Martin Kvietok
ID: 127214

Implementácia dátového modelu v PostgreSQL

(Zadanie 3 – Dokumentácia)

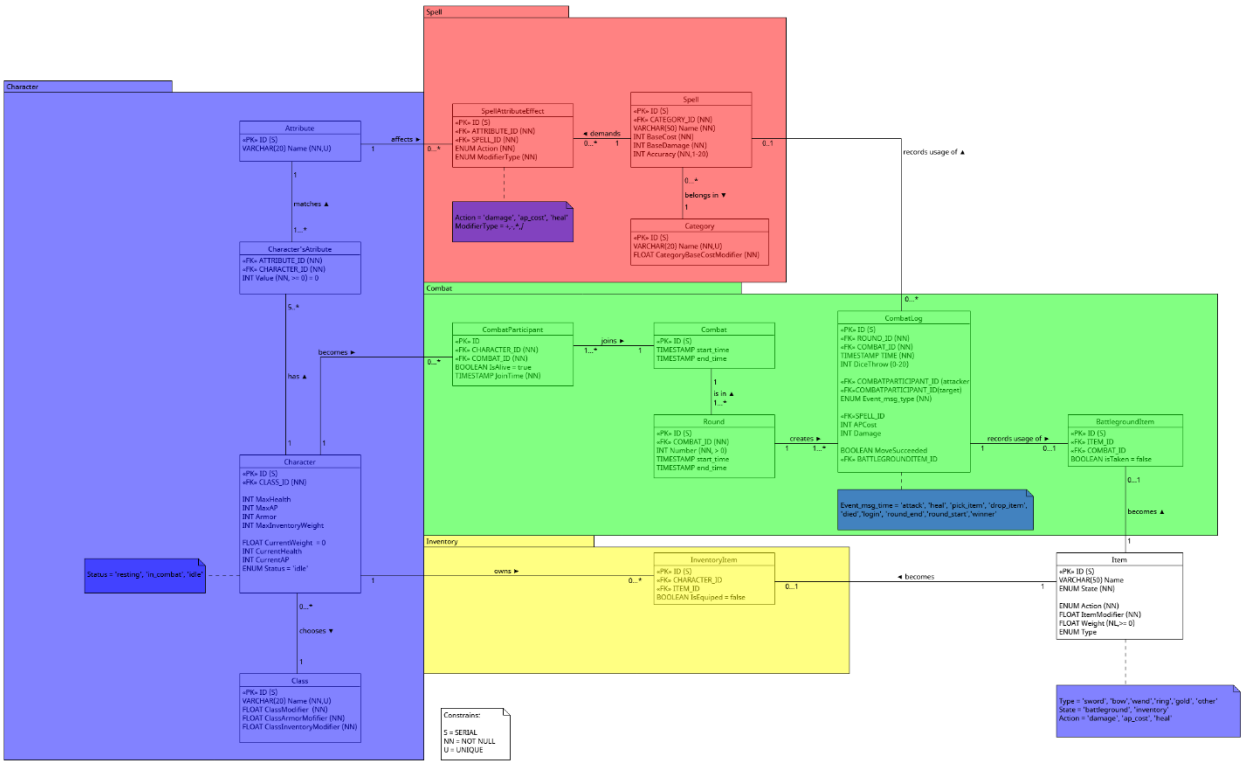
LS 2024/2025

Contents

Martin Kvietok
ID: 127214

Logicko-fyzické mapovanie

Mapovanie vychádza z úprav logického modelu, kde sa pre každý atribút v entite pridávajú obmedzenia (constraints). Každá entita sa transformuje na samostatnú tabuľku vo fyzickom modeli, čím sa zabezpečuje funkčnosť databázy. Do týchto tabuliek sa následne pridávajú záznamy s nastaviteľnými atribútmi. Všetky detaily sú obsiahnuté v diagrame fyzického modelu.



