Zadanie 3: Implementácia dátového modelu v PostgreSQL

Obsah

1. Logicko-Fyzické mapovanie

Priraďuje logickému návrhu databázy už konkrétnu fyzickú podobu

1.1 spell_category

LOG: Kategória ktorá určovala typ efektu pre útoky.

FYZ:

```
CREATE TABLE spell_category (
    category_ID SERIAL PRIMARY KEY,
    category_name TEXT
    CHECK (category_name IN ('SLASHING', 'PIERCING', 'BLUNT', 'FIRE','WATER', 'ICE', 'LIGHTNING', 'HEALING', 'EARTH'))
);
```

1.2 class

LOG: Trieda pre postavu

FYZ:

```
CREATE TABLE class (
    class_ID SERIAL PRIMARY KEY,
    class_name TEXT,
    base_dex INT,
    base_int INT,
    base_stg INT,
    base_cos INT,
    base_hlt INT,
    base_reg INT,
    inv_factor NUMERIC(10,2),
    ap_factor NUMERIC(10,2),
    ac_factor NUMERIC(10,2)
);
```

1.3 shop

LOG: Obchod na nakupovanie vecí a kúziel

FYZ:

```
CREATE TABLE shop (
    shop_ID SERIAL PRIMARY KEY,
    shop_name TEXT,
    profit_margin NUMERIC(10,2)
);
```

1.4 item

LOG: vec na zbieranie, predávanie a zlepšovanie atribútov

FYZ:

```
CREATE TABLE item (
    item_ID SERIAL PRIMARY KEY,
    item_name TEXT,
    item_weight NUMERIC(10,2),
    eff_category INT REFERENCES spell_category( category_ID),
    eff_factor NUMERIC(10,2),
    cost_factor NUMERIC(10,2),
    rarity NUMERIC(10,2),
    sell_cost INT,
    pref_class INT REFERENCES class( class_ID)
);
```

1.5 spell

LOG: kúzlo na útok alebo vyliečenie

FYZ:

```
CREATE TABLE spell (
    spell_ID SERIAL PRIMARY KEY,
    spell_name TEXT,
    eff_category INT REFERENCES spell_category(category_ID),
    base NUMERIC(10,2),
    is_aoe BOOLEAN,
    atribute_to_use TEXT CHECK (atribute_to_use IN ('dex', 'stg', 'int', 'cos'))
);
```

1.6 character

LOG: Hlavná tabuľka pre hráča, symbolizuje jeho atribúty

FYZ:

```
CREATE TABLE character (
    character_ID SERIAL PRIMARY KEY,
    class_ID INT REFERENCES class(class_ID),
    shop_ID INT REFERENCES shop(shop_ID) DEFAULT NULL,
    item_equipped INT REFERENCES item(item_ID) DEFAULT NULL,
    off_hand_item INT REFERENCES item(item_ID) DEFAULT NULL,
    character_name TEXT,
    action_points INT,
    armor_class INT,
    health INT,
    max_health INT,
    strength INT,
    dexterity INT,
    intelligence INT,
    constitution INT,
    regeneration INT,
    max_cap_inv NUMERIC(10,2),
    curr_inv_state NUMERIC(10,2) DEFAULT 0,
    head_bounty INT DEFAULT 10,
    money_bag INT DEFAULT 100
);
```

1.7 combat

LOG: Zápis pre boj

FYZ:

```
CREATE TABLE combat (
    combat_ID SERIAL PRIMARY KEY,
    is_active BOOLEAN,
    combat_name TEXT
);
```

1.8 rounds

LOG: Zápis pre kolo

FYZ:

```
CREATE TABLE rounds (
    round_ID SERIAL PRIMARY KEY,
    combat_ID INT REFERENCES combat(combat_ID),
    round_num INT
);
```

1.9 spell_inventory

LOG: Spojovacia tabuľka pre hráča a kúzlo

FYZ:

```
CREATE TABLE spell_inventory (
    spell_inv_ID SERIAL PRIMARY KEY,
    character_ID INT NOT NULL,
    is_shop BOOLEAN DEFAULT FALSE,
    spell_ID INT REFERENCES spell(spell_ID)
);
```

1.10 actions

LOG: Záznam akcií čo sa stali počas boja

FYZ:

```
CREATE TABLE actions (
    action_ID SERIAL PRIMARY KEY,
    combat_ID INT REFERENCES combat(combat_ID),
    round_ID INT REFERENCES rounds(round_ID),
    target INT DEFAULT NULL,
    actor INT REFERENCES character(character_ID),
    used_spell INT REFERENCES spell(spell_ID) DEFAULT NULL,
    action_type TEXT CHECK( action_type in ('flee', 'cast', 'pickup', 'pursue', 'switch', 'drop', 'end', 'join')),
    action_num INT,
    is_success BOOLEAN,
    eff_dealt INT DEFAULT NULL,
    ap_cost INT DEFAULT NULL,
    dice_roll INT CHECK(dice_roll <= 20 AND dice_roll > 0) DEFAULT NULL,
    time_stamp TIMESTAMP DEFAULT NOW() -- add to the doku lebo inak bitka!
);
```

