

SELECT name FROM c INNER JOIN unit\_constructions on c.id = unit\_constructions.id

WHERE c.count = 0;

**8. Получить перечень военных частей, в которых число единиц указанного вида боевой техники больше 5 (нет указанной боевой техники).**

WITH t as (SELECT t.type, unit\_id, sum(unit\_transport.amount) as total\_amout from unit\_transport

RIGHT JOIN transport t on t.id = unit\_transport.transport\_id

RIGHT JOIN military\_unit on unit\_transport.unit\_id = military\_unit.id

group by t.type, unit\_id

HAVING sum(unit\_transport.amount) > 5)

SELECT wt.name as type, ms.name as unit, total\_amout FROM t INNER JOIN weapon\_types wt on wt.id = t.type INNER JOIN military\_subdivisions ms on t.unit\_id = ms.id;

WITH t as (SELECT t.type, unit\_id, sum(unit\_transport.amount) as total\_amout from unit\_transport

RIGHT JOIN transport t on t.id = unit\_transport.transport\_id

RIGHT JOIN military\_unit on unit\_transport.unit\_id = military\_unit.id

group by t.type, unit\_id

HAVING sum(unit\_transport.amount) = 0)

SELECT wt.name as type, ms.name as unit, total\_amout FROM t INNER JOIN weapon\_types wt on wt.id = t.type INNER JOIN military\_subdivisions ms on t.unit\_id = ms.id;

**9. Получить данные о наличии вооружения в целом и с учетом указанной категории или вида во всех частях военного округа, в отдельной армии, дивизии, корпусе, военной части.**

Параметры: $region\_id$, $weapon\_type$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

t as (SELECT weapon\_id, sum(unit\_weapons.amount) FROM unit\_weapons

INNER JOIN region\_units on region\_units.id = unit\_weapons.unit\_id

INNER JOIN weapons on unit\_weapons.weapon\_id = weapons.id

GROUP BY weapon\_id)

SELECT weapons.name as name, t.sum as total\_amout, weapon\_types.name as type FROM t INNER JOIN weapons on t.weapon\_id = weapons.id

INNER JOIN weapon\_types on weapons.type = weapon\_types.id WHERE weapon\_types.id = $weapon\_type$ ;

Параметры: $army\_id$, $weapon\_type$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$ ),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

t as (SELECT weapon\_id, sum(unit\_weapons.amount) FROM unit\_weapons

INNER JOIN specific\_units on specific\_units.id = unit\_weapons.unit\_id

INNER JOIN weapons on unit\_weapons.weapon\_id = weapons.id

GROUP BY weapon\_id)

SELECT weapons.name as name, t.sum as total\_amout, weapon\_types.name as type FROM t INNER JOIN weapons on t.weapon\_id = weapons.id

INNER JOIN weapon\_types on weapons.type = weapon\_types.id WHERE weapon\_types.id = $weapon\_type$ ;

Параметры: $under\_army\_id$, $weapon\_type$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

t as (SELECT weapon\_id, sum(unit\_weapons.amount) FROM unit\_weapons

INNER JOIN specific\_units on specific\_units.id = unit\_weapons.unit\_id

INNER JOIN weapons on unit\_weapons.weapon\_id = weapons.id

GROUP BY weapon\_id)

SELECT weapons.name as name, t.sum as total\_amout, weapon\_types.name as type FROM t INNER JOIN weapons on t.weapon\_id = weapons.id

INNER JOIN weapon\_types on weapons.type = weapon\_types.id WHERE weapon\_types.id = $weapon\_type$ ;

Параметры: $unit\_id$, $weapon\_type$

WITH t as (SELECT weapon\_id, sum(unit\_weapons.amount) FROM unit\_weapons

INNER JOIN weapons on unit\_weapons.weapon\_id = weapons.id

WHERE unit\_id = $unit\_id$

GROUP BY weapon\_id)