Obrázok 1 Diagram fyzického modelu

Martin Kvietok
ID: 127214

Vysvetlenie procesných tokov

p_rest_character

Popis

Táto funkcia slúži na simuláciu odpočinku postavy v hre. Počas odpočinku sa postupne obnovuje zdravie (**current_health**) a akčné body (**current_ap**) postavy až na ich maximálnu hodnotu (**max_health** a **max_ap**). Pred začatím odpočinku musí byť postava v stave 'idle' (čiže mimo boja). Počas odpočinku sa stav postavy zmení na 'resting' a po dosiahnutí plného zdravia a akčných bodov sa opäť zmení na 'idle'.

Vstupné parametre: *selected_character_id (INTEGER)*
Identifikátor postavy, ktorá si má odpočinúť. Tento ID zodpovedá primárnemu kľúču v tabuľke **character**.

Výstupné hodnoty: *VOID*
Funkcia nevracia žiadnu explicitnú hodnotu. Efektom funkcie sú zmeny v riadku zodpovedajúcej postavy v tabuľke **character**.

Logika funkcie

1. **Načítanie dát postavy:** Na začiatku funkcie sa z tabuľky **character** načítajú aktuálny stav (*status*), aktuálne zdravie (*current_health*), maximálne zdravie (max_health), aktuálne akčné body (*current_ap*) a maximálne akčné body (*max_ap*) pre postavu so zadaným **selected_character_id**. Riadok postavy sa uzamkne (FOR UPDATE) počas trvania transakcie, aby sa predišlo konfliktom pri súbežných zmenách.

2. **Kontrola existencie postavy:** Funkcia overí, či postava so zadaným ID existuje. Ak neexistuje, vyvolá sa výnimka s informáciou o nenájdení postavy.

3. **Kontrola stavu postavy:** Pred začatím odpočinku sa skontroluje, či je aktuálny stav postavy 'idle'. Ak postava nie je v stave 'idle', vyvolá sa výnimka s informáciou o tom, že postava musí byť v stave 'idle', aby mohla odpočívať, a uvedie sa jej aktuálny stav.

4. **Nastavenie stavu 'resting':** Ak sú počiatočné podmienky splnené, stav postavy sa v tabuľke **character** zmení na 'resting'.

5. **Regeneračný cyklus:** Nasleduje cyklus WHILE, ktorý pokračuje, kým nie je aktuálne zdravie rovné maximálnemu zdraviu a zároveň aktuálne akčné body nie sú rovné maximálnym akčným bodom. V každej iterácii cyklu sa vykonajú nasledujúce kroky:

   - PERFORM pg_sleep(1): Simuluje sa uplynutie jednej sekundy času.
   - **Regenerácia zdravia:** Ak je aktuálne zdravie menšie ako maximálne zdravie, zvýši sa o 5 bodov, pričom sa zabezpečí, že neprekročí maximálnu hodnotu pomocou funkcie LEAST.
   - **Regenerácia akčných bodov:** Ak sú aktuálne akčné body menšie ako maximálne, zvýšia sa o 5 bodov, pričom sa opäť zabezpečí, že neprekročia maximálnu hodnotu pomocou funkcie LEAST.
   - **Aktualizácia štatistík postavy:** Ak došlo k zmene aktuálneho zdravia alebo akčných bodov, tieto hodnoty sa aktualizujú v tabuľke character pre danú postavu.
   - **Predčasné ukončenie cyklu:** Ak sa v aktuálnej iterácii dosiahne plné zdravie aj plné akčné body, cyklus sa predčasne ukončí pomocou príkazu EXIT.