1.11 character_combat_state

LOG: zálohovacia tabuľka pre hráča počas boja

FYZ:

```
CREATE TABLE character_combat_state (
    character_state_ID SERIAL PRIMARY KEY,
    character_ID INT REFERENCES character(character_ID),
    item_equipped INT REFERENCES item(item_ID),
    off_hand_item INT REFERENCES item(item_ID),
    combat_ID INT REFERENCES combat(combat_ID),
    round_ID INT REFERENCES rounds(round_ID),
    action_points INT,
    health INT,
    head_bounty INT,
    money_bag INT
);
```

1.12 inventory

LOG: Spojovacia tabuľka pre hráča a nástroj

FYZ:

```
CREATE TABLE inventory (
    inventory_ID SERIAL PRIMARY KEY,
    owner_ID INT NOT NULL,
    inv_description TEXT CHECK( inv_description in ('shp', 'ply', 'cmb')),
    item_ID INT REFERENCES item(item_ID)
);
```

1.13 inventory_state

LOG: zálohovacia tabuľka pre stav inventára počas boja

FYZ:

```
CREATE TABLE inventory_state (
    inventory_state_ID SERIAL PRIMARY KEY,
    combat_ID INT REFERENCES combat(combat_ID),
    round_ID INT REFERENCES rounds(round_ID),
    owner_ID INT NOT NULL,
    inv_description TEXT CHECK( inv_description in ('shp', 'ply', 'cmb')),
    item_ID INT REFERENCES item(item_ID)
);
```

1.14 spell_state

LOG: zálohovacia tabuľka pre stav kúziel počas boja

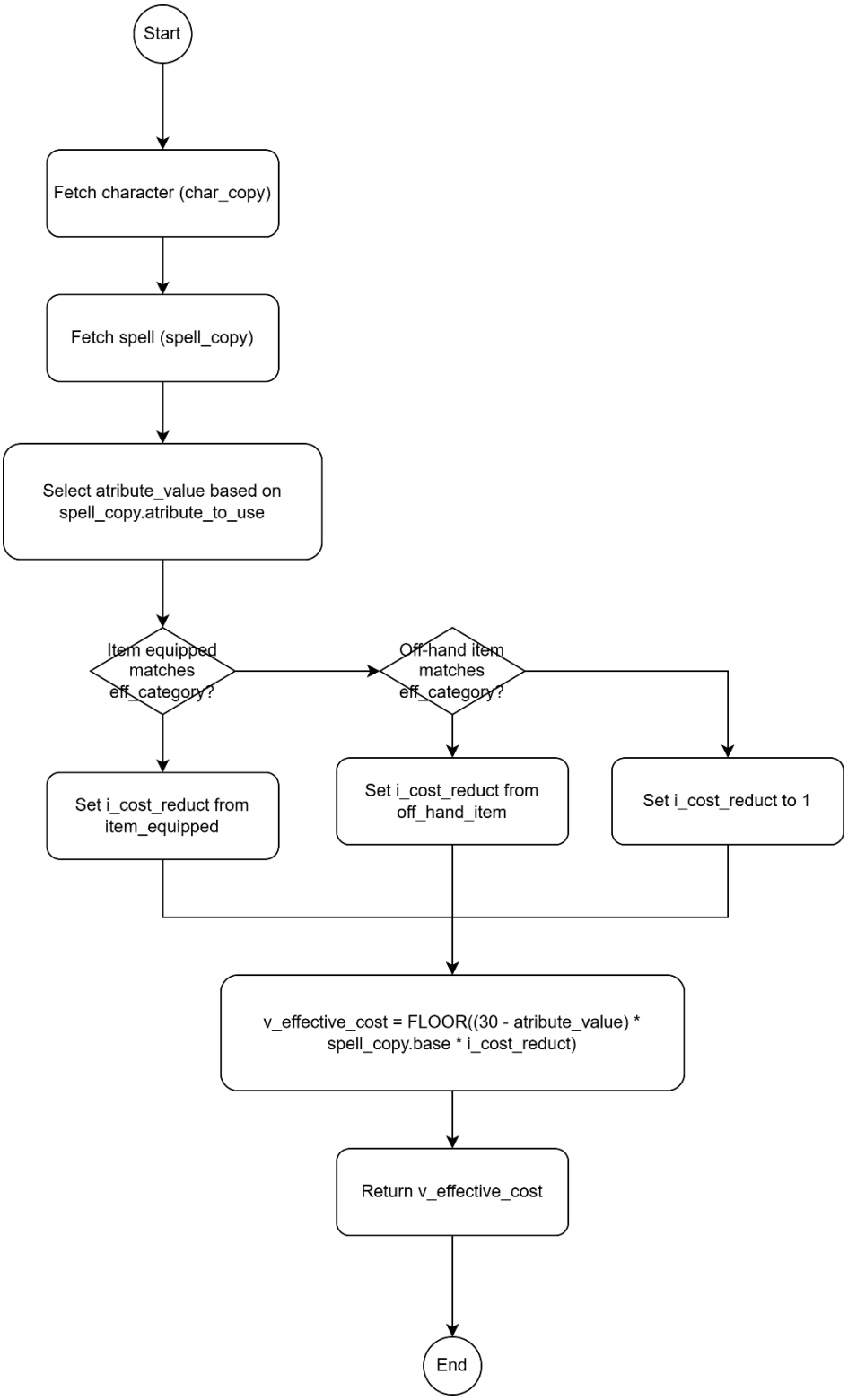
FYZ:

```
CREATE TABLE spell_state (
    spell_state_ID SERIAL PRIMARY KEY,
    character_ID INT NOT NULL,
    spell_ID INT REFERENCES spell(spell_ID),
    combat_ID INT REFERENCES combat(combat_ID),
    round_ID INT REFERENCES rounds(round_ID)
);
```