SELECT weapons.name as name, t.sum as total\_amout, weapon\_types.name as type FROM t INNER JOIN weapons on t.weapon\_id = weapons.id

INNER JOIN weapon\_types on weapons.type = weapon\_types.id WHERE weapon\_types.id = $weapon\_type$ ;

10. **Получить перечень военных специальностей, по которым в округе, в отдельной армии, дивизии, корпусе, военной части более пяти специалистов (нет специалистов).**

Параметр: $region\_id$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

specific\_companies as (SELECT ms.id as id FROM region\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = region\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from region\_units UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads),

t as (SELECT es.spec\_id, count(es.spec\_id) as total\_amount from subdivisions INNER JOIN employees on employees.work = subdivisions.id

RIGHT JOIN employees\_specializations es on employees.id = es.employee\_id

GROUP BY es.spec\_id

HAVING count(es.spec\_id) > 5)

SELECT ms.name as spec, t.total\_amount as total\_amount FROM t INNER JOIN military\_specializations ms on ms.id = t.spec\_id;

Параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from specific\_army UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units),

t as (SELECT es.spec\_id, count(es.spec\_id) as total\_amount from subdivisions INNER JOIN employees on employees.work = subdivisions.id

RIGHT JOIN employees\_specializations es on employees.id = es.employee\_id

GROUP BY es.spec\_id

HAVING count(es.spec\_id) > 5)

SELECT ms.name as spec, t.total\_amount as total\_amount FROM t INNER JOIN military\_specializations ms on ms.id = t.spec\_id;

Парметр: $under\_army\_id$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units),

t as (SELECT es.spec\_id, count(es.spec\_id) as total\_amount from subdivisions INNER JOIN employees on employees.work = subdivisions.id

RIGHT JOIN employees\_specializations es on employees.id = es.employee\_id

GROUP BY es.spec\_id

HAVING count(es.spec\_id) > 5)

SELECT ms.name as spec, t.total\_amount as total\_amount FROM t INNER JOIN military\_specializations ms on ms.id = t.spec\_id;

Параметр: $unit\_id$

WITH specific\_units as (SELECT ms.id FROM military\_subdivisions ms WHERE id = $unit\_id$),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_units),

t as (SELECT es.spec\_id, count(es.spec\_id) as total\_amount from subdivisions INNER JOIN employees on employees.work = subdivisions.id

RIGHT JOIN employees\_specializations es on employees.id = es.employee\_id

GROUP BY es.spec\_id

HAVING count(es.spec\_id) > 5)

SELECT ms.name as spec, t.total\_amount as total\_amount FROM t INNER JOIN military\_specializations ms on ms.id = t.spec\_id;

**11.** **Получить перечень военнослужащих указанной специальности в округе, в отдельной армии, дивизии, корпусе, военной части, в указанном подразделении некоторой военной части.**

Параметр: $region\_id$

WITH

region\_units as (SELECT military\_unit.id as id from military\_unit INNER JOIN places ON military\_unit.place\_id = places.id

WHERE places.region\_id = $region\_id$),

specific\_companies as (SELECT ms.id as id FROM region\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = region\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from region\_units UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads)

SELECT employees.name from subdivisions INNER JOIN employees on employees.work = subdivisions.id

INNER JOIN employees\_specializations es on employees.id = es.employee\_id

WHERE es.spec\_id = 2;

Параметр: $army\_id$

WITH

specific\_army as (SELECT id from military\_subdivisions WHERE id = $army\_id$ ),

specific\_under\_army as (SELECT ms.id FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id from specific\_army UNION SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name from subdivisions INNER JOIN employees on employees.work = subdivisions.id

INNER JOIN employees\_specializations es on employees.id = es.employee\_id

WHERE es.spec\_id = 2;

Параметр: $under\_army\_id$

WITH specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE ms.id = $under\_army\_id$),

specific\_units as (SELECT ms.id FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_under\_army UNION SELECT id FROM specific\_units)

