**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю. А.»**

Институт  прикладных информационных технологий и коммуникаций

Кафедра Прикладные информационные технологии

**КУРСОВАЯ РАБОТА**

по дисциплине Б.1.2.0.8 «Современные технологии управления данными в объектно-реляционных СУБД»

на тему «***Реализация объектно-ориентированной базы данных для видеоигры»***

Выполнил студент группы Б1-ИФСТ-31

Дувалов Владислав Станиславович

Проверил: преподаватель

Кузьмин Алексей Константинович

Комиссия по защите:

доцент кафедры ПИТ Кузьмин А.К.

ассистент кафедры ПИТ Печкин И.О.

Курсовая работа защищена на оценку «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмин А. К.

(дата, подпись члена комиссии)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Печкин И. О.

(дата, подпись члена комиссии)

Саратов 2023

**Замечания**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмин А. К.

(дата, подпись члена комиссии)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Печкин И. О.

(дата, подпись члена комиссии)

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Кафедра «Прикладные информационные технологии»

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине «Технологии управления данными

в объектно-реляционных СУБД»

студенту ИнПИТ группы Б1ИФСТ-31 Дувалову Владиславу Станиславовичу

В курсовой работе необходимо:

Разработать базу данных в объектно-реляционной СУБД PostgreSQL для ***видеоигры***, для чего

1. Провести анализ выбранной предметной области,
2. Реализовать базу данных, используя возможности объектно-реляционной СУБД PostgreSQL
3. Протестировать базу данных на тестовых данных

Дата выдачи:    «5»    сентября 2023

Срок выполнения: «23»    декабря 2023

Руководитель:\_\_\_\_\_\_\_\_\_ Кузьмин А. К.

Студент:\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дувалов В. С.

**Оглавление**

**Цель и задачи курсовой работы**

Цель курсовой работы - получение практических навыков управления и разработки баз данных в объектно-реляционной СУБД  PostgreSQL, в определенной предметной области.

Задачи курсовой работы:

1. Проектирование базы данных с более чем 6 таблицами, в которых должны быть ограничения: primary key, foreign key, default, identity, check, not null, unique.

2. Заполнение таблиц тестовыми данными.

3. Создание индексов для таблиц, чтобы увеличить производительность.

4. Реализация аналитических представлений, в которых должны быть следующие возможности SELECT: операторы группировки, итогов и подитогов, ранжирования и агрегирования (в том числе с выражением partition by), а также обобщённые (в том числе рекурсивные) табличные выражения.

5. Создание 3 курсоров и курсорных переменных различных типов.

6. Создание 2 функций, 3 процедур и 2 триггеров.

7. Создание сложных составных типов и использование их в качестве полей в таблицах и параметров в блоках.

8. Организация транзакционной обработки данных при различных установленных уровнях изолированности транзакций.

В результате выполнения курсовой работы, должны быть сформированы компетенции:

ОПК-2. Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности

ОПК-6. Способен представлять полученные результаты научно-исследовательской деятельности на высоком уровне и с учетом соблюдения авторских прав.

# Краткое описание используемой СУБД

PostgreSQL – это мощная объектно-реляционная система управления базами данных, обладающая рядом ключевых характеристик, делающих ее привлекательным выбором для широкого спектра приложений:

**Открытый исходный код:** PostgreSQL является открытым программным обеспечением, что позволяет пользователям свободно распространять, изменять и улучшать его. Эта особенность способствует активному сообществу разработчиков, постоянным обновлениям и высокой степени надежности.

**Расширенные возможности SQL:** СУБД поддерживает полный стандарт SQL и предлагает дополнительные возможности, такие как сложные типы данных, геопространственные запросы, а также возможности расширения собственных функций и операторов.

**Многопользовательская поддержка:** PostgreSQL обеспечивает эффективное управление множеством пользователей, предоставляя гибкие механизмы контроля доступа и обеспечивая изоляцию данных между параллельными сессиями.

**Транзакционная безопасность:** Система гарантирует ACID-свойства транзакций (Atomicity, Consistency, Isolation, Durability), обеспечивая надежность и целостность данных даже в случае сбоев.

**Масштабируемость и производительность:** PostgreSQL спроектирована для эффективной работы с различными объемами данных и обладает оптимизациями для повышения производительности в условиях высоких нагрузок.

Выбор PostgreSQL в качестве СУБД для проекта обеспечивает надежность, гибкость и расширенные возможности для эффективного управления данными.

# Краткое описание предметной области

Видеоигра, как предметная область,  представляет собой интерпретацию настольной ролевой игры, перенесенную на персональные компьютеры. Последнее время все больше студий стараются создать свою настольную ролевую игру на ПК, как по оригинальной вселенной, так и по лицензии. Фанаты же делают настольные ролевые игры без лицензии.

Данная видеоигра сделана по вселенной Warhammer 40,000 и ей требуется база данных из-за обилия информации и необходимости ее структурировать. В базе данных будут  храниться все необходимые аспекты для проведения ролевых игр как в рамках видеоигры, так и вне ее. Это достигается путем создания множества представлений и грамотного наполнения самой базы данных исключительно необходимой информацией:

1. Какие способности используют те или иные персонажи

2. Какую броню  используют персонажи

3. Какое снаряжение используют персонажи

4. Сколько и какие персонажи находятся на данный момент в ролевой игре

Это большой пласт знаний и неопытный ведущий не сможет грамотно оперировать всей этой информацией. А видеоигра не способна столько данных хранить “в себе” без ущерба сложности или производительности.

# Реализация базы данных для видеоигры

Так как работа выполнялась в паре часть зависимостей была создана Сергеем Кузьминым. Вот диаграмма распределения обязанностей:  


Рисунок 1- Диаграмма связей в таблице

Зеленым обозначена сфера обязанностей Владислава Дувалова, а желтым обозначена сфера обязанностей Сергея Кузьмина.

### **Проектирование и реализация таблиц и ограничений базы.**

1. Создание таблицы “attackAbility” или “Способности атаки”. Данная таблица представляет собой перечень атак и приемов, которые могут применять герои напрямую через класс или опосредованно через оружие.

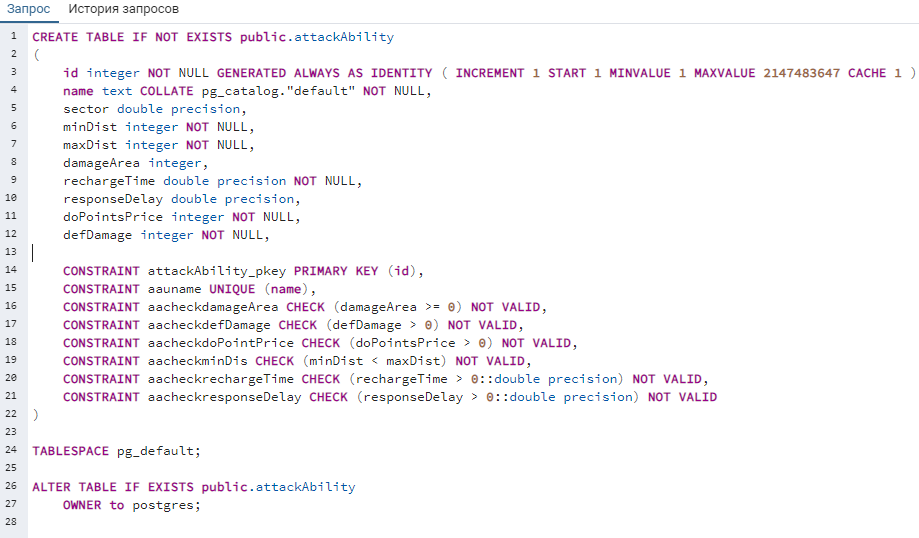


Рисунок 2 - Создание таблицы attackAbility и ее ограничений

CREATE TABLE IF NOT EXISTS public.attackAbility

(

    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    name text COLLATE pg\_catalog."default" NOT NULL,

    sector double precision,

    minDist integer NOT NULL,

    maxDist integer NOT NULL,

    damageArea integer,

    rechargeTime double precision NOT NULL,

    responseDelay double precision,

    doPointsPrice integer NOT NULL,

    defDamage integer NOT NULL,

    CONSTRAINT attackAbility\_pkey PRIMARY KEY (id),

    CONSTRAINT aauname UNIQUE (name),

    CONSTRAINT aacheckdamageArea CHECK (damageArea >= 0) NOT VALID,

    CONSTRAINT aacheckdefDamage CHECK (defDamage > 0) NOT VALID,

    CONSTRAINT aacheckdoPointPrice CHECK (doPointsPrice > 0) NOT VALID,

    CONSTRAINT aacheckminDis CHECK (minDist < maxDist) NOT VALID,

    CONSTRAINT aacheckrechargeTime CHECK (rechargeTime > 0::double precision) NOT VALID,

    CONSTRAINT aacheckresponseDelay CHECK (responseDelay > 0::double precision) NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.attackAbility

    OWNER to postgres;

2. Создание таблицы “moveAbility” или “Способоности передвижения”. Данная таблица представляет собой перечень способностей, которые активируются героем для перемещения по миру.