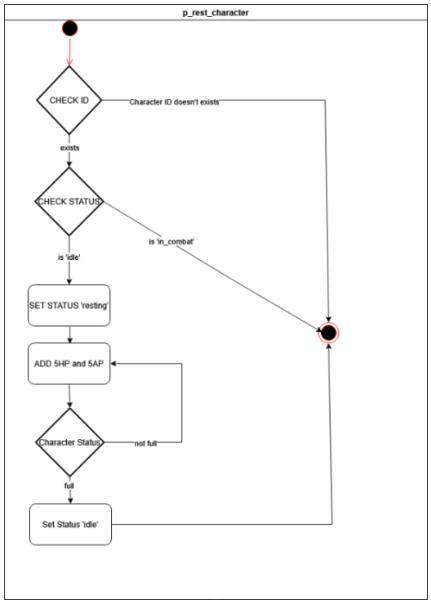
6. **Nastavenie stavu na 'idle':** Po skončení regeneračného cyklu sa stav postavy v tabuľke character opäť zmení na 'idle'.

Použitie:

Funkciu p_rest_character je možné zavolať v SQL príkazoch alebo z iných procedúr/funkcií s poskytnutím ID postavy, ktorá má začať odpočívať. Napríklad:

SELECT p_rest_character(123); --- Tento príkaz spustí proces odpočinku pre postavu s ID 123.

Diagram



Obrázok 2 Diagram aktivít UML - p_rest_character

p_enter_combat

Popis

Táto funkcia umožňuje postave (**character**) vstúpiť do existujúceho aktívneho súboja (**combat**) a priamo sa zapojiť do aktuálne prebiehajúceho kola (**round**). Pred samotným vstupom overí existenciu súboja aj postavy a zároveň skontroluje, či postava už nie je zapojená v inom aktívnom súboji, ktorý ešte prebieha. Po úspešnom overení pridá postavu ako účastníka do tabuľky **combat_participant**, zaznamená čas pripojenia a vytvorí záznam o vstupe postavy do súboja v tabuľke **combat_log**.

Vstupné parametre:

- *selected_combat_id (INTEGER)* Identifikátor súboja, do ktorého má postava vstúpiť. Tento ID zodpovedá primárnemu kľúču v tabuľke **combat**.
- selected_character_id (INTEGER): Identifikátor postavy, ktorá vstupuje do súboja. Tento ID zodpovedá primárnemu kľúču v tabuľke **character**.

Výstupné hodnoty: *VOID*
Funkcia nevracia žiadnu explicitnú hodnotu. Efektom funkcie je pridanie záznamu do tabuľky **combat_participant** a **combat_log**.

Logika funkcie

1. **Získanie aktuálneho času:** Na začiatku funkcie sa získa aktuálny čas pomocou funkcie NOW() a uloží sa do premennej time_joined.

2. **Kontrola existencie súboja a postavy:** Funkcia overí, či existuje súboj so zadaným selected_combat_id v tabuľke combat a či tento súboj ešte neskončil (end_time IS NULL). Následne overí existenciu postavy so zadaným selected_character_id v tabuľke character. Nakoniec skontroluje, či postava už nie je účastníkom iného aktívneho súboja — teda či neexistuje záznam v combat_participant s aktívnym súbojom, kde je is_alive = TRUE. Ak niektorá z týchto podmienok nie je splnená, vyvolá sa výnimka.

3. **Pridanie postavy do combat_participant:** Ak všetky kontroly prešli úspešne, vloží sa nový záznam do tabuľky combat_participant s nasledujúcimi hodnotami:
   - *combat_id:* selected_combat_id
   - *character_id:* selected_character_id
   - *is_alive:* TRUE
   - *join_time:* time_joined

4. **Nájdenie aktuálneho kola:** Funkcia sa pokúsi nájsť aktuálne kolo súboja, do ktorého postava vstúpila. Hľadá sa záznam v tabuľke **round** pre daný *selected_combat_id*, kde začiatok kola (*start_time*) je menší alebo rovný času pripojenia postavy (*time_joined*) a koniec kola (*end_time*) je NULL (kolo ešte neskončilo).

5. **Ošetrenie prípadu nenájdenia aktuálneho kola:** Ak sa pre daný čas pripojenia nenájde žiadne aktuálne kolo, vyvolá sa výnimka. Toto môže nastať, ak napríklad postava vstúpi do súboja medzi kolami alebo ak nie je správne nastavené trvanie kôl.

