Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Курсовая работа этап 3

по «Информационным системам и базам данных»

<u>Выполнили</u>: Группа P33312 Верещагин Е. С.

Группа P33312 Хайкин О.И.

<u>Преподаватель</u>:

Наумова Н. А.

Санкт-Петербург, 2023

Описание предметной области

“Dungeons and Dragons” (далее D&D, DnD, ДНД, ДнД) - ролевая настольная игра, в которой игроки погружаются в вымышленный мир, управляя созданными ими персонажами. Каждой игрой управляет ведущий, описывающий окружающий игроков мир и происходящие с ними события.

D&D полна различных игровых механик, относящихся к тому, способен ли персонаж сделать то или иное действие, которое игрок хочет. Отдельного внимания заслуживают правила сражения в игре.

Для ведущего игры составляет важность возможность следить за происходящим в битве, быстро ориентироваться в том, чей ход идёт за кем и как изменились характеристики персонажей игроков и персонажей ведущего.

Для этой задачи на просторах интернета встречаются приложения-инструменты, называемые “Трекером инициативы“. Реализация подобного инструмента и является темой данной курсовой работы

Игра в D&D использует набор игровых костей с различным числом граней (от четырёхгранника, называемого d4 до сто-гранника, называемого d100)

Одной из основных сущностей проекта является “Character” - игровой персонаж. Им может быть как герой-паладин, которым управляет один из игроков, так и злобный гоблин, которым управляет ведущий (так называемым “Not Playable Character/ Не Игровой Персонаж”, далее NPC/НПС/НИП).

Любой персонаж обладает своим набором различных характеристик - от значения его силы до значения его способности убеждать. Для отражения характеристик в рамках проекта, выделим 2 сущности: “Stat block” и “Current Stats”.

Stat block (блок характеристик) отображает своего рода шаблон для характеристик персонажа. Current Stats (текущие характеристики) же отражают текущие характеристики существа.

Это разделение обусловлено тем, что у нескольких персонажей может быть один и тот же блок характеристик (представьте себе отряд идентичных гоблинов), но в то же время разные текущие характеристики (одного гоблина ударили кирпичом по голове, а другого нет).

Кроме того, существование блока характеристик как отдельной сущности сильно облегчит работу ведущего, т.к. Они работают как шаблон персонажа - чтобы добавить в битву противников. Ему достаточно просто указать шаблон для создания нового персонажа.

Для выражения характеристик персонажа выделяются:

“Обыкновенные” характеристики - такие как количество хитов (жизней), класс доспеха (сложность попадания по персонажу), скорость и т.д.

Способность - основа всех механик D&D. Обычно используется 6 стандартных характеристик - Сила, Ловкость, Телосложение, Интеллект, Мудрость и Харизма. Эти характеристики используются повсеместно, а их значения являются, по сути, основной частью персонажа.

Навык - навык это, своего рода, “применённая” способность. Модификатор навыка зависит от значения связанной с ним способности. Это легче понять на

примере: Навык “Убеждение” зависит от значения “Харизмы” у персонажа, а навык “Акробатика” от “Ловкости”.

Помимо определяющих персонажа характеристик, существует ряд других немаловажных механик:

Тип урона - любая атака в игре обладает своим типом (колющая, режущая, дробящая, огонь, холод, некротический и т.д.) Персонажи могут обладать иммунитетом, сопротивлением или уязвимостью к каким-либо типам урона, что влияет на то, как хорошо они могут выстоять атаки по себе.

Состояние - любой персонаж может по ходу битвы оказать в каком-то состоянии, отличном от обычного (опутанный, оглушённый, лежащий ничком, истощённый и т.д.). Состояния имеют свои эффекты, которые влияют на возможности персонажей во время игры.

Наконец, опишем сам процесс битвы. Битва состоит из раундов, а каждый раунд состоит из ходов персонажей.

В начале битвы все участвующие в ней персонажи совершают бросок инициативы, который определит порядок их ходов в раунде (персонажи, которые получили значение выше, идут первее тех, у которых получилось ниже).

Ведущему нужно всегда следить за тем, чей ход происходит сейчас, чей ход следующий, и какой раунд боя по счёту происходит сейчас. Именно для этого и предназначается Трекер Инициативы.