SELECT employees.name from subdivisions INNER JOIN employees on employees.work = subdivisions.id

INNER JOIN employees\_specializations es on employees.id = es.employee\_id

WHERE es.spec\_id = 2;

Параметр: $unit\_id$, $spec\_id$

WITH specific\_units as (SELECT ms.id FROM military\_subdivisions ms WHERE id = $unit\_id$),

specific\_companies as (SELECT ms.id as id FROM specific\_units INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_units.id),

specific\_platoons as (SELECT ms.id as id FROM specific\_companies INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_companies.id),

specific\_squads as (SELECT ms.id as id FROM specific\_platoons INNER JOIN military\_subdivisions ms ON ms.part\_of = specific\_platoons.id),

subdivisions as (SELECT id FROM specific\_companies UNION SELECT id FROM specific\_platoons UNION SELECT id FROM specific\_squads

UNION SELECT id FROM specific\_units)

SELECT employees.name from subdivisions INNER JOIN employees on employees.work = subdivisions.id

INNER JOIN employees\_specializations es on employees.id = es.employee\_id

WHERE es.spec\_id = $spec\_id$;

**12. Получить перечень военных частей, в которых число единиц указанного вида вооружения больше 10 (нет указанного вооружения).**

Параметр: $weapon\_type$

SELECT military\_subdivisions.name, unit\_weapons.weapon\_id, unit\_weapons.amount from unit\_weapons RIGHT JOIN military\_unit on unit\_weapons.unit\_id = military\_unit.id

INNER JOIN weapons on unit\_weapons.weapon\_id = weapons.id

INNER JOIN military\_subdivisions on military\_unit.id = military\_subdivisions.id

WHERE weapons.type = $weapon\_type$ AND unit\_weapons.amount > 10;

**13. Получить данные об армии, дивизии, корпусе, в которые входит больше всего (меньше всего) военных частей.**WITH

specific\_army as (SELECT id from military\_subdivisions WHERE military\_subdivisions.type = 1),

specific\_under\_army as (SELECT ms.id, specific\_army.id as army FROM specific\_army INNER JOIN military\_subdivisions ms ON specific\_army.id = ms.part\_of),

specific\_units as (SELECT count(ms.id), army FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of

group by army),

army\_units as ( SELECT specific\_units.army as army, specific\_units.count as number\_of\_units FROM specific\_units INNER JOIN military\_subdivisions on military\_subdivisions.id = specific\_units.army)

SELECT name, number\_of\_units FROM army\_units INNER JOIN military\_subdivisions on army\_units.army = military\_subdivisions.id

WHERE army\_units.number\_of\_units = (SELECT min(number\_of\_units) from army\_units);

Параметр: $under\_army\_type$

WITH

specific\_under\_army as (SELECT ms.id FROM military\_subdivisions ms WHERE type = $under\_army\_type$),

specific\_units as (SELECT count(ms.id), specific\_under\_army.id as under\_army FROM specific\_under\_army INNER JOIN military\_subdivisions ms ON specific\_under\_army.id = ms.part\_of

group by specific\_under\_army.id),

army\_units as ( SELECT specific\_units.under\_army as under\_army, specific\_units.count as number\_of\_units FROM specific\_units INNER JOIN military\_subdivisions on military\_subdivisions.id = specific\_units.under\_army)

SELECT name, number\_of\_units FROM army\_units INNER JOIN military\_subdivisions on army\_units.under\_army = military\_subdivisions.id

WHERE army\_units.number\_of\_units = (SELECT min(number\_of\_units) from army\_units);

## Триггеры и хранимые процедуры

1. Триггеры на изменение в таблицах для разных рангов военнослужащих (generals, majors…). Данный триггер добавляется на UPDATE и INSERT в эти таблицы с параметром равным идентификатору званию, соответствующему таблице. Этот триггер проверят, что мы положили в данные таблицы военнослужащего соответствующего ранга.