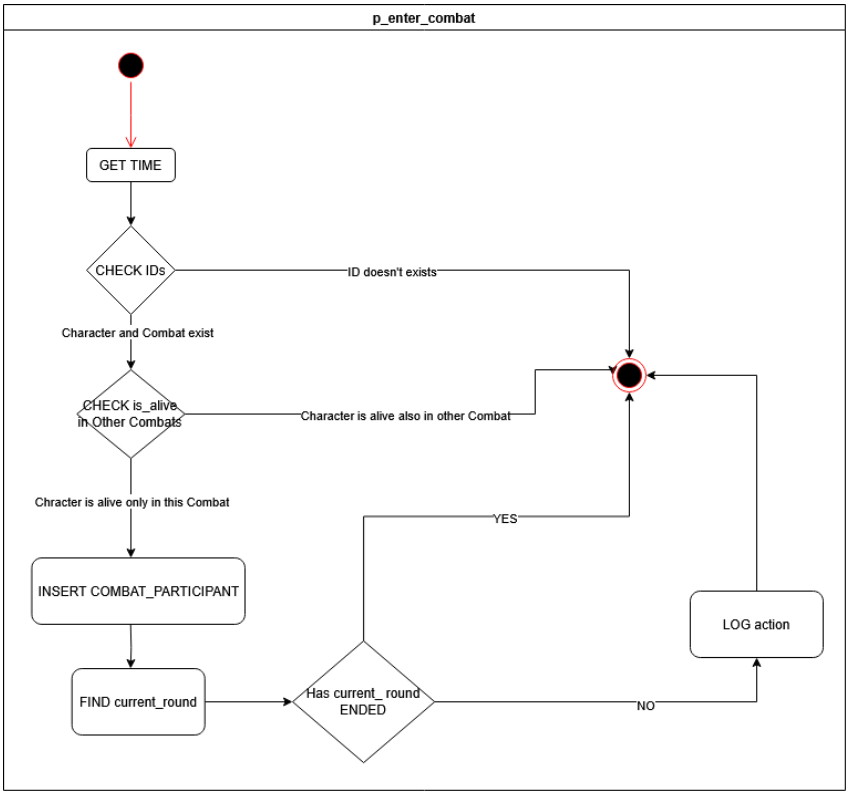
6. **Zaznamenanie vstupu do combat_log:** Do tabuľky combat_log sa vloží nový záznam o vstupe postavy do súboja:
   - *round_id*: current_round_id (ID aktuálneho kola)
   - *combat_id*: selected_combat_id
   - *time*: time_joined (čas udalosti)
   - *target_id*: combat_participant_id (ID záznamu postavy v combat_participant)
   - *event_msg_type*: 'login' (typ udalosti - prihlásenie do súboja)
   - *move_success*: TRUE (vstup do súboja sa považuje za úspešný)

Použitie

Funkciu p_enter_combat je možné zavolať s ID súboja a ID postavy, ktorá sa má do tohto súboja pridať:

SELECT p_enter_combat(101, 205); --- Tento príkaz pokúsi sa pridať postavu s ID 205 do súboja s ID 101.

Diagram



Obrázok 3 Diagram aktivít UML - p_enter_combat

p_reset_round

Popis

Táto funkcia slúži na ukončenie aktuálneho kola a inicializáciu nového. Po zavolaní najprv zaznamená koniec prebiehajúceho kola (**round**), následne resetuje akčné body (AP) všetkým živým účastníkom. V prípade, že zostal nažive len jeden účastník, ukončí súboj (**combat**) a označí víťaza. Ak je živých účastníkov viac, vytvorí nové kolo a zaznamená jeho začiatok.

Vstupné parametre: *selected_combat_id (INTEGER)*
Identifikátor súboja, pre ktorý sa má resetovať kolo. Tento ID zodpovedá primárnemu kľúču v tabuľke **combat**.