Рисунок 3 - Создание таблицы moveAbility и ее ограничений

CREATE TABLE IF NOT EXISTS public.moveAbility

(

    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    name text COLLATE pg\_catalog."default" NOT NULL,

    rangeIndex double precision,

    doPointPrice integer NOT NULL,

    type integer NOT NULL,

    CONSTRAINT moveAbility\_pkey PRIMARY KEY (id),

    CONSTRAINT moveabilityunamew UNIQUE (name),

    CONSTRAINT ma\_to\_typeofma FOREIGN KEY (type)

        REFERENCES public.moveAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT macheckdoPointPrice CHECK (doPointPrice > 0) NOT VALID,

    CONSTRAINT macheckrangeindex CHECK (rangeIndex > 0::double precision) NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.moveAbility

    OWNER to postgres;

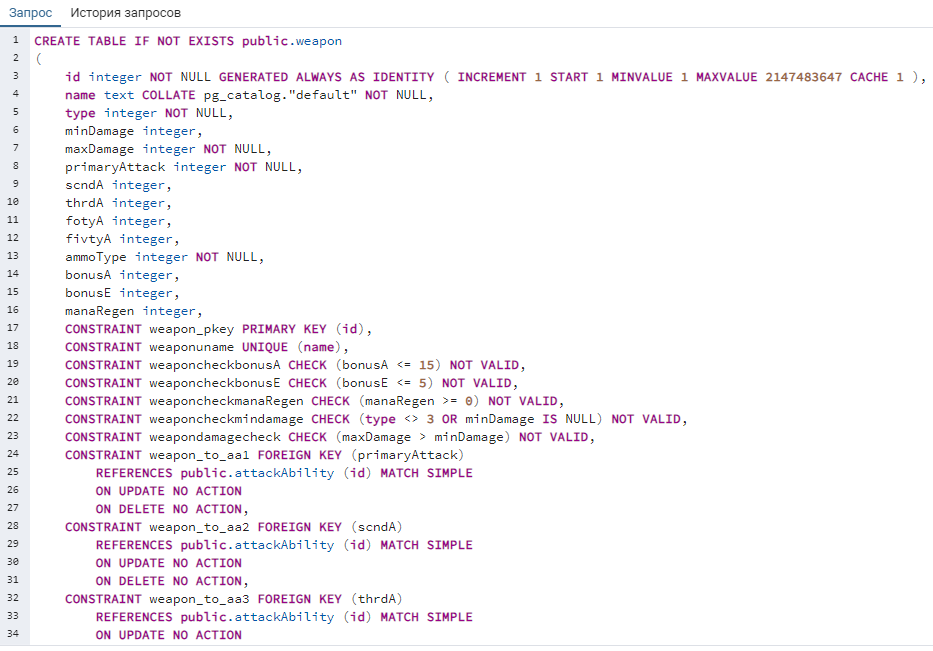
3. Создание таблицы “weapon” или “Оружие”. Данная таблица представляет собой перечень вооружения, что может использовать герой в своих приключениях. 

Рисунок 4 – Создание таблицы weapon, ее ограничений и связей.

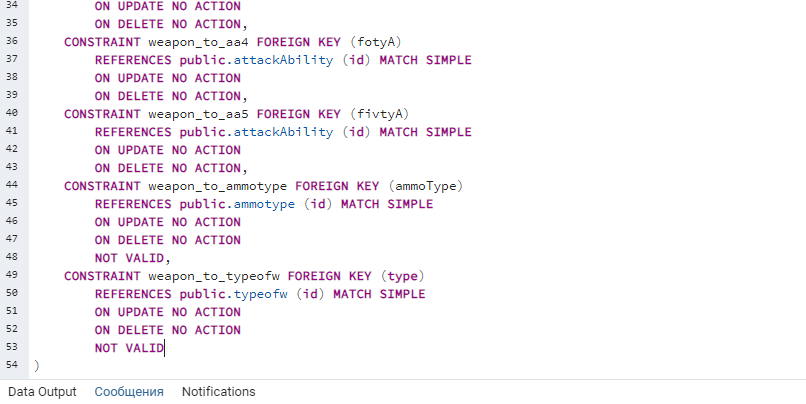


Рисунок 5 - Создание таблицы weapon, ее ограничений и связей.

CREATE TABLE IF NOT EXISTS public.weapon

(

    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    name text COLLATE pg\_catalog."default" NOT NULL,

    type integer NOT NULL,

    minDamage integer,

    maxDamage integer NOT NULL,

    primaryAttack integer NOT NULL,

    scndA integer,

    thrdA integer,

    fotyA integer,

    fivtyA integer,

    ammoType integer NOT NULL,

    bonusA integer,

    bonusE integer,

    manaRegen integer,

    CONSTRAINT weapon\_pkey PRIMARY KEY (id),

    CONSTRAINT weaponuname UNIQUE (name),

    CONSTRAINT weapon\_to\_aa1 FOREIGN KEY (primaryAttack)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT weapon\_to\_aa2 FOREIGN KEY (scndA)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT weapon\_to\_aa3 FOREIGN KEY (thrdA)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT weapon\_to\_aa4 FOREIGN KEY (fotyA)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT weapon\_to\_aa5 FOREIGN KEY (fivtyA)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT weapon\_to\_ammotype FOREIGN KEY (ammoType)

        REFERENCES public.ammotype (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT weapon\_to\_typeofw FOREIGN KEY (type)

        REFERENCES public.typeofw (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT weaponcheckbonusA CHECK (bonusA <= 15) NOT VALID,

    CONSTRAINT weaponcheckbonusE CHECK (bonusE <= 5) NOT VALID,

    CONSTRAINT weaponcheckmanaRegen CHECK (manaRegen >= 0) NOT VALID,

    CONSTRAINT weaponcheckmindamage CHECK (type <> 3 OR minDamage IS NULL) NOT VALID,

    CONSTRAINT weapondamagecheck CHECK (maxDamage > minDamage) NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.weapon

    OWNER to postgres;

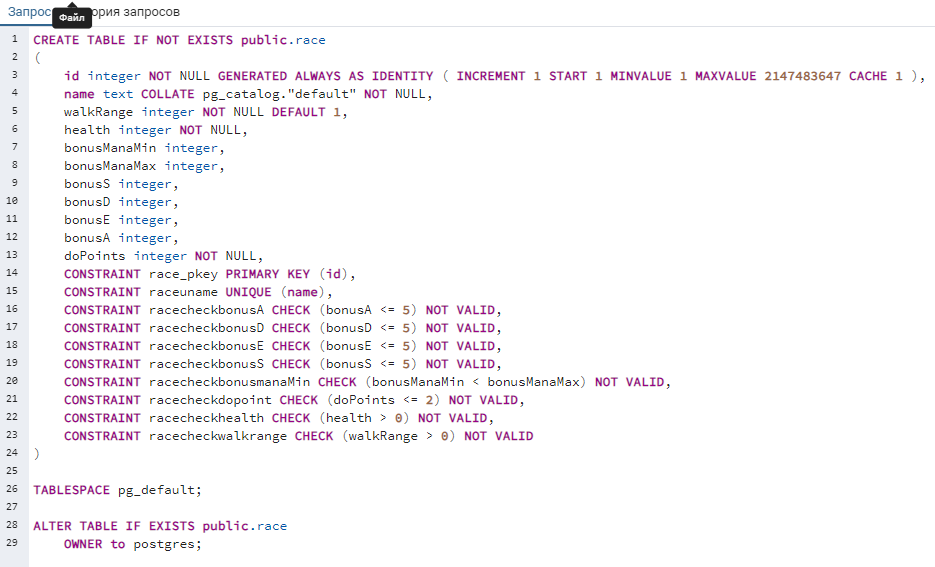
1. Создание таблицы race или Раса. Данная таблица содержит в себе всевозможные расы за которые может играть игрок 

Рисунок 6 – Создание таблицы race и ее ограничений

1. Создание таблицы armor или “Броня”. Данная таблица представляет собой перечень брони, которую может носить герой.