Реализация этапа 3 курсовой работы

Скрипт для создания базы данных

```
CREATE TABLE dice_type (
 side_count int4 NOT NULL,
 CONSTRAINT dice_type_pk PRIMARY KEY (side_count),
 CONSTRAINT side_count CHECK (
   (
     side_count > 0
   )
 )
);
CREATE TABLE creature_type (
 creature_type_id serial8 NOT NULL,
 creature_type_name varchar(100) NOT NULL,
 CONSTRAINT creature_type_pk PRIMARY KEY (creature_type_id),
 CONSTRAINT creature_type_un UNIQUE (creature_type_name),
 CONSTRAINT name_check CHECK (
   (
     (creature_type_name):: text <> '' :: text
   )
 )
```

```
);
CREATE TABLE ability (
 ability_id serial8 NOT NULL,
 ability_name varchar(20) NOT NULL,
 CONSTRAINT ability_pk PRIMARY KEY (ability_id),
 CONSTRAINT ability_un UNIQUE (ability_name),
 CONSTRAINT name_check CHECK (
   (
     (ability_name):: text <> '' :: text
   )
 )
);
CREATE TABLE skill (
 skill_id serial8 NOT NULL,
 skill_name varchar(100) NOT NULL,
 ability_id int8 NOT NULL,
 CONSTRAINT skill_pk PRIMARY KEY (skill_id),
 CONSTRAINT skill_un UNIQUE (skill_name),
 CONSTRAINT skill_fk FOREIGN KEY (ability_id) REFERENCES
ability(ability_id),
 CONSTRAINT name_check CHECK (
   (
     (skill_name):: text <> '' :: text
   )
 )
);
CREATE TABLE damage_type (
 damage_type_id serial8 NOT NULL,
 damage_type_name varchar(100) NOT NULL,
 CONSTRAINT damage_type_pk PRIMARY KEY (damage_type_id),
 CONSTRAINT damage_type_un UNIQUE (damage_type_name),
 CONSTRAINT name_check CHECK (
   (
     (damage_type_name):: text <> '' :: text
   )
 )
);
CREATE TABLE stat_block (
 stat_block_id serial8 NOT NULL,
 entity_name varchar(20) NOT NULL,
 hit_points int4 NOT NULL,
 hit_dice_type int4 NULL,
 hit_dice_count int4 NULL,
 armor_class int4 NOT NULL,
 speed int4 NOT NULL,
 level int4 NULL,
 creature_type_id int8 NOT NULL,
 CONSTRAINT stat_block_pk PRIMARY KEY (stat_block_id),
```

```
 CONSTRAINT stat_block_fk FOREIGN KEY (hit_dice_type) REFERENCES
dice_type(side_count),
 CONSTRAINT stat_block_fk1 FOREIGN KEY (creature_type_id) REFERENCES
creature_type(creature_type_id),
 CONSTRAINT name_check CHECK (
   (
     (entity_name):: text <> '' :: text
   )
 ),
 CONSTRAINT hit_points_check CHECK (
   (
     hit_points > 0
   )
 ),
 CONSTRAINT hit_dice_count_check CHECK (
   (
     hit_dice_count > 0
   )
 ),
 CONSTRAINT armor_class_check CHECK (
   (
     armor_class > 0
   )
 ),
 CONSTRAINT level_check CHECK (
   (
     level >= 0
   )
 )
);
CREATE TABLE ability_scores (
 stat_block_id int8 NOT NULL,
 ability_id int8 NOT NULL,
 score int4 NOT NULL,
 CONSTRAINT ability_scores_pk PRIMARY KEY (stat_block_id, ability_id),
 CONSTRAINT ability_scores_fk FOREIGN KEY (stat_block_id) REFERENCES
stat_block(stat_block_id),
 CONSTRAINT ability_scores_fk1 FOREIGN KEY (ability_id) REFERENCES
ability(ability_id),
 CONSTRAINT score_check CHECK (
   (
     score >= 1
   )
   AND (
     score <= 30
   )
 )
);
```

```
CREATE TABLE proficient_skills (
 stat_block_id int8 NOT NULL,
 skill_id int8 NOT NULL,
 CONSTRAINT proficient_skills_pk PRIMARY KEY (stat_block_id, skill_id),
 CONSTRAINT proficient_skills_fk FOREIGN KEY (stat_block_id) REFERENCES
stat_block(stat_block_id),
 CONSTRAINT proficient_skills_fk1 FOREIGN KEY (skill_id) REFERENCES
skill(skill_id)
);
CREATE TABLE damage_type_modifiers (
 stat_block_id int8 NOT NULL,
 damage_type_id int8 NOT NULL,
 modifier float4 NOT NULL,
 CONSTRAINT damage_type_modifiers_pk PRIMARY KEY (stat_block_id,
damage_type_id),
 CONSTRAINT damage_type_modifiers_fk FOREIGN KEY (stat_block_id) REFERENCES
stat_block(stat_block_id),
 CONSTRAINT damage_type_modifiers_fk1 FOREIGN KEY (damage_type_id)
REFERENCES damage_type(damage_type_id),
 CONSTRAINT modifier_check CHECK (
   (
     modifier >= 0
   ) AND (
     modifier <= 2
   )
 )
);
-----------------------------------------------------------------
CREATE TABLE player (
 player_id serial8 NOT NULL,
 player_name varchar(20),
 CONSTRAINT player_pk PRIMARY KEY (player_id),
 CONSTRAINT name_check CHECK (
   (
     (player_name):: text <> '' :: text
   )
 )
);
CREATE TABLE character (
 character_id serial8 NOT NULL,
 player_id int8,
 stat_block_id int8,
 CONSTRAINT character_pk PRIMARY KEY (character_id),
 CONSTRAINT character_fk FOREIGN KEY (stat_block_id) REFERENCES
stat_block(stat_block_id),
 CONSTRAINT character_fk1 FOREIGN KEY (player_id) REFERENCES
player(player_id)
);
```

```
CREATE TABLE current_stats (
 current_stats_id serial8 NOT NULL,
 character_id int8 NOT NULL,
 current_hit_points int4 NULL,
 temporary_hit_points int4 NOT NULL DEFAULT 0,
 current_hit_dice_count int4 NULL,
 current_armor_class int4 NULL,
 current_speed int4 NULL,
 CONSTRAINT current_stats_pk PRIMARY KEY (current_stats_id),
 CONSTRAINT current_stats_fk FOREIGN KEY (character_id) REFERENCES
character(character_id),
 CONSTRAINT temporary_hit_points_check CHECK (
   (
     temporary_hit_points >= 0
   )
 )
);
CREATE TABLE condition (
 condition_id serial8 NOT NULL,
 condition_name varchar(20) NOT NULL,
 CONSTRAINT condition_pk PRIMARY KEY (condition_id),
 CONSTRAINT condition_un UNIQUE (condition_name),
 CONSTRAINT name_check CHECK (
   (
     (condition_name):: text <> '' :: text
   )
 )
);
CREATE TABLE current_conditions (
 current_stats_id int8 NOT NULL,
 condition_id int8 NOT NULL,
 CONSTRAINT current_conditions_pk PRIMARY KEY (current_stats_id,