2. Procesné toky

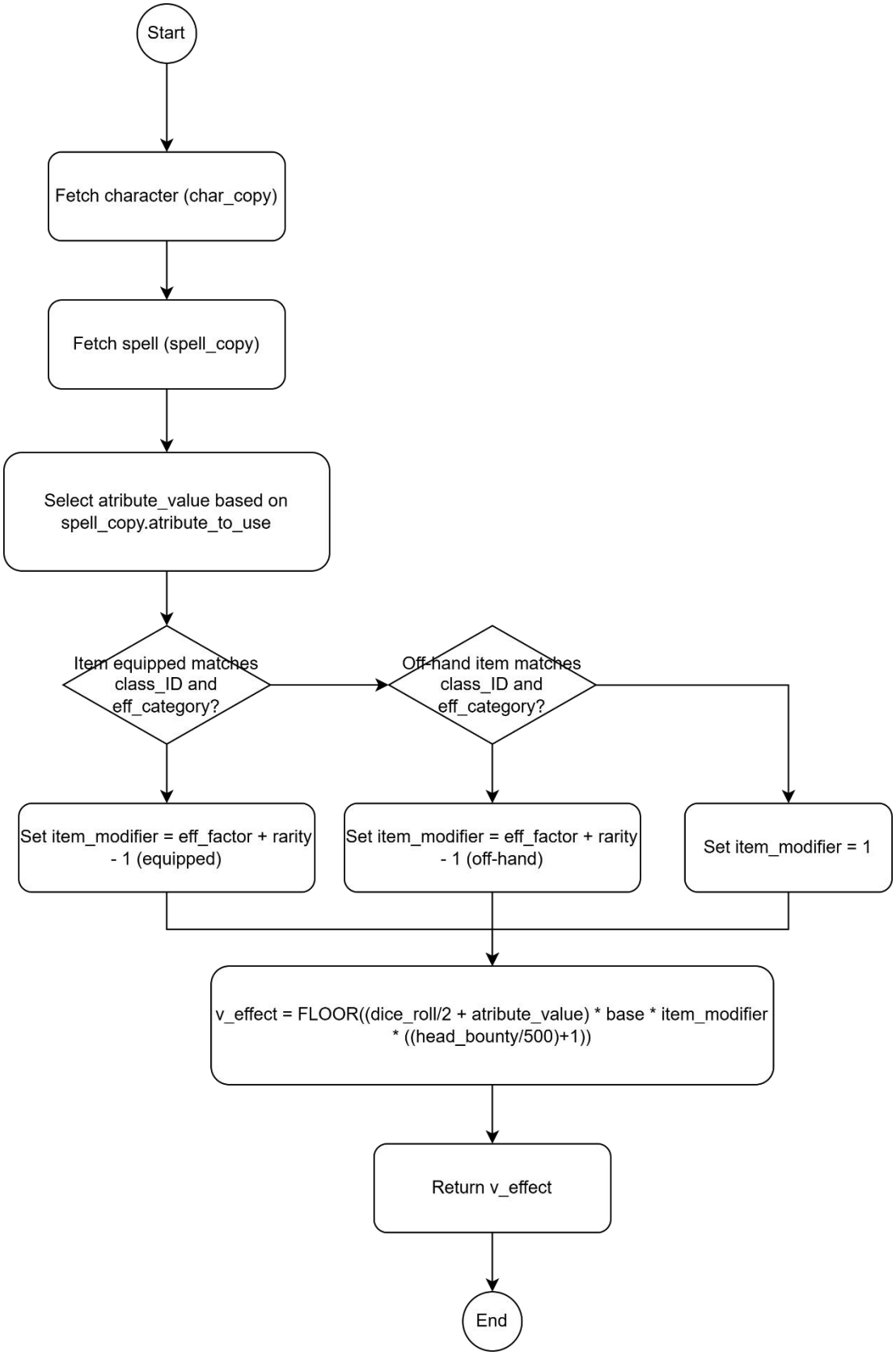
2.1 f_effective_spell_cost

Funkcia na výpočet meniacej sa ceny jednotlivých kúziel



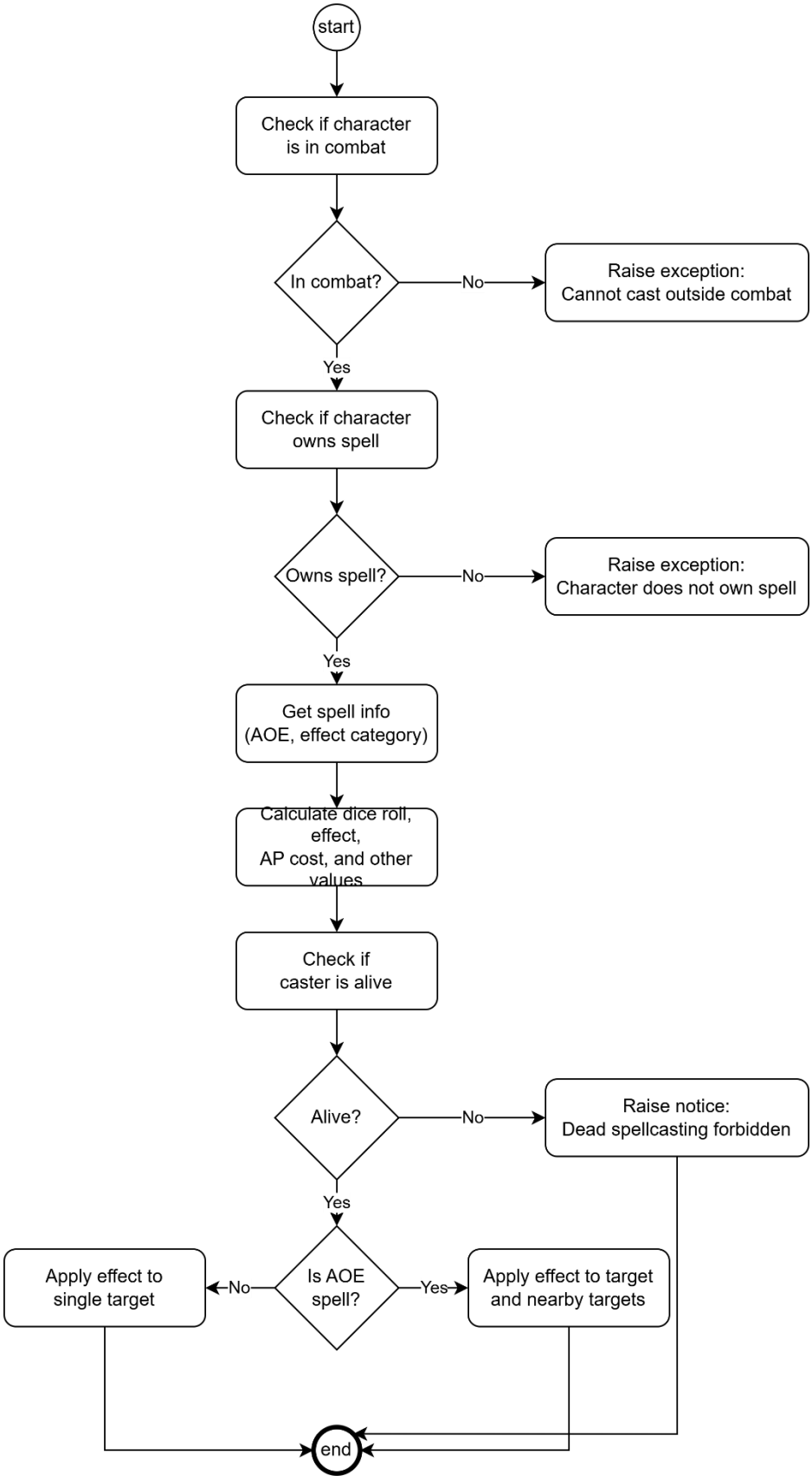
2.2 f_effective_spell_effect

Funkcia na výpočet menaicej sa hodnoty poškodenia



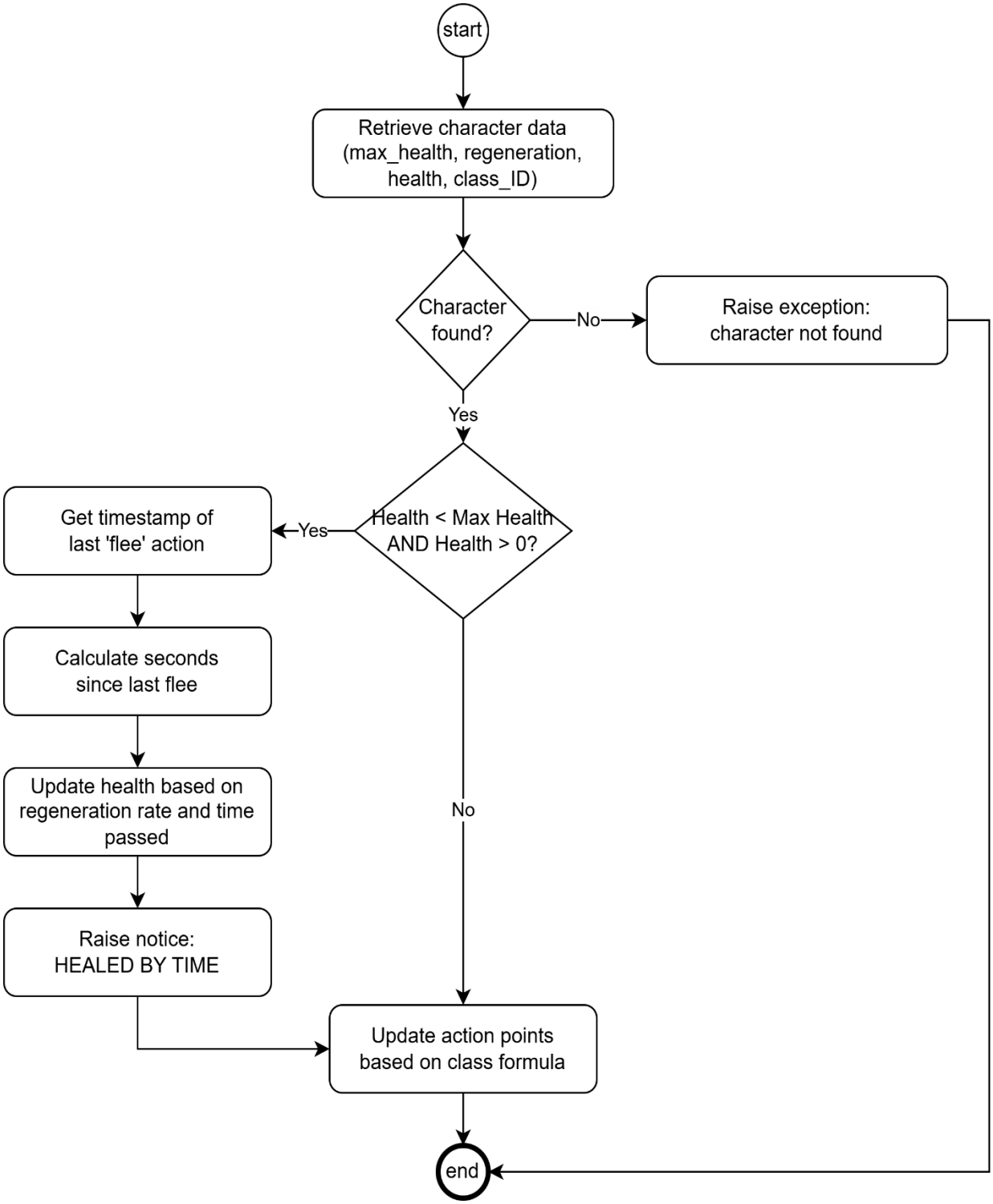
2.3 sp_cast_spell

Funkcia na zoslanie kúzla na cieľ



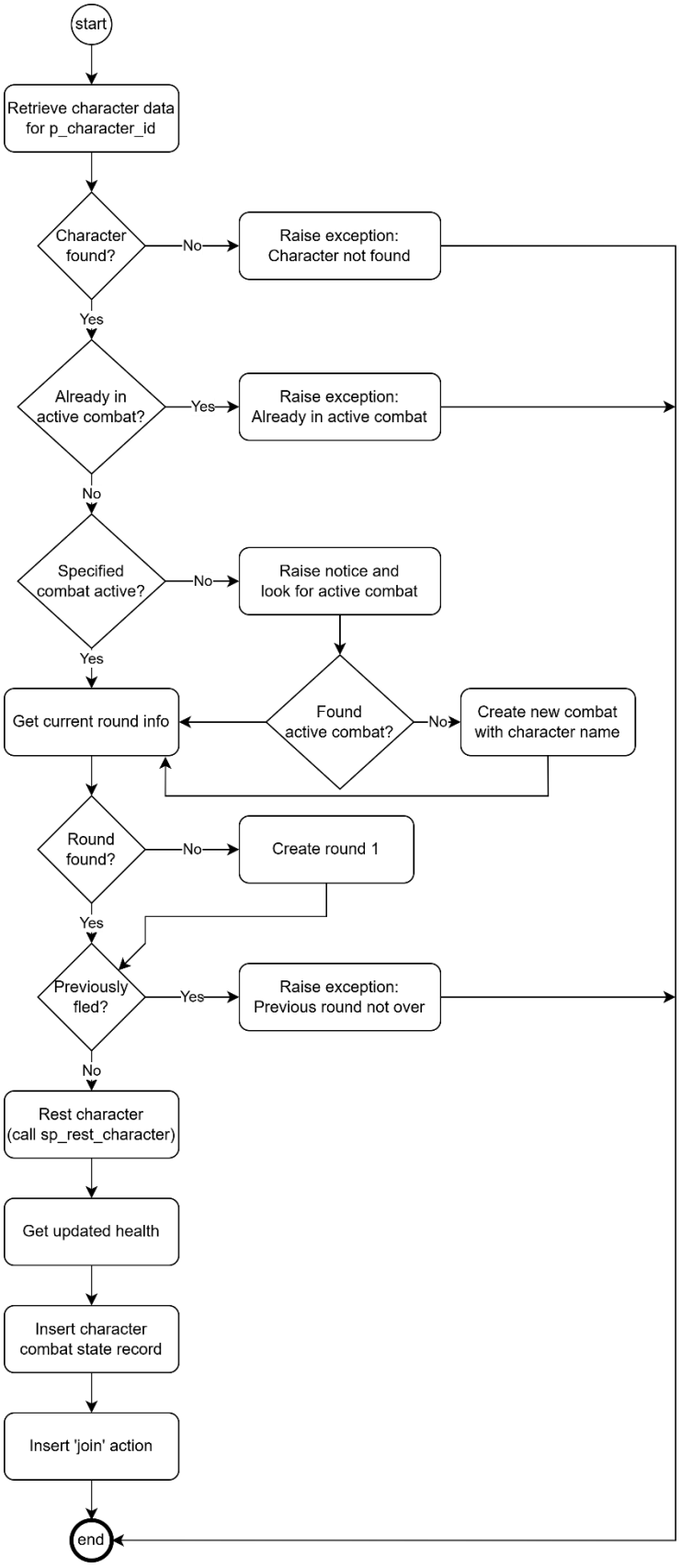
2.4 sp_rest_character

Funkica na výpočet zregenerovaného života počas oddychu



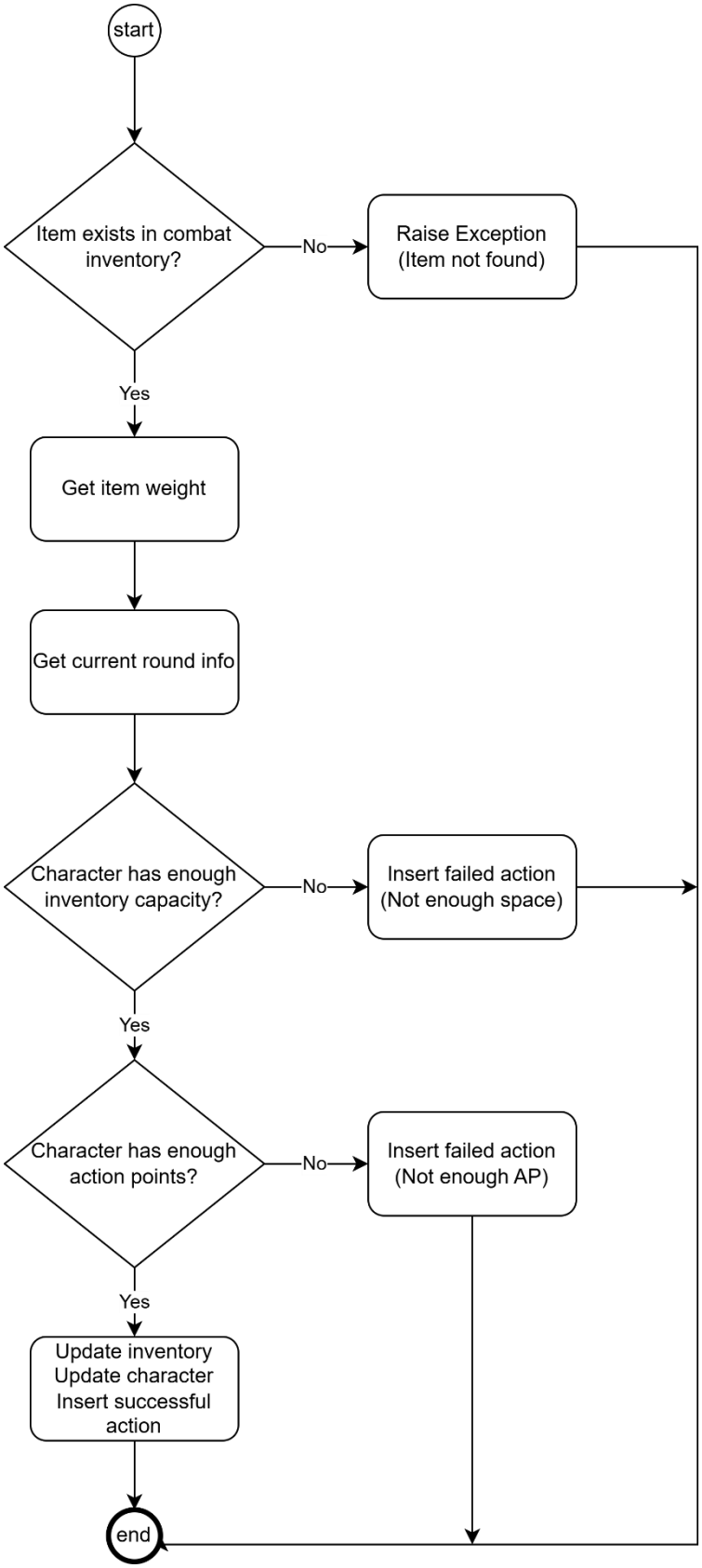
2.5 sp_enter_combat

Funkcia na vloženie postavy do boja