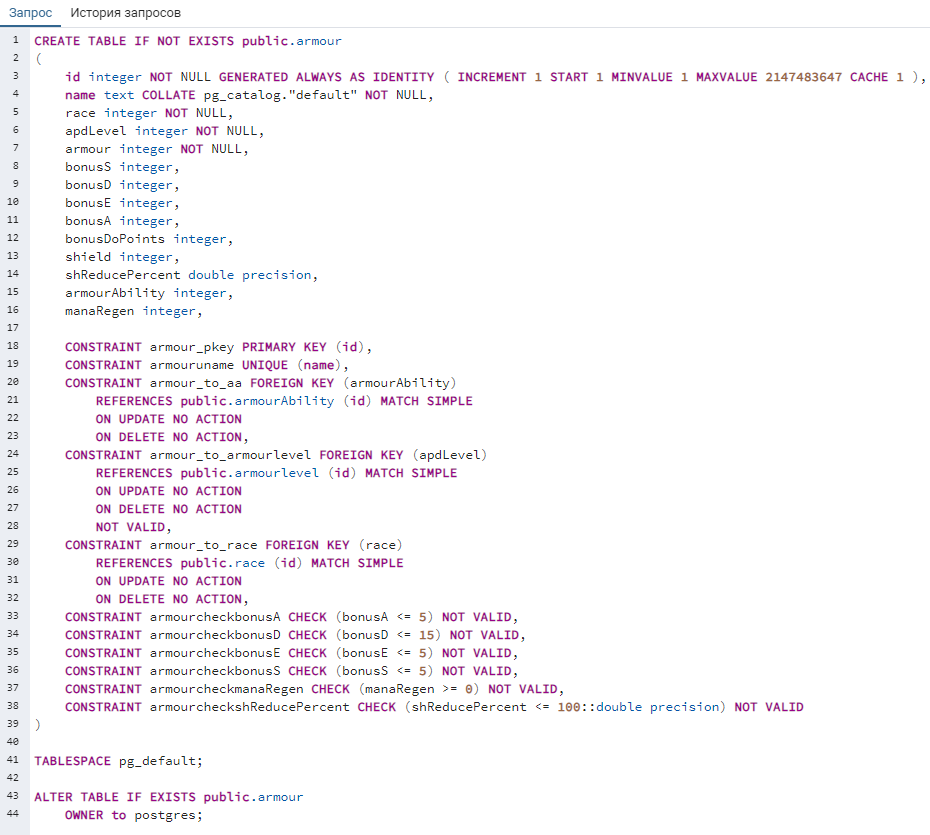


Рисунок 7 - Создание таблицы armour, ее ограничений и связей

CREATE TABLE IF NOT EXISTS public.armour

(

    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    name text COLLATE pg\_catalog."default" NOT NULL,

    race integer NOT NULL,

    apdLevel integer NOT NULL,

    armour integer NOT NULL,

    bonusS integer,

    bonusD integer,

    bonusE integer,

    bonusA integer,

    bonusDoPoints integer,

    shield integer,

    shReducePercent double precision,

    armourAbility integer,

    manaRegen integer,

    CONSTRAINT armour\_pkey PRIMARY KEY (id),

    CONSTRAINT armouruname UNIQUE (name),

    CONSTRAINT armour\_to\_aa FOREIGN KEY (armourAbility)

        REFERENCES public.armourAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT armour\_to\_armourlevel FOREIGN KEY (apdLevel)

        REFERENCES public.armourlevel (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT armour\_to\_race FOREIGN KEY (race)

        REFERENCES public.race (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT armourcheckbonusA CHECK (bonusA <= 5) NOT VALID,

    CONSTRAINT armourcheckbonusD CHECK (bonusD <= 15) NOT VALID,

    CONSTRAINT armourcheckbonusE CHECK (bonusE <= 5) NOT VALID,

    CONSTRAINT armourcheckbonusS CHECK (bonusS <= 5) NOT VALID,

    CONSTRAINT armourcheckmanaRegen CHECK (manaRegen >= 0) NOT VALID,

    CONSTRAINT armourcheckshReducePercent CHECK (shReducePercent <= 100::double precision) NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.armour

    OWNER to postgres;

1. Создание таблицы class или Класс. Данная таблица содержит в себе всевозможные классы, которые доступны герою.

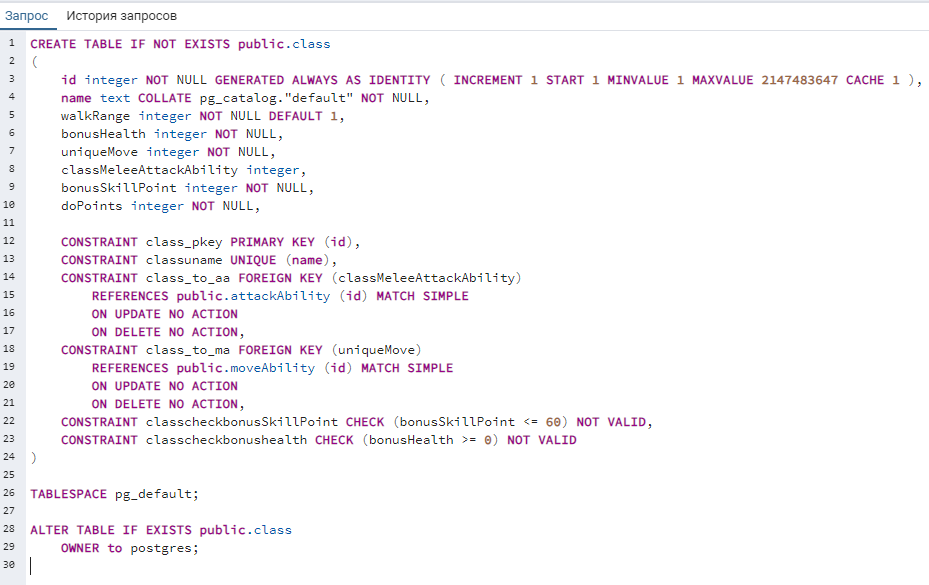


Рисунок 8 - Создание таблицы class, ее ограничений и зависимостей

CREATE TABLE IF NOT EXISTS public.class

(

    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    name text COLLATE pg\_catalog."default" NOT NULL,

    walkRange integer NOT NULL DEFAULT 1,

    bonusHealth integer NOT NULL,

    uniqueMove integer NOT NULL,

    classMeleeAttackAbility integer,

    bonusSkillPoint integer NOT NULL,

    doPoints integer NOT NULL,

    CONSTRAINT class\_pkey PRIMARY KEY (id),

    CONSTRAINT classuname UNIQUE (name),

    CONSTRAINT class\_to\_aa FOREIGN KEY (classMeleeAttackAbility)

        REFERENCES public.attackAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT class\_to\_ma FOREIGN KEY (uniqueMove)

        REFERENCES public.moveAbility (id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT classcheckbonusSkillPoint CHECK (bonusSkillPoint <= 60) NOT VALID,

    CONSTRAINT classcheckbonushealth CHECK (bonusHealth >= 0) NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.class

    OWNER to postgres;

1. Создание таблинцы hero или Герой. В данной таблицы содержатся записи о всевозможных героях.