condition_id),
 CONSTRAINT current_conditions_fk FOREIGN KEY (current_stats_id) REFERENCES
current_stats(current_stats_id) ON DELETE CASCADE,
 CONSTRAINT current_conditions_fk1 FOREIGN KEY (condition_id) REFERENCES
condition(condition_id)
);
-----------------------------------------------------------------
CREATE TABLE battle (
 battle_id serial8 NOT NULL,
 round_number int4 NOT NULL,
 current_character_index int4 NOT NULL,
 CONSTRAINT battle_pk PRIMARY KEY (battle_id),
 CONSTRAINT round_number_check CHECK (
   (
     round_number > 0
   )
```

```
 ),
 CONSTRAINT current_character_index_check CHECK (
   (
     current_character_index > 0
   )
 )
);
CREATE TABLE initiative_entry (
 battle_id int8 NOT NULL,
 character_id int8 NOT NULL,
 initiative_roll int4,
 CONSTRAINT initiative_entry_pk PRIMARY KEY (battle_id, character_id),
 CONSTRAINT initiative_entry_fk FOREIGN KEY (battle_id) REFERENCES
battle(battle_id),
 CONSTRAINT initiative_entry_fk1 FOREIGN KEY (character_id) REFERENCES
character(character_id)
);

-----------------------------------------------------------------
CREATE INDEX entry_index ON initiative_entry USING btree (initiative_roll);

CREATE INDEX index_hit_points ON stat_block USING btree (hit_points);
CREATE INDEX index_hit_dice_count ON stat_block USING btree
(hit_dice_count);
CREATE INDEX index_armor_class ON stat_block USING btree (armor_class);
CREATE INDEX index_speed ON stat_block USING btree (speed);
CREATE INDEX index_level ON stat_block USING btree (level);

CREATE INDEX index_entity_name ON stat_block USING hash (entity_name);

-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_delete_stats()
RETURNS TRIGGER AS $$
BEGIN
   DELETE FROM current_stats cs
   WHERE cs.character_id = OLD.character_id;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER delete_stats_trigger
AFTER DELETE ON initiative_entry
FOR EACH ROW
EXECUTE FUNCTION trigger_delete_stats();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_check_character_index()
RETURNS TRIGGER AS $$
BEGIN
```

```
 IF NEW.current_character_index > (
   SELECT COUNT(*) FROM initiative_entry
   WHERE initiative_entry.battle_id = NEW.battle_id
 )
 THEN
   RAISE EXCEPTION 'иди нахуй';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_character_index
BEFORE UPDATE ON battle
FOR EACH ROW EXECUTE PROCEDURE trigger_check_character_index();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_check_current_stats_hp()
RETURNS TRIGGER AS $$
DECLARE
 max_hp int4;
BEGIN
 SELECT sb.hit_points FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = NEW.current_stats_id
 INTO max_hp;

 IF NEW.current_hit_points > max_hp
 THEN
   RAISE EXCEPTION 'Текущее ХП не может быть больше максимального!';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_current_stats_hp
BEFORE UPDATE ON current_stats
FOR EACH ROW EXECUTE PROCEDURE trigger_check_current_stats_hp();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_create_ability_scores()
RETURNS TRIGGER AS $$
DECLARE
BEGIN
 INSERT INTO ability_scores (ability_id, stat_block_id, score)
 SELECT ability_id, NEW.stat_block_id, 10 FROM ability;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER create_ability_scores
AFTER INSERT ON stat_block
FOR EACH ROW EXECUTE PROCEDURE trigger_create_ability_scores();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_check_ability_scores()
RETURNS TRIGGER AS $$
DECLARE
 ability_count int4;
 sb_id int8;
 stat_blocks int8[];
BEGIN
 SELECT COUNT(*) FROM ability
 INTO ability_count;

 SELECT ARRAY(
   SELECT sb.id FROM stat_block sb
 ) INTO stat_blocks;

 FOR EACH sb_id IN stat_blocks
 LOOP
   IF ability_count != (
     SELECT COUNT(*) FROM ability_scores abs
     WHERE abs.stat_block_id = sb_id
   )
   THEN
     RAISE EXCEPTION 'Для статблока должны существовать все значения всех
характеристик!';
   END IF;
 END LOOP;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_ability_scores
BEFORE DELETE ON ability_scores
FOR EACH STATEMENT EXECUTE PROCEDURE trigger_check_ability_scores();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_check_player_in_battle()
RETURNS TRIGGER AS $$
DECLARE
 maybe_player_id int8;
BEGIN
 SELECT player_id FROM character c
 WHERE c.character_id = NEW.character_id
 INTO maybe_player_id;

 IF maybe_player_id IS NOT NULL AND (SELECT EXISTS(SELECT * FROM
initiative_entry ie WHERE ie.character_id = NEW.character_id))
 THEN
```

```
   RAISE EXCEPTION 'Игрок не может учавствовать в нескольких битвах
одновременно!';
 END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_player_in_battle
BEFORE INSERT OR UPDATE ON initiative_entry
FOR EACH ROW EXECUTE PROCEDURE trigger_check_player_in_battle();
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trigger_check_current_stats_hit_dice()
RETURNS TRIGGER AS $$
DECLARE
 max_dice int4;
BEGIN
 SELECT sb.hit_dice_count FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = NEW.current_stats_id
 INTO max_dice;

 IF NEW.current_hit_dice_count > max_dice
 THEN
   RAISE EXCEPTION 'Текущее число костей хитов не может быть больше
максимального!';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_current_stats_hit_dice
BEFORE UPDATE ON current_stats
FOR EACH ROW EXECUTE PROCEDURE trigger_check_current_stats_hit_dice();
-----------------------------------------------------------------------