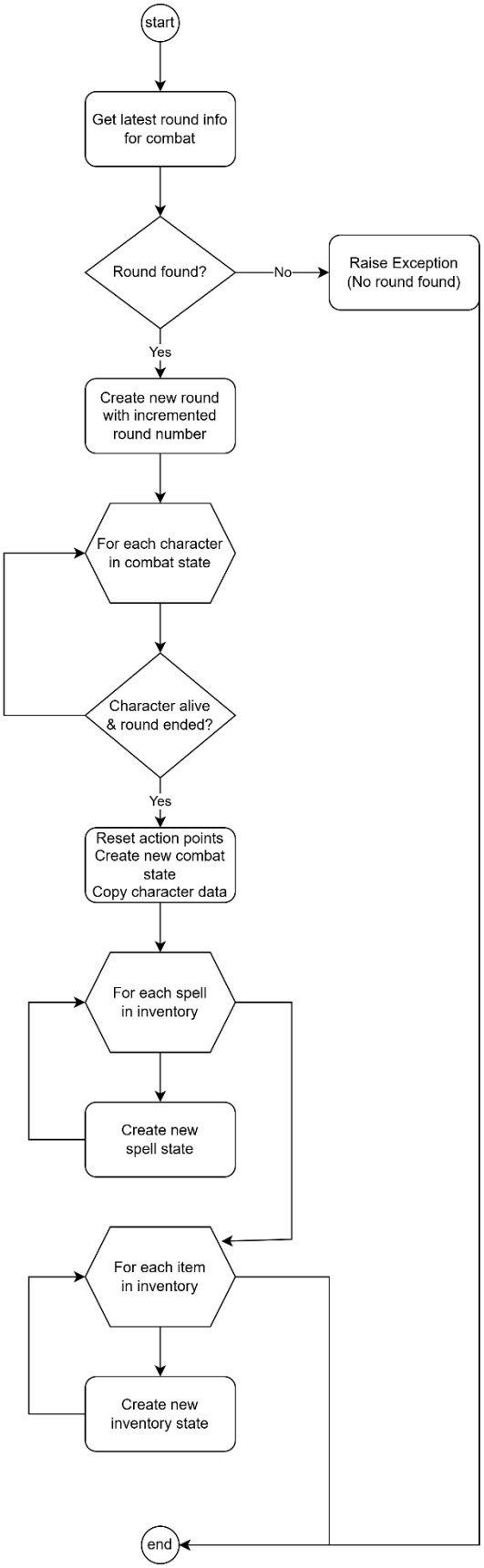
2.6sp_loot_item

Funkcia na zdvihnutie veci zo zeme počas boja

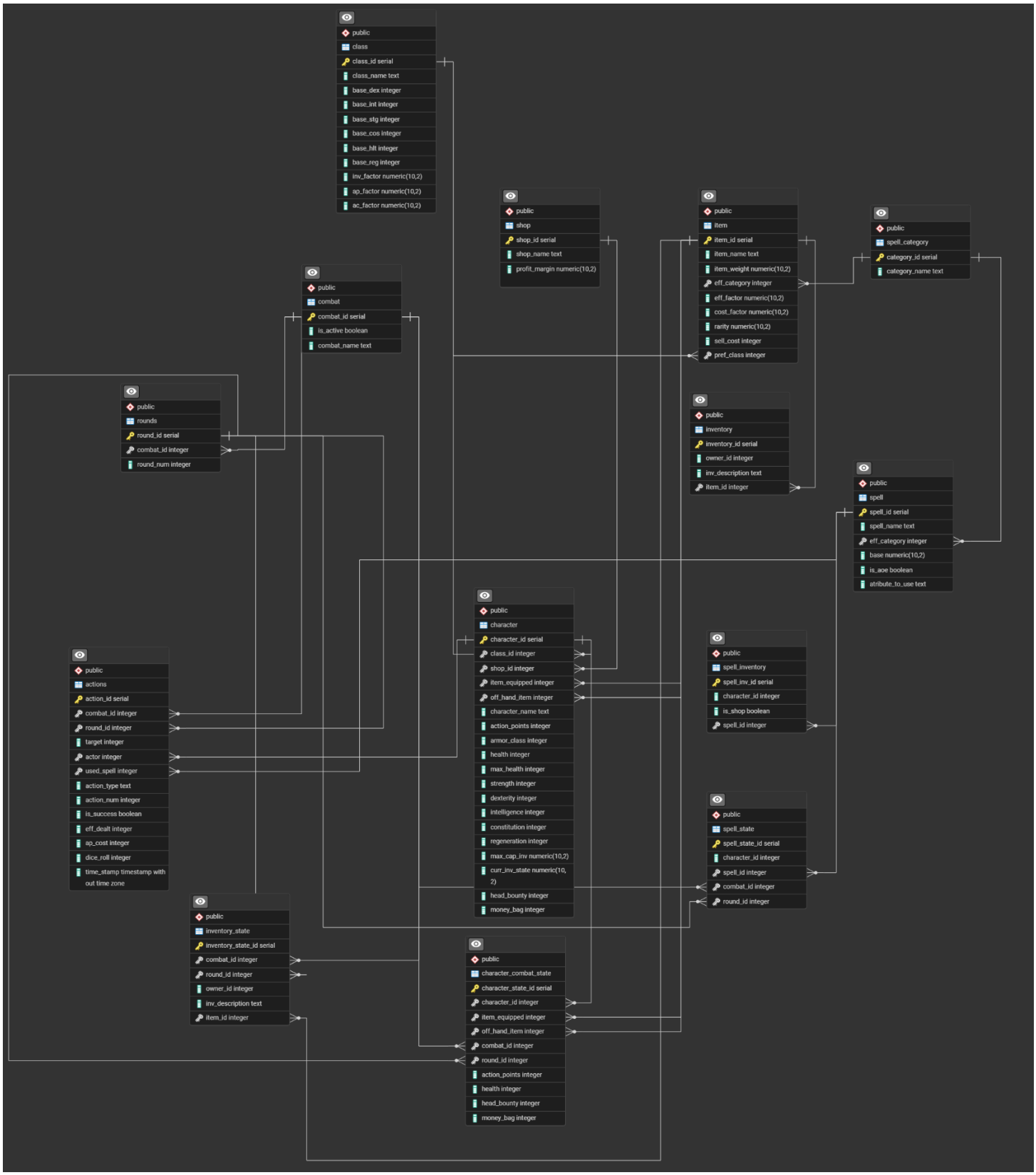


2.7sp_reset_round

Funkcia na resetovanie AP a prechodu do ďalšieho kola



3. Model fyzického diagramu



4. Indexy

Aplikoval som následujúce indexy pre zrýchlenie vyhľadávania pre niektoré tabuľky nakoľko sa može stat že budu veľmi rozsiahle a inak by vyhľadávanie v nich trvalo príliš dlho.

Inventory indexy

- index_inventory_search na (inv_description, owner_ID, item_ID)
- index_inventory_description na (inv_description)

Item indexy