CREATE OR REPLACE FUNCTION employees\_consistency() RETURNS TRIGGER AS

$$

DECLARE

rank\_check bigint;

real\_type bigint;

BEGIN

rank\_check := TG\_ARGV[0];

SELECT rank INTO real\_type from employees WHERE id = NEW.id;

IF NEW IS NOT NULL THEN

IF real\_type IS NULL OR real\_type != rank\_check THEN

RAISE EXCEPTION 'expected rank % for this table', (SELECT name FROM employee\_ranks WHERE id = rank\_check);

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггеры на изменение в таблицах для разных типов вооружения (rifles…). Данный триггер добавляется на UPDATE и INSERT в эти таблицы с параметром равным идентификатору типа, соответствующему таблице. Этот триггер проверят, что мы положили в данные таблицы оружие соответствующего типа.

CREATE OR REPLACE FUNCTION weapon\_consistency() RETURNS TRIGGER AS

$$

DECLARE

type\_check bigint;

real\_type bigint;

BEGIN

type\_check := TG\_ARGV[0];

SELECT type INTO real\_type from weapons WHERE id = NEW.id;

IF NEW IS NOT NULL THEN

IF real\_type IS NULL OR real\_type != type\_check THEN

RAISE EXCEPTION 'expected type % for this table', (SELECT name FROM weapon\_types WHERE id = type\_check);

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггеры на изменение в таблицах для разных типов транспорта (tanks…). Данный триггер добавляется на UPDATE и INSERT в эти таблицы с параметром равным идентификатору типа, соответствующему таблице. Этот триггер проверят, что мы положили в данные таблицы транспорт соответствующего типа.

CREATE OR REPLACE FUNCTION transport\_consistency() RETURNS TRIGGER AS

$$

DECLARE

type\_check bigint;

real\_type bigint;

BEGIN

type\_check := TG\_ARGV[0];

SELECT type INTO real\_type from transport WHERE id = NEW.id;

IF NEW IS NOT NULL THEN

IF real\_type IS NULL OR real\_type != type\_check THEN

RAISE EXCEPTION 'expected type % for this table', (SELECT name FROM transport\_types WHERE id = type\_check);

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггер на изменение в таблице **subdivision\_dislocation.** Данный триггер ставится на INSERT и UPDATE. Он проверяет, а можно ли замещать подразделения в этом сооружении.

CREATE OR REPLACE FUNCTION dislocation\_consistency() RETURNS TRIGGER AS

$$

DECLARE

can\_dislocate boolean;

BEGIN

SELECT dislocation INTO can\_dislocate from unit\_constructions WHERE id = NEW.construction\_id;

IF NEW IS NOT NULL THEN

IF can\_dislocate IS NULL OR NOT can\_dislocate THEN

RAISE EXCEPTION 'you cannot dislocate in this construction';

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггер на изменение в таблице **military\_subdivisions.** Данный триггер ставится на INSERT и UPDATE. Он проверяет, а правильно ли выстроена иерархия воинских подразделений.

CREATE OR REPLACE FUNCTION subdivision\_consistency() RETURNS TRIGGER AS

$$

DECLARE

part\_of\_type bigint;

BEGIN

IF NEW.part\_of IS NOT NULL THEN

SELECT type INTO part\_of\_type from military\_subdivisions WHERE id = NEW.part\_of;

IF (part\_of\_type = 1 AND (NEW.type = 2 OR NEW.type = 3 OR NEW.type = 4)) THEN

RETURN NEW;

end if;

IF ((part\_of\_type = 2 OR part\_of\_type = 3 OR part\_of\_type = 4) AND NEW.type = 5) THEN

RETURN NEW;

end if;

IF part\_of\_type + 1 != NEW.type THEN

RAISE EXCEPTION 'you cannot make % part of %.', (SELECT name from subdivision\_ranks WHERE id = NEW.type),