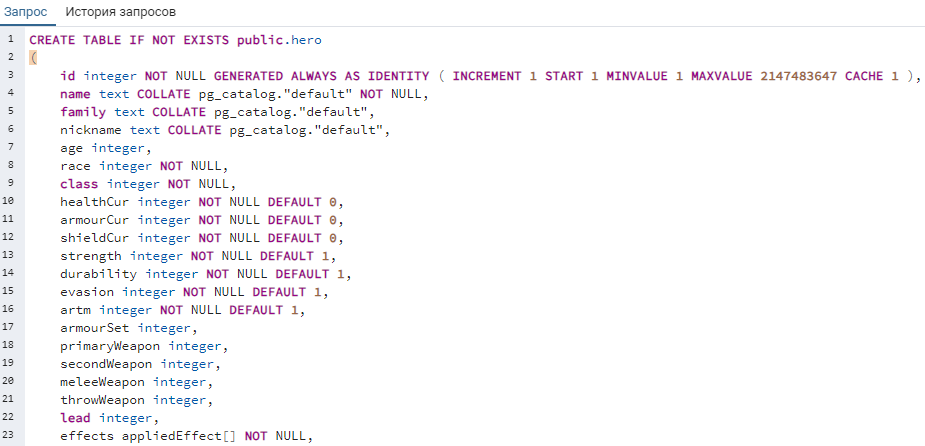


Рисунок 9 - Создание таблицы hero, её ограничений и связей

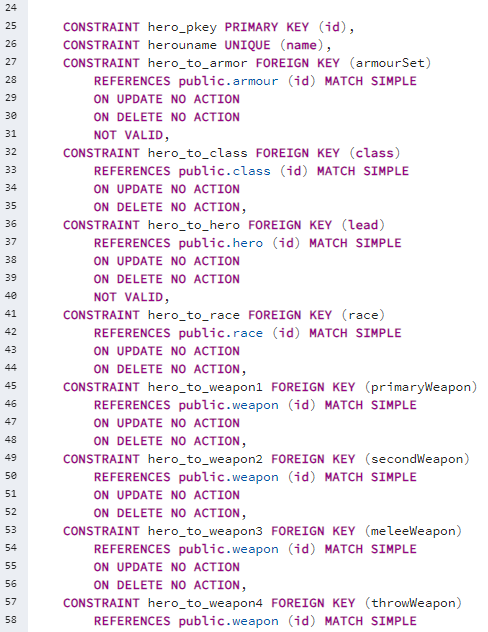


Рисунок 10 - Создание таблицы hero, её ограничений и связей

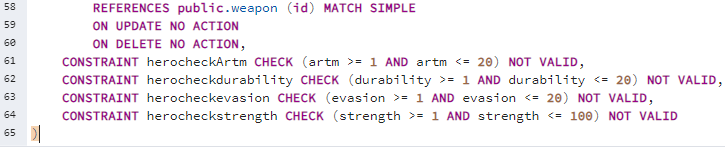


Рисунок 11 - Создание таблицы hero, её ограничений и связей

**Заполнение таблиц тестовыми данными**

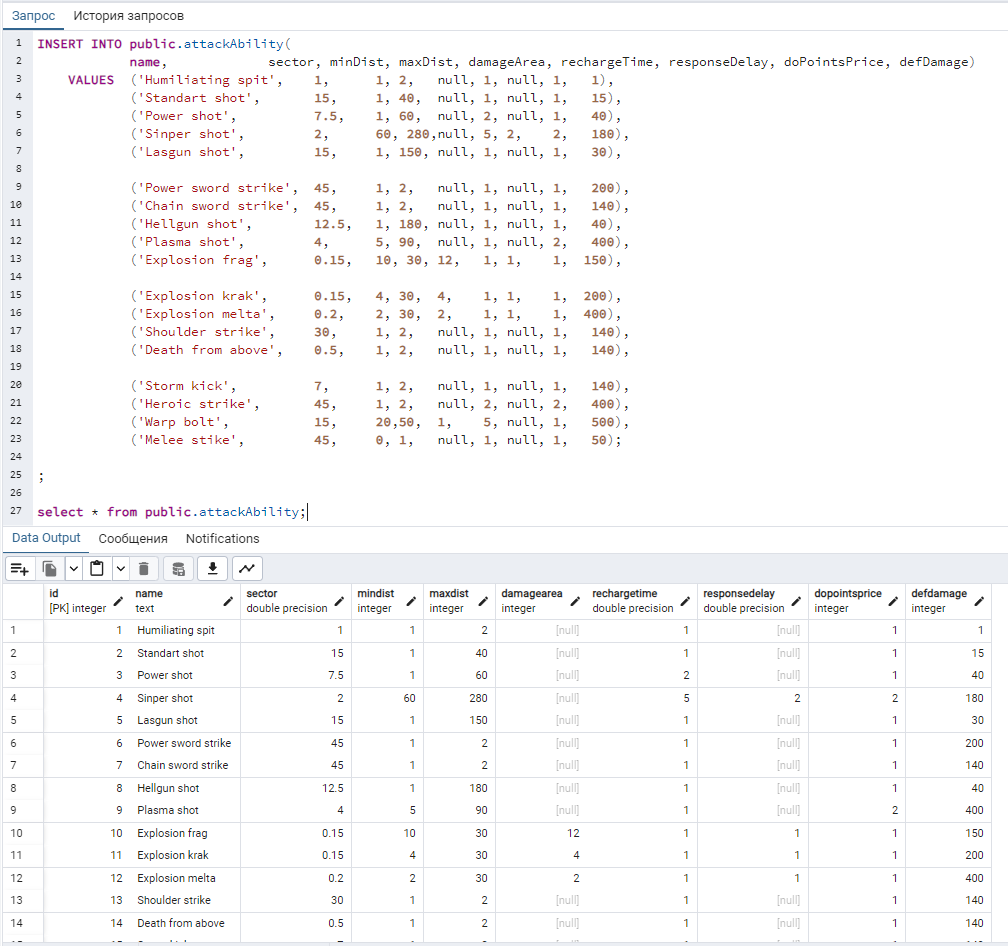
Далее будут приложены скриншоты и код выполнения заполнения базы данных.  


Рисунок 12 - Заполнение таблицы attackAbility

INSERT INTO public.attackAbility(

            name,             sector, minDist, maxDist, damageArea, rechargeTime, responseDelay, doPointsPrice, defDamage)

    VALUES  ('Humiliating spit',    1,      1, 2,   null, 1, null, 1,   1),

            ('Standart shot',       15,     1, 40,  null, 1, null, 1,   15),

            ('Power shot',          7.5,    1, 60,  null, 2, null, 1,   40),

            ('Sinper shot',         2,      60, 280,null, 5, 2,    2,   180),

            ('Lasgun shot',         15,     1, 150, null, 1, null, 1,   30),

            ('Power sword strike',  45,     1, 2,   null, 1, null, 1,   200),

            ('Chain sword strike',  45,     1, 2,   null, 1, null, 1,   140),

            ('Hellgun shot',        12.5,   1, 180, null, 1, null, 1,   40),

            ('Plasma shot',         4,      5, 90,  null, 1, null, 2,   400),

            ('Explosion frag',      0.15,   10, 30, 12,   1, 1,    1,  150),

            ('Explosion krak',      0.15,   4, 30,  4,    1, 1,    1,  200),

            ('Explosion melta',     0.2,    2, 30,  2,    1, 1,    1,  400),

            ('Shoulder strike',     30,     1, 2,   null, 1, null, 1,   140),

            ('Death from above',    0.5,    1, 2,   null, 1, null, 1,   140),

            ('Storm kick',          7,      1, 2,   null, 1, null, 1,   140),

            ('Heroic strike',       45,     1, 2,   null, 2, null, 2,   400),

            ('Warp bolt',           15,     20,50,  1,    5, null, 1,   500),

            ('Melee stike',         45,     0, 1,   null, 1, null, 1,   50);

select \* from public.attackAbility;