- index_item_item_id na (item_ID)

Rounds indexy

- index_rounds_combat_roundnum na (combat_ID, round_num DESC)

Character indexy

- index_character_character_id na (character_ID)

Spell indexy

- index_spell_spell_id na (spell_ID)

Character Combat State indexy

- index_character_combat_state_search na (character_ID, combat_ID, round_ID)

Combat indexy

- index_combat_active na (is_active)
    - podmienka: WHERE is_active = TRUE

Actions indexy

- index_actions_flee_actor na (actor, action_type, combat_ID, round_ID)
- index_actions_combat_round_type na (combat_ID, round_ID, action_type)

Class indexy

- index_class_class_id na (class_ID)

Spell Inventory indexy

- index_spell_inventory_char_shop na (character_ID, is_shop)

5. Zmeny oproti návrhu

Zmenil som všetky kľúče zo STRINGU na SERIAL pre lepšiu manipuláciu s nimi.
Niektoré výpočty som upravil a pridal ako napríklad:

$$if(user_{class} = pref_{class}\ AND\ spell_{eff} = item_{eff}) * (eff_{factor} + rarirty - 2) + 1 = item_{modifier}$$

$$\left(chosen_{atribute} + \frac{dice_{roll}}{2}\right) * base * item_{modifier} * \left(\frac{head_{bounty}}{500} + 1\right) = eff_{dealt}$$

$$(30 - chosen_{atribute}) * base * if(spell_{eff} = item_{eff}) * cost_{factor} + 1 = AP_{cost}$$