(SELECT name from subdivision\_ranks WHERE id = part\_of\_type);

end if;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггер на изменение в таблице **military\_unit.** Данный триггер ставится на INSERT и UPDATE. Он проверяет, что мы кладёт в эту таблицу именно unit из таблицы **military\_subdivisions.**

CREATE OR REPLACE FUNCTION unit\_consistency() RETURNS TRIGGER AS

$$

DECLARE

id\_military\_subdivisions\_type bigint;

BEGIN

SELECT type INTO id\_military\_subdivisions\_type FROM military\_subdivisions WHERE id = NEW.id;

if id\_military\_subdivisions\_type != 5 THEN

RAISE EXCEPTION 'you cannot add % to this table, because this only for units', (SELECT name FROM subdivision\_ranks WHERE id = NEW.id);

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Триггер на изменение в таблице **employees.** Данный триггер ставится на INSERT и UPDATE. Он проверяет, что военнослужащие рядового и сержантского состава могут руководить только **platoon** и **squad**.

CREATE OR REPLACE FUNCTION employee\_head\_consistency() RETURNS TRIGGER AS

$$

DECLARE

head\_type bigint;

BEGIN

IF NEW.head IS NOT NULL THEN

SELECT type INTO head\_type FROM military\_subdivisions WHERE id = NEW.head;

IF (NEW.rank > 6 AND head\_type > 7) THEN

RAISE EXCEPTION 'employees with ranks less than lieutenant can manage only squads and platoons';

end if;

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Для поддержания целостности были созданы представления, через которые и будет происходить манипуляция с данными военнослужащих, оружия, транспорта и подразделений. На представление для каждой сущности устанавливаются триггеры: INSTEAD OF INSERT, INSTEAD OF UPDATE и INSTEAD OF DELETE.

Триггеры INSTEAD OF INSERT работают так, сначала происходит вставка в общую таблицу для класса, потом в таблицу частную.

Триггеры INSTEAD OF UPDATE работают так, они проверят равенство идентификаторов в старой записи и в новой, если они не совпадают, то выбрасывается исключение, после этого данные изменяются в используемых триггером таблицах.

Триггеры INSTEAD OF DELETE сначала удаляют данные из частной таблицы, потом из общей.

Прилагаю создание представления и триггеров для сущности assault\_rifles.

CREATE OR REPLACE VIEW assault\_rifles\_view(id, name, ammo, fire\_rate) as

SELECT ar.id,

weapons.name,

ar.ammo,

ar.fire\_rate

FROM weapons

JOIN assault\_rifles ar ON weapons.id = ar.id;

CREATE OR REPLACE FUNCTION assault\_rifles\_view\_insert() RETURNS TRIGGER AS

$$

DECLARE

insert\_id bigint;

BEGIN

Insert Into weapons(type, name) VALUES (2, NEW.name) RETURNING id INTO insert\_id;

Insert Into assault\_rifles VALUES (insert\_id, NEW.fire\_rate, NEW.ammo);

return NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION assault\_rifles\_view\_delete() RETURNS TRIGGER AS

$$

DECLARE

BEGIN

DELETE FROM assault\_rifles WHERE id = OLD.id;

DELETE FROM weapons WHERE id = OLD.id;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION assault\_rifles\_view\_update() RETURNS TRIGGER AS

$$

DECLARE

BEGIN

if OLD.id != NEW.id THEN

RAISE EXCEPTION 'you cannot update id in this table';

end if;

UPDATE assault\_rifles SET fire\_rate = NEW.fire\_rate, ammo = NEW.ammo WHERE id = NEW.id;

UPDATE weapons SET name = NEW.name WHERE id = NEW.id;

return NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER assault\_rifles\_view\_insert\_trigger

INSTEAD OF INSERT

ON assault\_rifles\_view

FOR EACH ROW

EXECUTE FUNCTION assault\_rifles\_view\_insert();