Výstupné hodnoty: *VOID*
Funkcia nevracia žiadnu hodnotu. Jej účinkom sú zmeny v tabuľke **round**, pridanie záznamov do **combat_log** a aktualizácie v tabuľkách **combat** a **character**.

Logika funkcie

1. **Kontrola existencie súboja:** Funkcia overí, či existuje súboj so zadaným selected_combat_id v tabuľke combat. Ak súboj neexistuje, vyvolá sa výnimka.

2. **Získanie aktuálneho času konca kola:** Získa sa aktuálny čas pomocou funkcie NOW() a uloží sa do premennej time_ended. Táto časová pečiatka bude použitá na označenie konca aktuálneho kola a začiatku nového.

3. **Spracovanie živých účastníkov súboja:** Z tabuľky combat_participant sa pre daný selected_combat_id spočítajú všetci účastníci so stavom is_alive = TRUE a výsledok sa uloží do premennej alive_count. Následne sa pre každého živého účastníka zavolá funkcia p_reset_current_ap s jeho character_id, aby sa mu obnovili aktuálne akčné body.

4. **Získanie informácií o aktuálnom kole:** Z tabuľky round sa vyberie ID a číslo posledného začatého kola pre daný selected_combat_id. Predpokladá sa, že najnovšie začaté kolo je zároveň aktuálne kolo, ktoré sa resetuje. V nájdenom kole sa nastaví stĺpec end_time na hodnotu time_ended.

5. **Zaznamenanie konca kola do combat_log:** Do tabuľky combat_log sa vloží nový záznam o konci kola:
   - *round_id*: current_round_id (ID ukončeného kola)
   - *combat_id*: selected_combat_id
   - *time*: time_ended (čas ukončenia kola)
   - *event_msg_type*: 'round_end' (typ udalosti - koniec kola)
   - *move_success:* TRUE

6. **Kontrola víťaza :** Ak je po ukončení kola počet živých účastníkov (alive_count) je rovný 1, funkcia predpokladá, že súboj skončil a tento posledný žijúci hráč je víťaz. Vykonajú sa nasledujúce kroky:
   - Získa sa ID posledného žijúceho účastníka z combat_participant.
   - Do combat_log sa zaznamená udalosť 'winner' s ID víťazného účastníka ako target_id.
   - V tabuľke combat_participant sa pre víťazného účastníka nastaví is_alive na FALSE (aj keď už bol jediný živý).
   - V tabuľke combat sa nastaví end_time na time_ended, čím sa súboj oficiálne ukončí.
   - Funkcia sa ukončí pomocou RETURN, pretože súboj je už skončený a nie je potrebné vytvárať nové kolo.

7. **Vytvorenie nového kola:** Ak je počet živých účastníkov väčší ako 1, vytvorí sa nový záznam v tabuľke round:
   - *number*: current_round_number + 1 (číslo nového kola je o jedna väčšie ako číslo predchádzajúceho kola)
   - *combat_id*: selected_combat_id
   - *start_time*: time_ended (začiatok nového kola je v čase, keď sa skončilo predchádzajúce kolo)