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION start_battle(char_ids int8[])
 RETURNS void
 LANGUAGE plpgsql AS
$func$
declare
 b_id int8;
 char_id int8;
 hit_points int4;
 hit_dice_count int4;
 armor_class int4;
 speed int4;
BEGIN
 INSERT INTO battle (round_number,current_character_index) VALUES
```

```
  (1,1) RETURNING battle_id INTO b_id;
 foreach char_id in array char_ids
 LOOP
   SELECT sb.hit_points, sb.hit_dice_count, sb.armor_class, sb.speed
   FROM stat_block sb
   JOIN character c ON c.stat_block_id = sb.stat_block_id
   WHERE c.character_id = char_id
   INTO hit_points, hit_dice_count, armor_class, speed;
   INSERT INTO current_stats
(current_hit_points,character_id,temporary_hit_points,current_hit_dice_count
,current_armor_class,current_speed) VALUES
     (hit_points,char_id,0,hit_dice_count,armor_class,speed);
   INSERT INTO initiative_entry (battle_id,character_id,initiative_roll)
VALUES
     (b_id,char_id,0);
 END LOOP;
END;
$func$;
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION next_initiative(b_id int8)
 RETURNS SETOF initiative_entry AS
$$
DECLARE
 max_char_index int4;
 new_current_char_index int4;
 new_current_round_num int4;
 char_index int4;
BEGIN
 SELECT COUNT(*) FROM initiative_entry ie
 WHERE ie.battle_id = b_id
 INTO max_char_index;
 SELECT current_character_index + 1, round_number
 FROM battle
 WHERE battle_id = b_id
 INTO new_current_char_index, new_current_round_num;

 IF new_current_char_index > max_char_index
 THEN
   new_current_char_index = 1;
   new_current_round_num = new_current_round_num + 1;
 END IF;
  UPDATE battle SET
   current_character_index = new_current_char_index,
   round_number = new_current_round_num
 WHERE battle_id = b_id
 RETURNING current_character_index INTO char_index;

 RETURN QUERY
```

```
 SELECT * FROM initiative_entry ie
 WHERE ie.battle_id = b_id
 ORDER BY ie.initiative_roll
 LIMIT 1
 OFFSET (char_index - 1);
END
$$ LANGUAGE plpgsql;
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION damage(damage int4, curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 temp_hp int4;
 temp_damage int4;
 actual_damage int4;
BEGIN
 SELECT cs.temporary_hit_points FROM current_stats cs
 WHERE cs.current_stats_id = curr_stats_id
 INTO temp_hp;

 temp_damage = LEAST(temp_hp, damage);
 actual_damage = GREATEST(0, damage - temp_hp);

 UPDATE current_stats SET
   temporary_hit_points = temporary_hit_points - temp_damage,
   current_hit_points = current_hit_points - actual_damage
 WHERE current_stats_id = curr_stats_id;
END;
$func$;
-----------------------------------------------------------------------
CREATE OR REPLACE FUNCTION damage_with_type(damage int4, dmg_tp_id int8,
curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 modifier float4;
 actual_damage int4;
BEGIN
 SELECT modifier FROM damage_type_modifiers dtm
 JOIN character c ON dtm.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON c.character_id = cs.character_id
 WHERE cs.current_stats_id = curr_stats_id
   AND dtm.damage_type_id = dmg_tp_id
 INTO modifier;

 IF modifier IS NULL
```

```
 THEN
   actual_damage = damage;
 ELSE
   actual_damage = FLOOR(damage * modifier);
 END IF;

 SELECT damage(actual_damage, curr_stats_id);
END;
$func$;
-------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION heal(amount int4, curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 max_hp int4;
BEGIN
 SELECT sb.hit_points FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = curr_stats_id
 INTO max_hp;

 UPDATE current_stats SET
   current_hit_points = LEAST(max_hp, current_hit_points + amount)
 WHERE current_stats_id = curr_stats_id;
END;
$func$;
-------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION roll_dice(side_count int4)
  RETURNS int4 AS
$$
BEGIN
  RETURN FLOOR(random()* (side_count - 1) + 1);
END;
$$ language 'plpgsql';
-------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION get_proficiency_bonus(char_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 level int4;
BEGIN
 SELECT sb.level FROM stat_block sb
 JOIN character c ON c.stat_block_id = sb.stat_block_id
 WHERE c.character_id = char_id
 INTO level;
```

```
 RETURN ((level-1) / 4) + 2;
END;
$func$;
-------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION roll_ability_check(char_id int8, abil_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 ability_score int4;
 ability_bonus int4;
BEGIN
 SELECT a.score FROM ability_scores a
 JOIN character c ON c.stat_block_id = a.stat_block_id
 WHERE c.character_id = char_id
 AND a.ability_id = abil_id
 INTO ability_score;

 ability_bonus = (ability_score - 10) / 2;

 RETURN roll_dice(20) + ability_bonus;
END;
$func$;
-------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION roll_skill_check(char_id int8, sk_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 is_proficient boolean;
 roll int4;
BEGIN
 SELECT EXISTS (SELECT * FROM proficient_skills ps
 JOIN character c ON ps.stat_block_id = c.stat_block_id
 WHERE c.character_id = char_id
 AND ps.skill_id = sk_id)
 INTO is_proficient;

 roll = roll_ability_check(char_id, (
   SELECT s.ability_id FROM skill s
   WHERE s.skill_id = sk_id
 ));

 IF is_proficient
 THEN
   roll = roll + get_proficiency_bonus(char_id);
 END IF;
```

```
 RETURN roll;
END;
$func$;
```