CREATE OR REPLACE TRIGGER assault\_rifles\_view\_delete\_trigger

INSTEAD OF DELETE

ON assault\_rifles\_view

FOR EACH ROW

EXECUTE FUNCTION assault\_rifles\_view\_delete();

CREATE OR REPLACE TRIGGER assault\_rifles\_view\_update\_trigger

INSTEAD OF UPDATE

ON assault\_rifles\_view

FOR EACH ROW

EXECUTE FUNCTION assault\_rifles\_view\_update();

## Клиентское приложения

Приложение написано на языке программирования C++ с использованием набора библиотек Qt. Из данной библиотеки использовались модули Core, Widgets, SQL.

Приложение состоит из главного меню, набора форм для изменения данных, средств выбора параметров для запросов и системы отображения запросов.

Приложение так же включает в себя ролевую систему. Пользователи с ролью HR могут изменять данные связанные с военнослужащими. Пользователи с ролью SUPPLIER могут изменять данные связанные с вооружением и транспортом. Пользователи с ролью ADMIN могут менять все данные. Выполнять заранее подготовленные запросы с различными параметрами могут все пользователи.

Приложение встречает пользователя окном входа в систему.

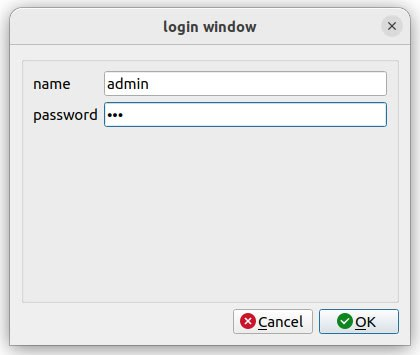


Рисунок 1 Окно входа в систему

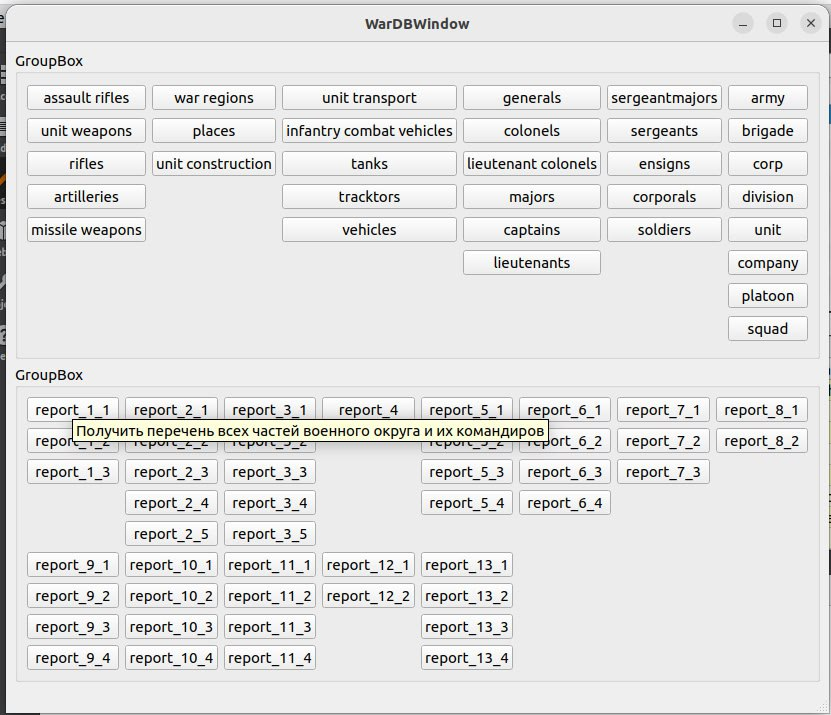
После успешного захода, пользователь видит главное меню приложения, в котором при наведении на запрос с номером он увидит, что делает данный запрос. 