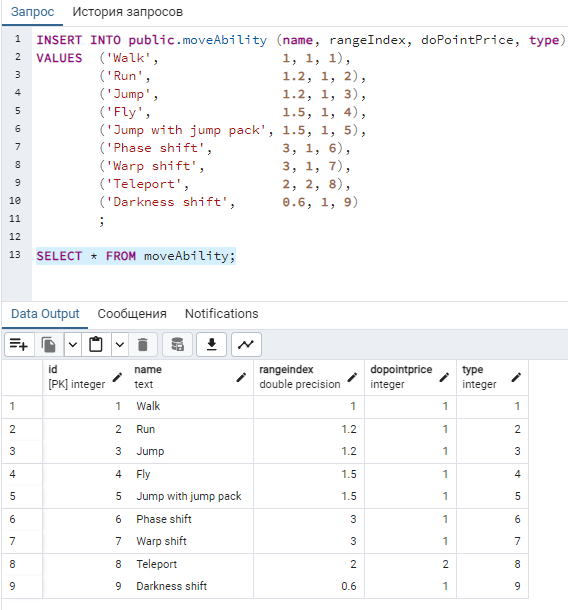


Рисунок 13 - Заполнение таблицы moveAbility

INSERT INTO public.moveAbility (name, rangeIndex, doPointPrice, type)

VALUES  ('Walk',                1, 1, 1),

        ('Run',                 1.2, 1, 2),

        ('Jump',                1.2, 1, 3),

        ('Fly',                 1.5, 1, 4),

        ('Jump with jump pack', 1.5, 1, 5),

        ('Phase shift',         3, 1, 6),

        ('Warp shift',          3, 1, 7),

        ('Teleport',            2, 2, 8),

        ('Darkness shift',      0.6, 1, 9)

        ;

SELECT \* FROM moveAbility;

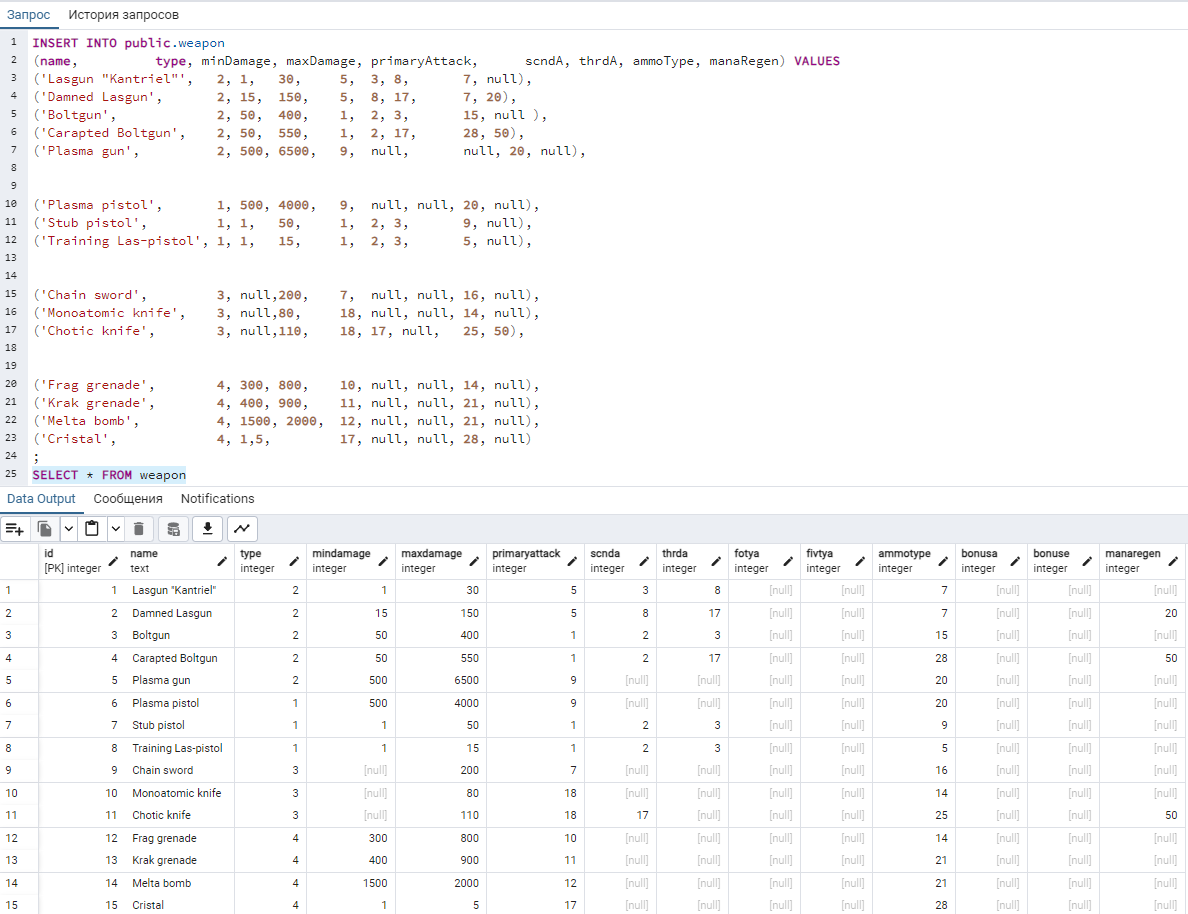


Рисунок 14 - Заполнение таблицы weapon

INSERT INTO public.weapon

(name,          type, minDamage, maxDamage, primaryAttack,      scndA, thrdA, ammoType, manaRegen) VALUES

('Lasgun "Kantriel"',   2, 1,   30,     5,  3, 8,       7, null),

('Damned Lasgun',       2, 15,  150,    5,  8, 17,      7, 20),

('Boltgun',             2, 50,  400,    1,  2, 3,       15, null ),

('Carapted Boltgun',    2, 50,  550,    1,  2, 17,      28, 50),

('Plasma gun',          2, 500, 6500,   9,  null,       null, 20, null),

('Plasma pistol',       1, 500, 4000,   9,  null, null, 20, null),

('Stub pistol',         1, 1,   50,     1,  2, 3,       9, null),

('Training Las-pistol', 1, 1,   15,     1,  2, 3,       5, null),

('Chain sword',         3, null,200,    7,  null, null, 16, null),

('Monoatomic knife',    3, null,80,     18, null, null, 14, null),

('Chotic knife',        3, null,110,    18, 17, null,   25, 50),

('Frag grenade',        4, 300, 800,    10, null, null, 14, null),

('Krak grenade',        4, 400, 900,    11, null, null, 21, null),

('Melta bomb',          4, 1500, 2000,  12, null, null, 21, null),

('Cristal',             4, 1,5,         17, null, null, 28, null)

;

SELECT \* FROM weapon

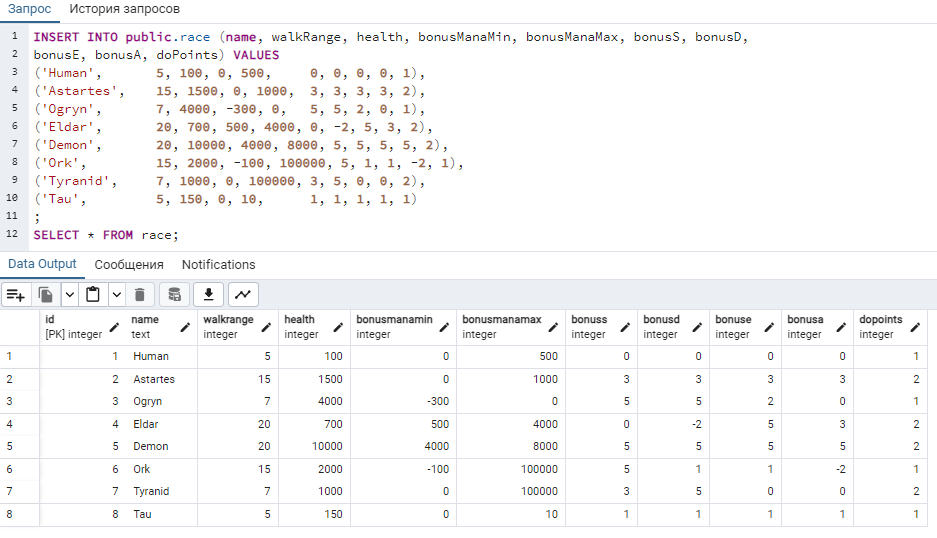


Рисунок 15 - Заполнение таблицы race

INSERT INTO public.race (name, walkRange, health, bonusManaMin, bonusManaMax, bonusS, bonusD,

bonusE, bonusA, doPoints) VALUES

('Human',       5, 100, 0, 500,     0, 0, 0, 0, 1),

('Astartes',    15, 1500, 0, 1000,  3, 3, 3, 3, 2),

('Ogryn',       7, 4000, -300, 0,   5, 5, 2, 0, 1),

('Eldar',       20, 700, 500, 4000, 0, -2, 5, 3, 2),

('Demon',       20, 10000, 4000, 8000, 5, 5, 5, 5, 2),

('Ork',         15, 2000, -100, 100000, 5, 1, 1, -2, 1),

('Tyranid',     7, 1000, 0, 100000, 3, 5, 0, 0, 2),

('Tau',         5, 150, 0, 10,      1, 1, 1, 1, 1)