Скрипт для удаления базы данных

```
DROP TABLE ability CASCADE;

DROP TABLE ability_scores CASCADE;

DROP TABLE battle CASCADE;

DROP TABLE player CASCADE;

DROP TABLE "character" CASCADE;

DROP TABLE "condition" CASCADE;

DROP TABLE creature_type CASCADE;

DROP TABLE current_conditions CASCADE;

DROP TABLE current_stats CASCADE;

DROP TABLE damage_type CASCADE;

DROP TABLE damage_type_modifiers CASCADE;

DROP TABLE dice_type CASCADE;

DROP TABLE initiative_entry CASCADE;

DROP TABLE proficient_skills CASCADE;

DROP TABLE skill CASCADE;

DROP TABLE stat_block CASCADE;
```

Скрипт для заполнения базы данных

```
INSERT INTO ability (ability_name) VALUES
    ('СИЛА'),
```

```
    ('ЛОВКОСТЬ'),
    ('ТЕЛОСЛОЖЕНИЕ'),
    ('ИНТЕЛЛЕКТ'),
    ('МУДРОСТЬ'),
    ('ХАРИЗМА');
INSERT INTO skill (skill_name,ability_id) VALUES
    ('Атлетика',1),
    ('Акробатика',2),
    ('Ловкость рук',2),
    ('Скрытность',2),
    ('Анализ',4),
    ('История',4),
    ('Магия',4),
    ('Природа',4),
    ('Религия',4),
    ('Внимательность',5),
    ('Выживание',5),
    ('Медицина',5),
    ('Проницательность',5),
    ('Уход за животными',5),
    ('Выступление',6),
    ('Запугивание',6),
    ('Обман',6),
    ('Убеждение',6);
INSERT INTO damage_type (damage_type_name) VALUES
    ('Кислота'),
    ('Дробящий'),
    ('Холод'),
    ('Огонь'),
    ('Силовое поле'),
    ('Электричество'),
    ('Некротический'),
    ('Колющий'),
    ('Яд'),
    ('Психический'),
    ('Излучение'),
    ('Режущий'),
    ('Звук');
INSERT INTO "condition" (condition_name) VALUES
    ('Бессознательный'),
    ('Испуганный'),
    ('Истощённый'),
    ('Невидимый'),
    ('Недееспособный'),
    ('Оглохший'),
    ('Окаменевший'),
    ('Опутанный'),
    ('Ослеплённый'),
```

```
    ('Отравленный'),
    ('Очарованный'),
    ('Ошеломлённый'),
    ('Парализованный'),
    ('Сбитый с ног'),
    ('Схваченный');
INSERT INTO creature_type (creature_type_name) VALUES
    ('Абберация'),
    ('Великан'),
    ('Гуманоид'),
    ('Дракон'),
    ('Зверь'),
    ('Исчадие'),
    ('Конструкт'),
    ('Монстр'),
    ('Небожитель'),
    ('Нежить'),
    ('Растение'),
    ('Слизь'),
    ('Фея'),
    ('Элементаль');
INSERT INTO dice_type (side_count) VALUES
    (4),
    (6),
    (8),
    (10),
    (12),
    (20);



INSERT INTO stat_block
(entity_name,hit_points,hit_dice_type,hit_dice_count,armor_class,speed,"leve
l",creature_type_id) VALUES
    ('Гоблин пси-драчун',31,NULL,NULL,15,30,2,1),
    ('Валл''акхад',36,8,4,18,30,4,3),
    ('Билли Бобёр',34,NULL,NULL,34,40,3,7);
INSERT INTO player (player_name) VALUES
    ('Олег');
INSERT INTO "character" (player_id,stat_block_id) VALUES
    (1,2),
    (NULL,1);
INSERT INTO current_stats
(current_hit_points,character_id,temporary_hit_points,current_hit_dice_count
,current_armor_class,current_speed) VALUES
    (1,2,0,NULL,NULL,NULL),
    (NULL,1,0,NULL,NULL,NULL);
```

```
INSERT INTO ability_scores (stat_block_id,ability_id,score) VALUES
    (1,1,9),
    (1,2,17),
    (1,3,12),
    (1,4,16),
    (1,5,15),
    (1,6,10),
    (2,1,16),
    (2,2,8),
    (2,3,14),
    (2,4,10);
INSERT INTO ability_scores (stat_block_id,ability_id,score) VALUES
    (2,5,14),
    (2,6,16),
    (3,1,20),
    (3,2,12),
    (3,3,17),
    (3,4,3),
    (3,5,12),
    (3,6,7);
INSERT INTO battle (round_number,current_character_index) VALUES
    (1,1);
INSERT INTO current_conditions (current_stats_id,condition_id) VALUES
    (1,2);
INSERT INTO damage_type_modifiers (stat_block_id,damage_type_id,modifier)
VALUES
    (1,10,0.5),
    (3,10,0.0),
    (3,9,0.0),
    (2,10,0.5);
INSERT INTO initiative_entry (battle_id,character_id,initiative_roll) VALUES
    (1,1,17),
    (1,2,13);
INSERT INTO proficient_skills (stat_block_id,skill_id) VALUES
    (1,4),
    (2,5),
    (2,9),
    (2,11),
    (2,12),
    (2,13),
    (2,18),
    (3,10);
```