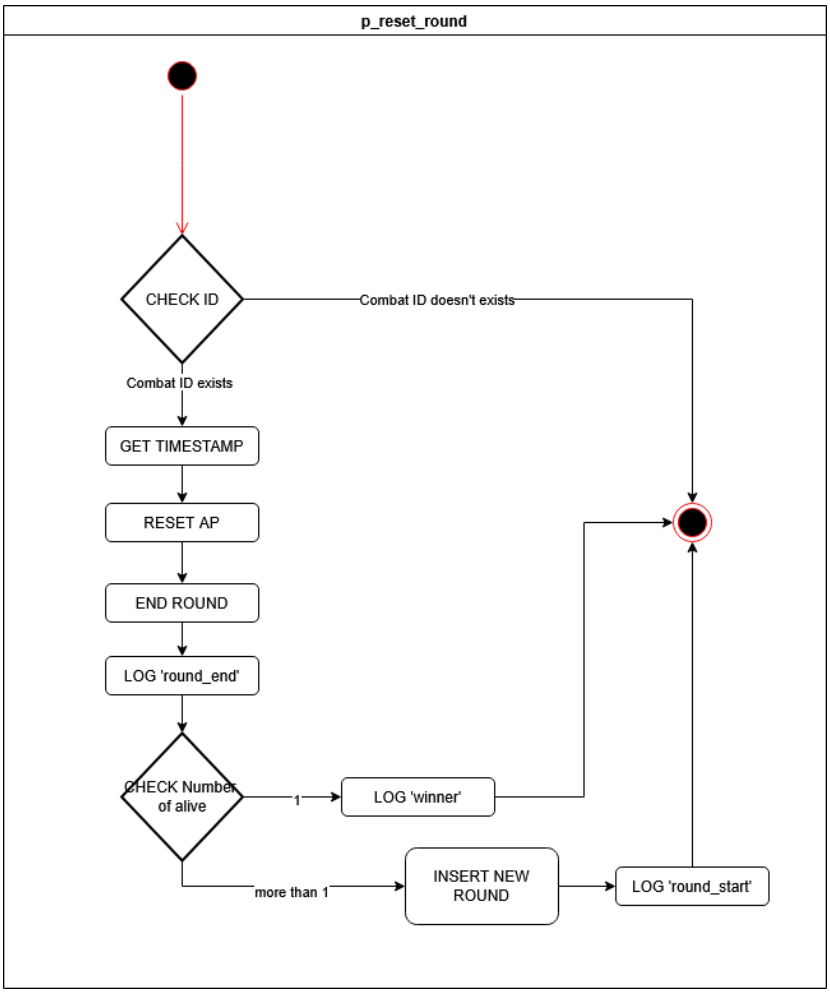
8. **Zaznamenanie začiatku nového kola do combat_log:** Do tabuľky combat_log sa vloží nový záznam o začiatku nového kola:
   - *round_id:* current_round_id + 1 (ID novovytvoreného kola)
   - *combat_id*: selected_combat_id
   - *time:* time_ended (čas začiatku kola)
   - *event_msg_type:* 'round_start' (typ udalosti - začiatok kola)
   - *move_success:* TRUE

Použitie:

Funkciu p_reset_round sa volá na konci každého kola súboja s ID daného súboja:

SELECT p_reset_round(101); --- Tento príkaz ukončí aktuálne kolo a pripraví nasledujúce kolo pre súboj s ID 101.

Diagram:



Obrázok 4 Diagram aktivít UML - p_reset_round

p_loot_item

Popis

Táto funkcia umožňuje postave počas súboja zobrať predmet z bojiska. Pred samotným pridaním predmetu musí overiť jeho dostupnosť, či je postava aktívnym účastníkom súboja a aktuálnu kapacitu inventára. Ak sú všetky podmienky splnené, funkcia pridá predmet postave do inventára (**inventory_item**) prostredníctvom *p_add_item*. Nakoniec označí predmet ako zobratý (*isTaken* = True) v tabuľke **battleground_item** a zaznamená udalosť do tabuľky **combat_log**.

Vstupné parametre:

- *selected_combat_id* (INTEGER): Identifikátor súboja, v ktorom sa postava nachádza a z ktorého chce predmet zobrať. Tento ID zodpovedá primárnemu kľúču v tabuľke **combat**.
- *selected_character_id* (INTEGER): Identifikátor postavy, ktorá chce predmet zobrať. Tento ID zodpovedá primárnemu kľúču v tabuľke **character**.
- *selected_item_id* (INTEGER): Identifikátor predmetu, ktorý chce postava zobrať. Tento ID zodpovedá stĺpcu *item_id* v tabuľke **battleground_item** a zodpovedá primárnemu kľúču v tabuľke **item**.

Výstupné hodnoty: *VOID*
Funkcia nevracia žiadnu explicitnú hodnotu. Efektom funkcie sú zmeny v tabuľkách **battleground_item**, **character** (nepriamo cez volanie p_add_item), **item**, **inventory_item** a **combat_log**.