$$(dex + int) * ap_{factor} = Max_{AP} \qquad (stg + cos) * inv_{factor} = Max_{inv}\ capacity$$

Pridal som do tabuľky actions nový stĺpec a to time_stamp aby som vedel určiť o koľko sa má postava vyliečiť po čase mimo boja, takisto som zmenil úlohu tabuľky actions, aktuálne slúži ako zálohovacia tabuľka pre všetky akcie vykonané počas boja.

6. Pokyny na spustenie

ZIP súbor obsahuje sql súbory ktoré sa majú spúšťať nasledovne:

1. table_creation.sql → vytvorí požadované tabuľky
2. table_filling.sql →naplní statické tabuľky okrem item
3. items.sql →naplní item tabuľku
4. indexes.sql → vytvorí indexy
5. functions_procedures.sql → vytvorí funkcie
6. views.sql → vytvorí pohľady
7. simulation.sql →naplní ostatné tabuľky (možno použiť views)
    a. simuláciu možno spustiť viacnásobne a pozorovať sp_rest_character
    b. simulácia má 2 „módy“ jeden prejde celý combat až do konca a druhý zastavý v 10 kole aby boly lepšie použiteľné views combat_state a strongest_character. Prepínať medzi nimi je možné zakomentovaním alebo odkomentovním zvýrazneného bloku kódu.
8. tests.sql → testy na niektoré use casy (treba premazať tabuľky naplnené od simulácie)