Скрипт для очистки базы данных

```
alter sequence ability_ability_id_seq restart with 1;
truncate ability cascade;
truncate ability_scores cascade;
alter sequence battle_battle_id_seq restart with 1;
truncate battle cascade;
alter sequence character_character_id_seq restart with 1;
truncate "character" cascade;
alter sequence condition_condition_id_seq restart with 1;
truncate "condition" cascade;
alter sequence creature_type_creature_type_id_seq restart with 1;
truncate creature_type  cascade;
truncate current_conditions  cascade;
alter sequence current_stats_current_stats_id_seq restart with 1;
truncate current_stats  cascade;
alter sequence damage_type_damage_type_id_seq restart with 1;
truncate damage_type  cascade;
truncate damage_type_modifiers  cascade;
truncate dice_type  cascade;
truncate initiative_entry  cascade;
alter sequence player_player_id_seq restart with 1;
truncate player  cascade;
truncate proficient_skills  cascade;
alter sequence skill_skill_id_seq restart with 1;
truncate skill  cascade;
alter sequence stat_block_stat_block_id_seq restart with 1;
truncate stat_block  cascade;
```

Реализованные индексы

```
CREATE INDEX entry_index ON initiative_entry USING btree
(initiative_roll);

CREATE INDEX index_hit_points ON stat_block USING btree (hit_points);
CREATE INDEX index_hit_dice_count ON stat_block USING btree
(hit_dice_count);
CREATE INDEX index_armor_class ON stat_block USING btree (armor_class);
CREATE INDEX index_speed ON stat_block USING btree (speed);
CREATE INDEX index_level ON stat_block USING btree (level);

CREATE INDEX index_entity_name ON stat_block USING hash (entity_name);
```

На поля, для которых будут производиться операции с сортировкой(например, hit_points, dice_count, armor_class, speed, level) были накинуты Btree индексы. Для быстрого поиска по имени сущности в stat_block добавлен hash индекс на entity_name.

На поле iterative_roll таблицы initiative_entry также добавлен BTree индекс, так как порядок ходов в игре определяется значением этого атрибута.

Реализованные триггеры

trigger_delete_stats

```
CREATE OR REPLACE FUNCTION trigger_delete_stats()
RETURNS TRIGGER AS $$
BEGIN
   DELETE FROM current_stats cs
   WHERE cs.character_id = OLD.character_id;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER delete_stats_trigger
AFTER DELETE ON initiative_entry
FOR EACH ROW
EXECUTE FUNCTION trigger_delete_stats();
```

Триггер, который удаляет сущности CURRENT_STATS, когда персонаж “выходит” из инициативы сражения.

trigger_check_character_index

```
CREATE OR REPLACE FUNCTION trigger_check_character_index()
RETURNS TRIGGER AS $$
BEGIN
 IF NEW.current_character_index > (
   SELECT COUNT(*) FROM initiative_entry
   WHERE initiative_entry.battle_id = NEW.battle_id
 )
 THEN
   RAISE EXCEPTION 'error';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_character_index
BEFORE UPDATE ON battle
FOR EACH ROW EXECUTE PROCEDURE trigger_check_character_index();
```

Триггер, который удостоверяется, что индекс хода персонажа в битве не станет больше, чем число персонажей в битве.

trigger_check_current_stats_hp

```
CREATE OR REPLACE FUNCTION trigger_check_current_stats_hp()
RETURNS TRIGGER AS $$
DECLARE
 max_hp int4;
BEGIN
 SELECT sb.hit_points FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = NEW.current_stats_id
 INTO max_hp;

 IF NEW.current_hit_points > max_hp
 THEN
   RAISE EXCEPTION 'Текущее ХП не может быть больше максимального!';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_current_stats_hp
BEFORE UPDATE ON current_stats
FOR EACH ROW EXECUTE PROCEDURE trigger_check_current_stats_hp();
```