;

SELECT \* FROM race;

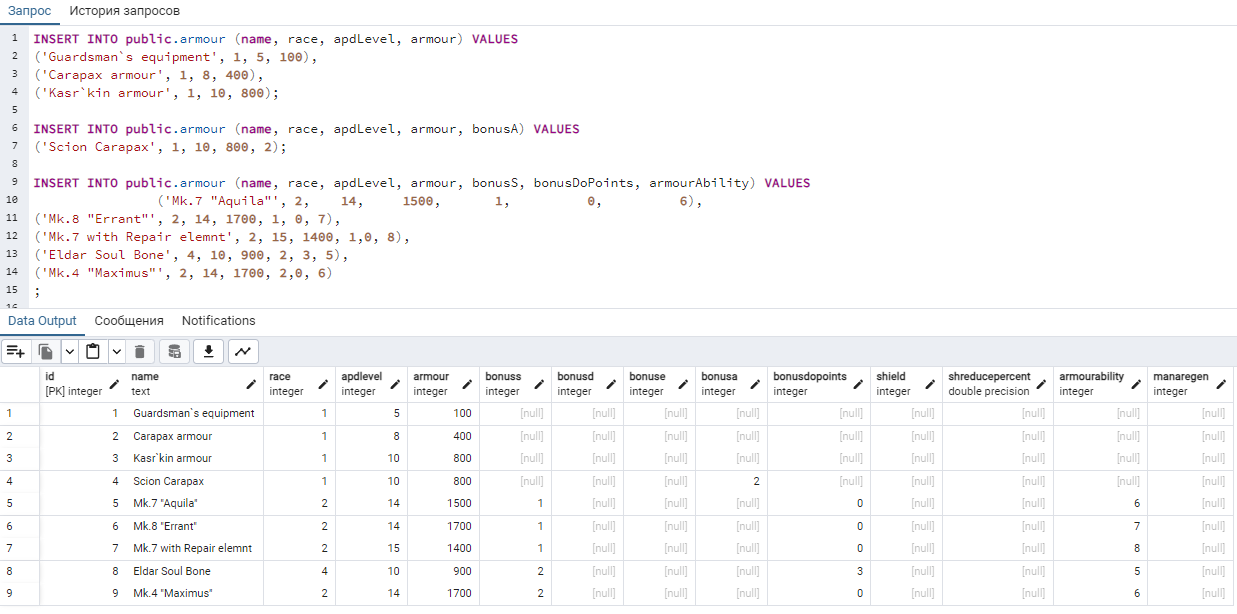


Рисунок 16 - Заполнение таблицы armour

INSERT INTO public.armour (name, race, apdLevel, armour) VALUES

('Guardsman`s equipment', 1, 5, 100),

('Carapax armour', 1, 8, 400),

('Kasr`kin armour', 1, 10, 800);

INSERT INTO public.armour (name, race, apdLevel, armour, bonusA) VALUES

('Scion Carapax', 1, 10, 800, 2);

INSERT INTO public.armour (name, race, apdLevel, armour, bonusS, bonusDoPoints, armourAbility) VALUES

                ('Mk.7 "Aquila"', 2,    14,     1500,       1,          0,          6),

('Mk.8 "Errant"', 2, 14, 1700, 1, 0, 7),

('Mk.7 with Repair elemnt', 2, 15, 1400, 1,0, 8),

('Eldar Soul Bone', 4, 10, 900, 2, 3, 5),

('Mk.4 "Maximus"', 2, 14, 1700, 2,0, 6)

;

SELECT \* FROM armour;

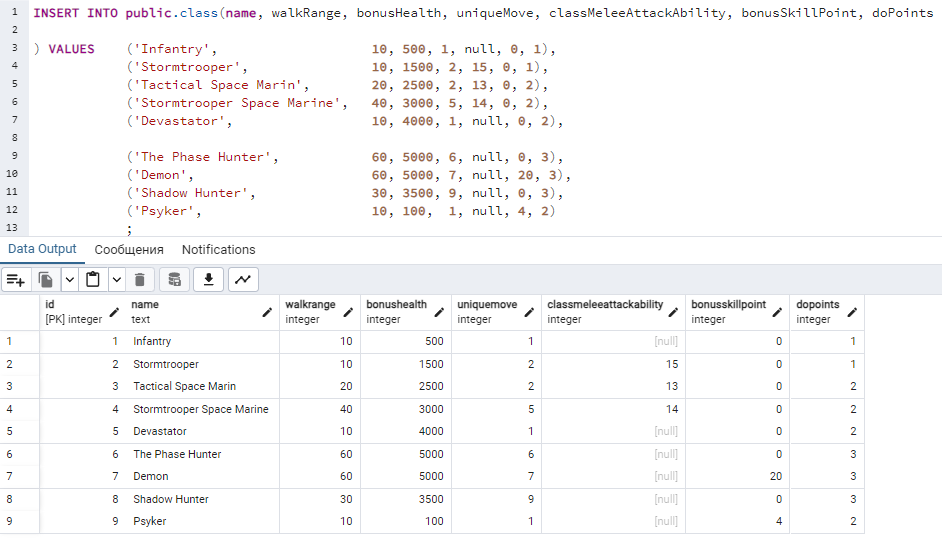


Рисунок 17 - Заполнение таблицы class

INSERT INTO public.class(name, walkRange, bonusHealth, uniqueMove, classMeleeAttackAbility, bonusSkillPoint, doPoints

) VALUES    ('Infantry',                    10, 500, 1, null, 0, 1),

            ('Stormtrooper',                10, 1500, 2, 15, 0, 1),

            ('Tactical Space Marin',        20, 2500, 2, 13, 0, 2),

            ('Stormtrooper Space Marine',   40, 3000, 5, 14, 0, 2),

            ('Devastator',                  10, 4000, 1, null, 0, 2),

            ('The Phase Hunter',            60, 5000, 6, null, 0, 3),

            ('Demon',                       60, 5000, 7, null, 20, 3),

            ('Shadow Hunter',               30, 3500, 9, null, 0, 3),

            ('Psyker',                      10, 100,  1, null, 4, 2)

            ;

SELECT \* FROM public.class;