Рисунок 2 Главное меню

Дальше пользователь может выбрать таблицу для редактирования, только ту на которую у него хватит прав.

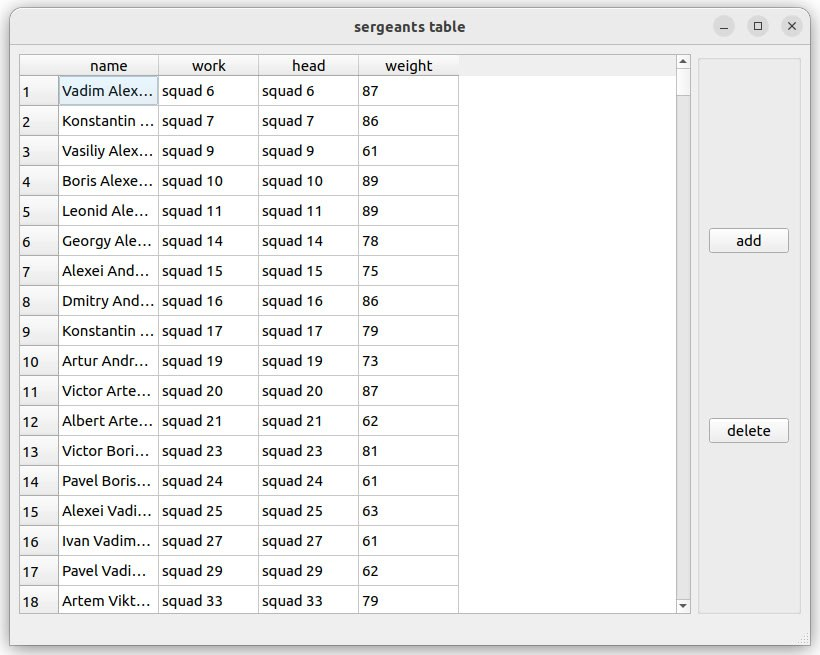


Рисунок 3 Таблица Сержантов

Пользователь может менять данные в строках таблицы, или он может выбрать строку или строки и удалить их, нажав на кнопку delete, или он может решить добавить данные, для этого он может нажать кнопку add и добавить новую запись в таблицу.

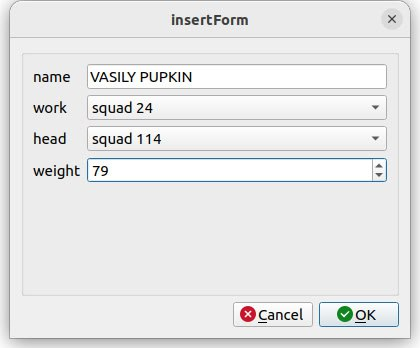


Рисунок 4 Форма добавления данных в таблицу

На данном Этапе вернёмся в главное меню и рассмотрим выполнение запросов. Пользователь выбирает интересующей его запрос, и в зависимости от запроса ему могут предложить выбрать параметры для запроса.

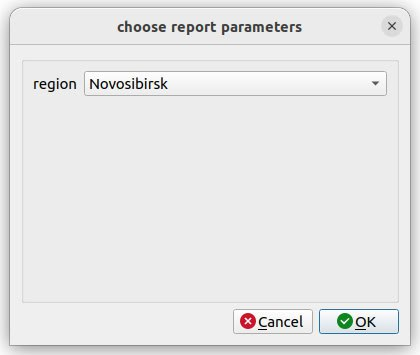


Рисунок 5 Форма для выбора параметра запроса

И после выбора всех параметров пользователю покажут результат выполнения запроса.



Рисунок 6 Результат выполнения запроса

## Выводы

За время выполнения данной работы был получен ценный опыт проектирования базы данных и написания сложных запросов к ней. Были реализованы хранимые функции и триггеры на основе них, всё это было написано на новом для меня языке plpgsql. Во время реализации клиентского приложения, был получен опыт работы с базой данных на языке высокого уровня C++.