Logika funkcie

1. **Overenie dostupnosti predmetu a uzamknutie riadku:** Funkcia overí, či existuje predmet so zadaným *selected_item_id* v tabuľke ***battleground_item*** pre daný *selected_combat_id* a či je predmet ešte dostupný (*is_taken* = FALSE). Ak predmet nie je v tabuľke alebo už bol zobratý, vyvolá sa výnimka. Riadok v tabuľke **battleground_item** sa uzamkne pomocou FOR UPDATE, aby sa predišlo súbežným pokusom o zobratie toho istého predmetu.

2. **Overenie účasti postavy v súboji:** Funkcia overí, či je postava so zadaným *selected_character_id* účastníkom súboja so zadaným *selected_combat_id* a či je nažive (*is_alive* = TRUE) v tabuľke **combat_participant**. Ak postava nie je účastníkom súboja, vyvolá sa výnimka. Získava sa id postavy.

3. **Kontrola nosnosti postavy:** Funkcia najprv získa z tabuľky character aktuálnu (*current_weight*) a maximálnu (*max_inventory*) nosnosť postavy podľa *selected_character_id* a z tabuľky **item** váhu predmetu (*weight*) podľa *selected_item_id*. Ak postava alebo predmet neexistujú, vyvolá sa výnimka. Následne overí, či súčet aktuálnej nosnosti postavy a váhy predmetu nepresahuje jej maximálnu nosnosť ak áno vyvolá sa výnimka.

4. **Pridanie predmetu postave a označenie:** Ak postava môže predmet zobrať, zavolá sa existujúca funkcia *p_add_item* s parametrami *selected_character_id* a *selected_item_id*, ktorá buď vytvorí nový záznam v **inventory_item**, alebo aktualizuje existujúci záznam tak, že nastaví novú hodnotu *character_id* a zmení stav(*state*) v tabuľke **item** na hodnotu "inventory". Následne sa v tabuľke

battleground_item pre daný predmet (*selected_item_id*) v rámci súboja (*selected_combat_id*) nastaví stĺpec *is_taken* na hodnotu TRUE.

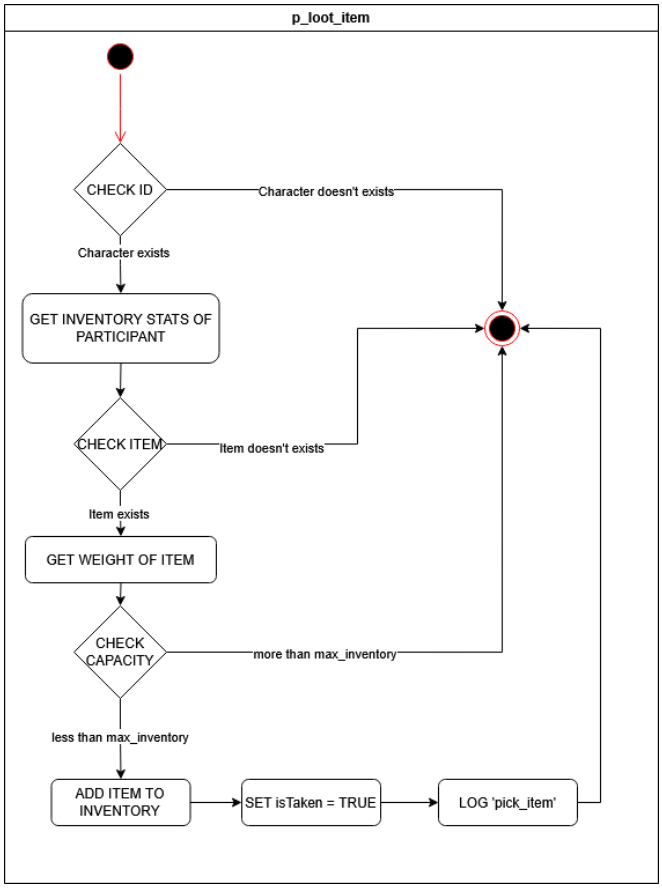
5. **Zaznamenanie udalosti do combat_log:** Do tabuľky combat_log sa vloží nový záznam o zobratí predmetu:
   - *round_id*: ID posledného kola (last_round_id)
   - *combat_id*: selected_combat_id
   - *time*: Aktuálny čas (now())
   - *target_id*: combat_character_id (ID postavy, ktorá zobrala predmet)
   - *event_msg_type*: 'pick_item' (typ udalosti - zobratie predmetu)
   - *move_success*: TRUE
   - *item_id*: cur_item_id (ID záznamu predmetu na bojisku)