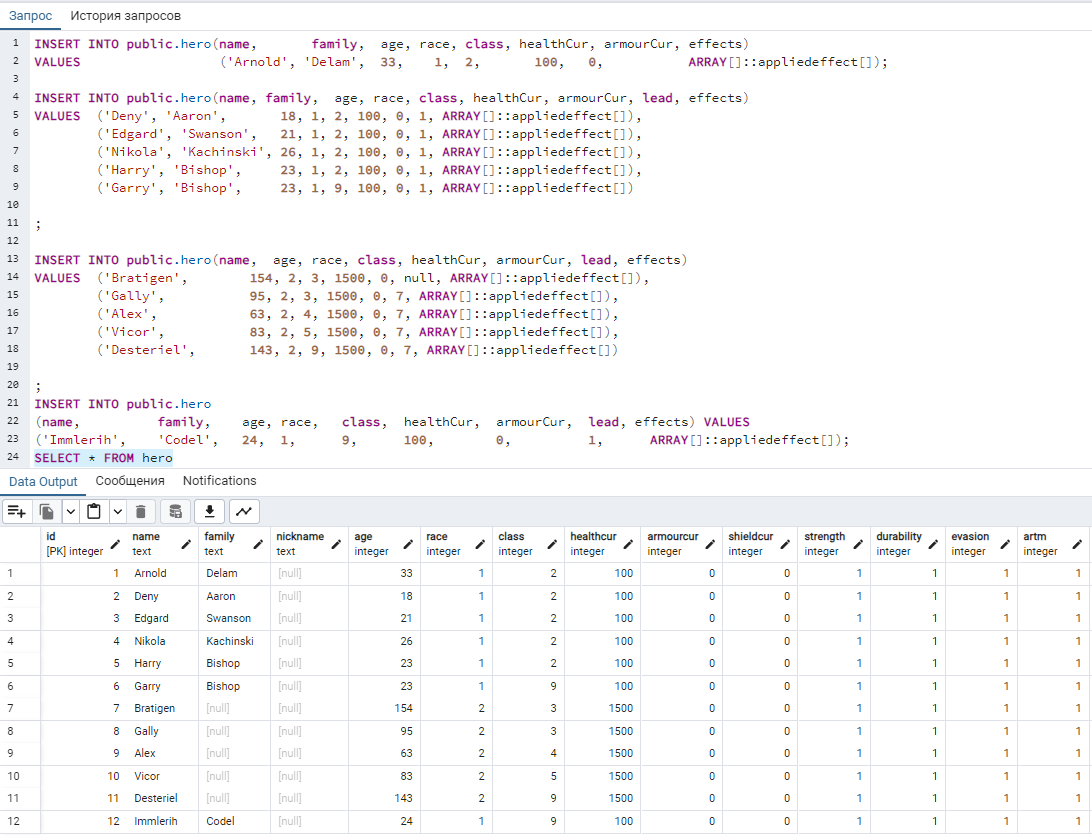


Рисунок 18 - Заполнение таблицы hero

INSERT INTO public.hero(name,       family,  age, race, class, healthCur, armourCur, effects)

VALUES                  ('Arnold', 'Delam',  33,    1,  2,       100,   0,           ARRAY[]::appliedeffect[]);

INSERT INTO public.hero(name, family,  age, race, class, healthCur, armourCur, lead, effects)

VALUES  ('Deny', 'Aaron',       18, 1, 2, 100, 0, 1, ARRAY[]::appliedeffect[]),

        ('Edgard', 'Swanson',   21, 1, 2, 100, 0, 1, ARRAY[]::appliedeffect[]),

        ('Nikola', 'Kachinski', 26, 1, 2, 100, 0, 1, ARRAY[]::appliedeffect[]),

        ('Harry', 'Bishop',     23, 1, 2, 100, 0, 1, ARRAY[]::appliedeffect[]),

        ('Garry', 'Bishop',     23, 1, 9, 100, 0, 1, ARRAY[]::appliedeffect[])

;

INSERT INTO public.hero(name,  age, race, class, healthCur, armourCur, lead, effects)

VALUES  ('Bratigen',        154, 2, 3, 1500, 0, null, ARRAY[]::appliedeffect[]),

        ('Gally',           95, 2, 3, 1500, 0, 7, ARRAY[]::appliedeffect[]),

        ('Alex',            63, 2, 4, 1500, 0, 7, ARRAY[]::appliedeffect[]),

        ('Vicor',           83, 2, 5, 1500, 0, 7, ARRAY[]::appliedeffect[]),

        ('Desteriel',       143, 2, 9, 1500, 0, 7, ARRAY[]::appliedeffect[])

;

INSERT INTO public.hero

(name,          family,    age, race,   class,  healthCur,  armourCur,  lead, effects) VALUES

('Immlerih',    'Codel',   24,  1,      9,      100,        0,          1,      ARRAY[]::appliedeffect[]);

SELECT \* FROM hero

**Создание индексов**

Индексы в PostgreSQL - это структуры данных, которые ускоряют поиск и сортировку записей в таблицах базы данных. Они создаются на одном или нескольких столбцах таблицы и позволяют оптимизировать выполнение запросов к базе данных.

В больших базах данных индексы являются необходимостью. С их помощью можно кратно сократить время запроса.

Были выбраны 2 таблицы для создания индекса: Оружие и броня. Их будет очень большое количество и создавать индексы там – целесообразно.

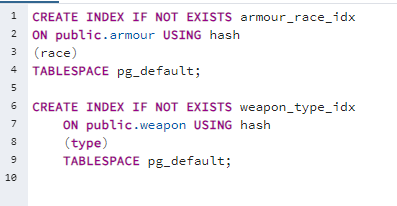
****

Рисунок 19 - Создание индексов

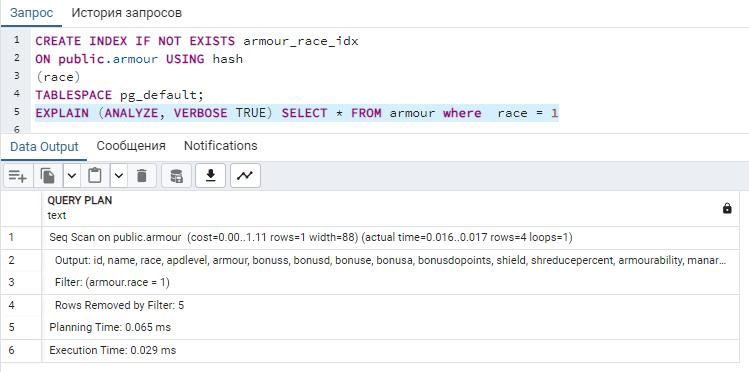


Рисунок 20 - Проверка первого индекса

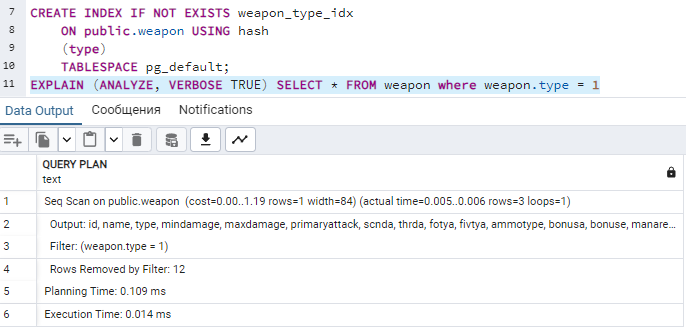


Рисунок 21 - Проверка второго индекса

Из тестов можно заметить, что фактическое время выполнения кратно меньше, чем плановое время выполнения, что положительно сказывается на работоспособности БД. Нет смысла создавать множество индексов на одну таблицу. От этого скорость выполнения может не то что не вырасти, но и даже уменьшиться.