Триггер, который удостоверяется, что число хитов персонажа не превысит его максимального.

trigger_create_ability_scores

```
CREATE OR REPLACE FUNCTION trigger_create_ability_scores()
RETURNS TRIGGER AS $$
DECLARE
BEGIN
 INSERT INTO ability_scores (ability_id, stat_block_id, score)
 SELECT ability_id, NEW.stat_block_id, 10 FROM ability;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER create_ability_scores
AFTER INSERT ON stat_block
FOR EACH ROW EXECUTE PROCEDURE trigger_create_ability_scores();
```

Триггер, который автоматически создаёт записи для характеристик, при создании статблока.

trigger_check_ability_scores

```
CREATE OR REPLACE FUNCTION trigger_check_ability_scores()
RETURNS TRIGGER AS $$
DECLARE
 ability_count int4;
 sb_id int8;
 stat_blocks int8[];
BEGIN
 SELECT COUNT(*) FROM ability
 INTO ability_count;

 SELECT ARRAY(
   SELECT sb.id FROM stat_block sb
 ) INTO stat_blocks;

 FOR EACH sb_id IN stat_blocks
 LOOP
   IF ability_count != (
     SELECT COUNT(*) FROM ability_scores abs
     WHERE abs.stat_block_id = sb_id
   )
   THEN
     RAISE EXCEPTION 'Для статблока должны существовать все значения всех
характеристик!';
   END IF;
 END LOOP;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_ability_scores
BEFORE DELETE ON ability_scores
FOR EACH STATEMENT EXECUTE PROCEDURE trigger_check_ability_scores();
```

Триггер, который проверяет, что для статблока существуют все значения характеристик.

trigger_check_player_in_battle

```
CREATE OR REPLACE FUNCTION trigger_check_player_in_battle()
RETURNS TRIGGER AS $$
DECLARE
 maybe_player_id int8;
BEGIN
 SELECT player_id FROM character c
 WHERE c.character_id = NEW.character_id
 INTO maybe_player_id;
```

```
 IF maybe_player_id IS NOT NULL AND (SELECT EXISTS(SELECT * FROM
initiative_entry ie WHERE ie.character_id = NEW.character_id))
 THEN
   RAISE EXCEPTION 'Игрок не может участвовать в нескольких битвах
одновременно!';
 END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_player_in_battle
BEFORE INSERT OR UPDATE ON initiative_entry
FOR EACH ROW EXECUTE PROCEDURE trigger_check_player_in_battle();
```

Триггер, который проверяет, что игрок (не существо ведущего) не окажется в нескольких битвах одновременно

trigger_check_current_stats_hit_dice

```
CREATE OR REPLACE FUNCTION trigger_check_current_stats_hit_dice()
RETURNS TRIGGER AS $$
DECLARE
 max_dice int4;
BEGIN
 SELECT sb.hit_dice_count FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = NEW.current_stats_id
 INTO max_dice;

 IF NEW.current_hit_dice_count > max_dice
 THEN
   RAISE EXCEPTION 'Текущее число костей хитов не может быть больше
максимального!';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_current_stats_hit_dice
BEFORE UPDATE ON current_stats
FOR EACH ROW EXECUTE PROCEDURE trigger_check_current_stats_hit_dice();
```

Триггер, который удостоверяется, что число костей хитов персонажа не превысит его максимального.

Реализованные бизнес-функции

start_battle

```
CREATE OR REPLACE FUNCTION start_battle(char_ids int8[])
 RETURNS void
 LANGUAGE plpgsql AS
$func$
declare
 b_id int8;
 char_id int8;
 hit_points int4;
 hit_dice_count int4;
 armor_class int4;
 speed int4;
BEGIN
 INSERT INTO battle (round_number,current_character_index) VALUES
  (1,1) RETURNING battle_id INTO b_id;
 foreach char_id in array char_ids
 LOOP
   SELECT sb.hit_points, sb.hit_dice_count, sb.armor_class, sb.speed
   FROM stat_block sb
   JOIN character c ON c.stat_block_id = sb.stat_block_id
   WHERE c.character_id = char_id
   INTO hit_points, hit_dice_count, armor_class, speed;
   INSERT INTO current_stats
(current_hit_points,character_id,temporary_hit_points,current_hit_dice_count
,current_armor_class,current_speed) VALUES
     (hit_points,char_id,0,hit_dice_count,armor_class,speed);
   INSERT INTO initiative_entry (battle_id,character_id,initiative_roll)
VALUES
     (b_id,char_id,0);
 END LOOP;
END;
$func$;
```

Функция начала сражения - при исполнении создаст записи о порядке инициативы и текущих характеристиках персонажа.