**Представления**

Были созданы 5 представление, позволяющих

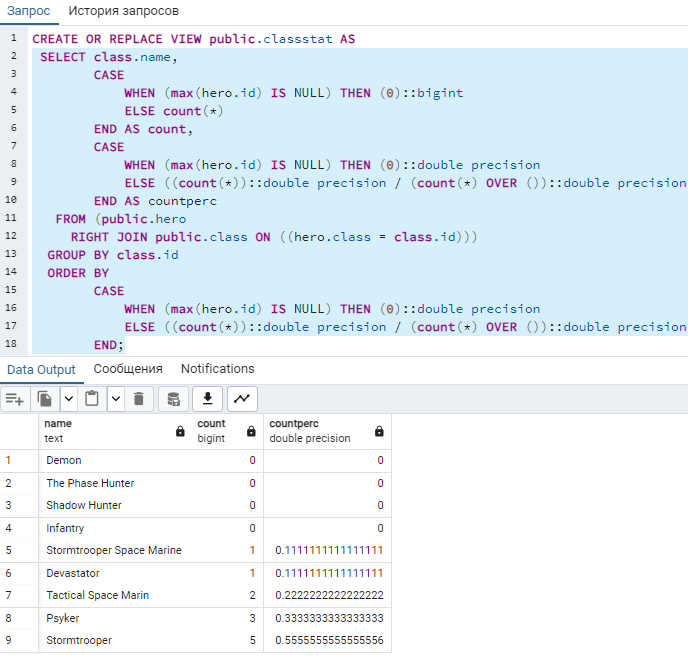


Рисунок 22 - Представление статистики классов

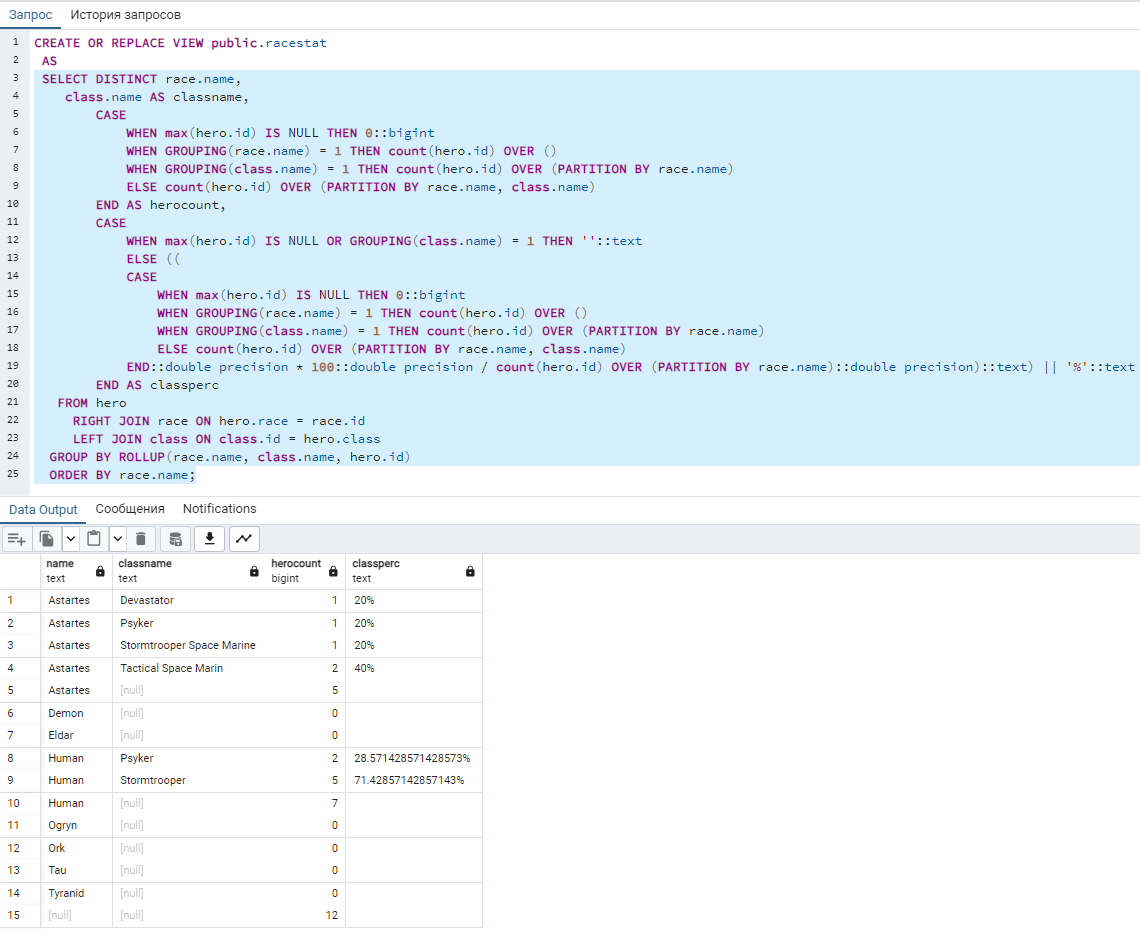


Рисунок 23- Представление статистики расы

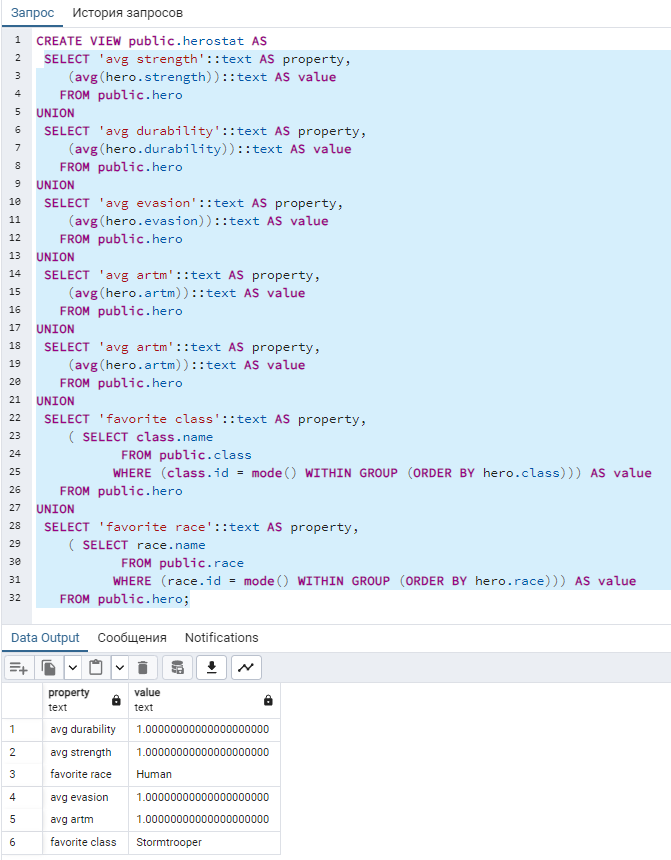


Рисунок 24 - Статистика пресонажей

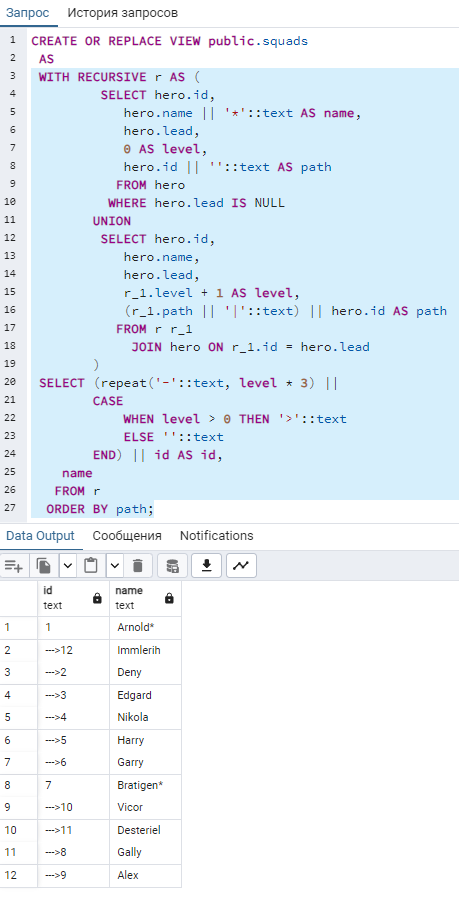


Рисунок 25 - Представление "отрядов"

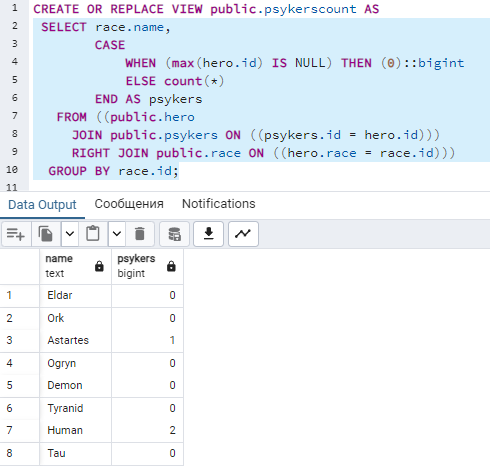


Рисунок 26 - Представление количества псайкеров среди героев