next_initiative

```
CREATE OR REPLACE FUNCTION next_initiative(b_id int8)
 RETURNS SETOF initiative_entry AS
$$
DECLARE
 max_char_index int4;
 new_current_char_index int4;
```

```
 new_current_round_num int4;
 char_index int4;
BEGIN
 SELECT COUNT(*) FROM initiative_entry ie
 WHERE ie.battle_id = b_id
 INTO max_char_index;
 SELECT current_character_index + 1, round_number
 FROM battle
 WHERE battle_id = b_id
 INTO new_current_char_index, new_current_round_num;

 IF new_current_char_index > max_char_index
 THEN
   new_current_char_index = 1;
   new_current_round_num = new_current_round_num + 1;
 END IF;
  UPDATE battle SET
   current_character_index = new_current_char_index,
   round_number = new_current_round_num
 WHERE battle_id = b_id
 RETURNING current_character_index INTO char_index;

 RETURN QUERY
 SELECT * FROM initiative_entry ie
 WHERE ie.battle_id = b_id
 ORDER BY ie.initiative_roll
 LIMIT 1
 OFFSET (char_index - 1);
END
$$ LANGUAGE plpgsql;
```

Функция выборки персонажа, находящегося в инициативе следующим. Обновляет информацию о текущем индексе персонажа и номере раунда при необходимости.

damage

```
CREATE OR REPLACE FUNCTION damage(damage int4, curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 temp_hp int4;
 temp_damage int4;
 actual_damage int4;
BEGIN
 SELECT cs.temporary_hit_points FROM current_stats cs
 WHERE cs.current_stats_id = curr_stats_id
```

```
 INTO temp_hp;

 temp_damage = LEAST(temp_hp, damage);
 actual_damage = GREATEST(0, damage - temp_hp);

 UPDATE current_stats SET
   temporary_hit_points = temporary_hit_points - temp_damage,
   current_hit_points = current_hit_points - actual_damage
 WHERE current_stats_id = curr_stats_id;
END;
$func$;
```

Функция нанесения урона персонажу в сражении. Учитывает временные хиты персонажа.

damage_with_type

```
CREATE OR REPLACE FUNCTION damage_with_type(damage int4, dmg_tp_id int8,
curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 modifier float4;
 actual_damage int4;
BEGIN
 SELECT modifier FROM damage_type_modifiers dtm
 JOIN character c ON dtm.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON c.character_id = cs.character_id
 WHERE cs.current_stats_id = curr_stats_id
   AND dtm.damage_type_id = dmg_tp_id
 INTO modifier;

 IF modifier IS NULL
 THEN
   actual_damage = damage;
 ELSE
   actual_damage = FLOOR(damage * modifier);
 END IF;

 SELECT damage(actual_damage, curr_stats_id);
END;
$func$;
```

Функция нанесения урона персонажу в сражении с учётом его модификаторов в зависимости от типа урона.

heal

```
CREATE OR REPLACE FUNCTION heal(amount int4, curr_stats_id int8)
 RETURNS void
 LANGUAGE plpgsql AS
$func$
DECLARE
 max_hp int4;
BEGIN
 SELECT sb.hit_points FROM stat_block sb
 JOIN character c ON sb.stat_block_id = c.stat_block_id
 JOIN current_stats cs ON cs.character_id = c.character_id
 WHERE cs.current_stats_id = curr_stats_id
 INTO max_hp;

 UPDATE current_stats SET
   current_hit_points = LEAST(max_hp, current_hit_points + amount)
 WHERE current_stats_id = curr_stats_id;
END;
$func$;
```

Функция лечения персонажа. Учитывает максимальное количество хитов персонажа.

roll_dice

```
CREATE OR REPLACE FUNCTION roll_dice(side_count int4)
  RETURNS int4 AS
$$
BEGIN
  RETURN FLOOR(random()* (side_count - 1) + 1);
END;
$$ language 'plpgsql';
```

Функция, симулирующая бросок кубика.

get_proficiency_bonus

```
CREATE OR REPLACE FUNCTION get_proficiency_bonus(char_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 level int4;
BEGIN
 SELECT sb.level FROM stat_block sb
 JOIN character c ON c.stat_block_id = sb.stat_block_id
```

```
 WHERE c.character_id = char_id
 INTO level;

 RETURN ((level-1) / 4) + 2;
END;
$func$;
```

Функция, вычисляющая бонус мастерства персонажа.

roll_ability_check

```
CREATE OR REPLACE FUNCTION roll_ability_check(char_id int8, abil_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 ability_score int4;
 ability_bonus int4;
BEGIN
 SELECT a.score FROM ability_scores a
 JOIN character c ON c.stat_block_id = a.stat_block_id
 WHERE c.character_id = char_id
 AND a.ability_id = abil_id
 INTO ability_score;

 ability_bonus = (ability_score - 10) / 2;

 RETURN roll_dice(20) + ability_bonus;
END;
$func$;
```

Функция, выполняющая проверку характеристики персонажа.

roll_skill_check

```
CREATE OR REPLACE FUNCTION roll_skill_check(char_id int8, sk_id int8)
 RETURNS int4
 LANGUAGE plpgsql AS
$func$
DECLARE
 is_proficient boolean;
 roll int4;
BEGIN
 SELECT EXISTS (SELECT * FROM proficient_skills ps
 JOIN character c ON ps.stat_block_id = c.stat_block_id
 WHERE c.character_id = char_id
 AND ps.skill_id = sk_id)
 INTO is_proficient;
```

```
 roll = roll_ability_check(char_id, (
   SELECT s.ability_id FROM skill s
   WHERE s.skill_id = sk_id
 ));

 IF is_proficient
 THEN
   roll = roll + get_proficiency_bonus(char_id);
 END IF;
 RETURN roll;
END;
$func$;
```

Функция, выполняющая проверку навыка персонажа.