Použitie

Funkciu p_loot_item je možné zavolať s ID súboja, ID postavy a ID predmetu, ktorý chce postava zobrať:

SELECT p_loot_item(101, 205, 302); --- Tento príkaz umožní postave s ID 205 zobrať predmet s ID 302 zo súboja s ID 101.

Diagram



Obrázok 5 Diagram aktivít UML - p_loot_item

p_effective_spell_cost

Popis

Táto funkcia vypočítava efektívnu cenu kúzla pre konkrétnu postavu (**character**) na základe základnej ceny kúzla, modifikátorov kategórie kúzla, atribútov postavy a vybavených predmetov. Funkcia berie do úvahy modifikátory atribútov postavy, ktoré ovplyvňujú cenu kúzla (zvyšujú alebo znižujú), a modifikátory vybavených predmetov, ktoré tiež ovplyvňujú cenu kúzla.

Vstupné parametre:

- *selected_spell_id* (INTEGER): Identifikátor kúzla, pre ktoré sa má vypočítať efektívna cena. Tento ID zodpovedá primárnemu kľúču v tabuľke **spell**.
- *selected_caster_id* (INTEGER): Identifikátor postavy (castera), ktorá chce kúzlo použiť. Tento ID zodpovedá primárnemu kľúču v tabuľke **character**.

Výstupné hodnoty: *NUMERIC*
Efektívna cena kúzla pre daného castera, zaokrúhlená na celé číslo.

Logika funkcie

1. **Načítanie údajov o kúzle a jeho kategórii:** Funkcia pre zadané *selected_spell_id* načíta z tabuľky **spell** základnú cenu kúzla (*base_cost*) a ID kategórie (*category_id*). Následne z tabuľky **category** získa modifikátor základnej ceny (*category_base_cost_modifier*) podľa načítaného *category_id*. Ak kúzlo so zadaným *selected_spell_id* neexistuje, vyvolá sa výnimka.

2. **Aplikácia modifikátorov atribútov:** Funkcia iteruje cez všetky atribúty spojené s daným kúzlom v tabuľke **spell_attribute**, ktoré ovplyvňujú cenu akčných bodov (*action* = 'ap_cost'). Pre každý takýto atribút:
   - Získa hodnotu atribútu (*value*) pre daného castera z tabuľky **character_attribute**.
   - Ak hodnota atribútu existuje, aplikuje sa modifikátor na základe typu modifikátora (modifier_type):
     - '+': Hodnota atribútu delená 100 sa pripočíta k *attribute_modifier_sum*.
     - '-': Hodnota atribútu delená 100 sa odpočíta od *attribute_modifier_sum*.
     - '*': attribute_modifier_sum sa vynásobí (hodnotou atribútu delenou 100)
     - '/': attribute_modifier_sum sa vydelí (hodnotou atribútu delenou 100), ak hodnota atribútu nie je 0

3. **Aplikácia modifikátorov predmetov:** Funkcia iteruje cez všetky predmety postavy, ktoré sú aktuálne nasadené (*isEquipped* = True). Pre každý vybavený predmet:
   - Získa modifikátor predmetu (*item_modifier*) z tabuľky **item**.
   - Pripočíta modifikátor predmetu k *item_modifier_sum*.

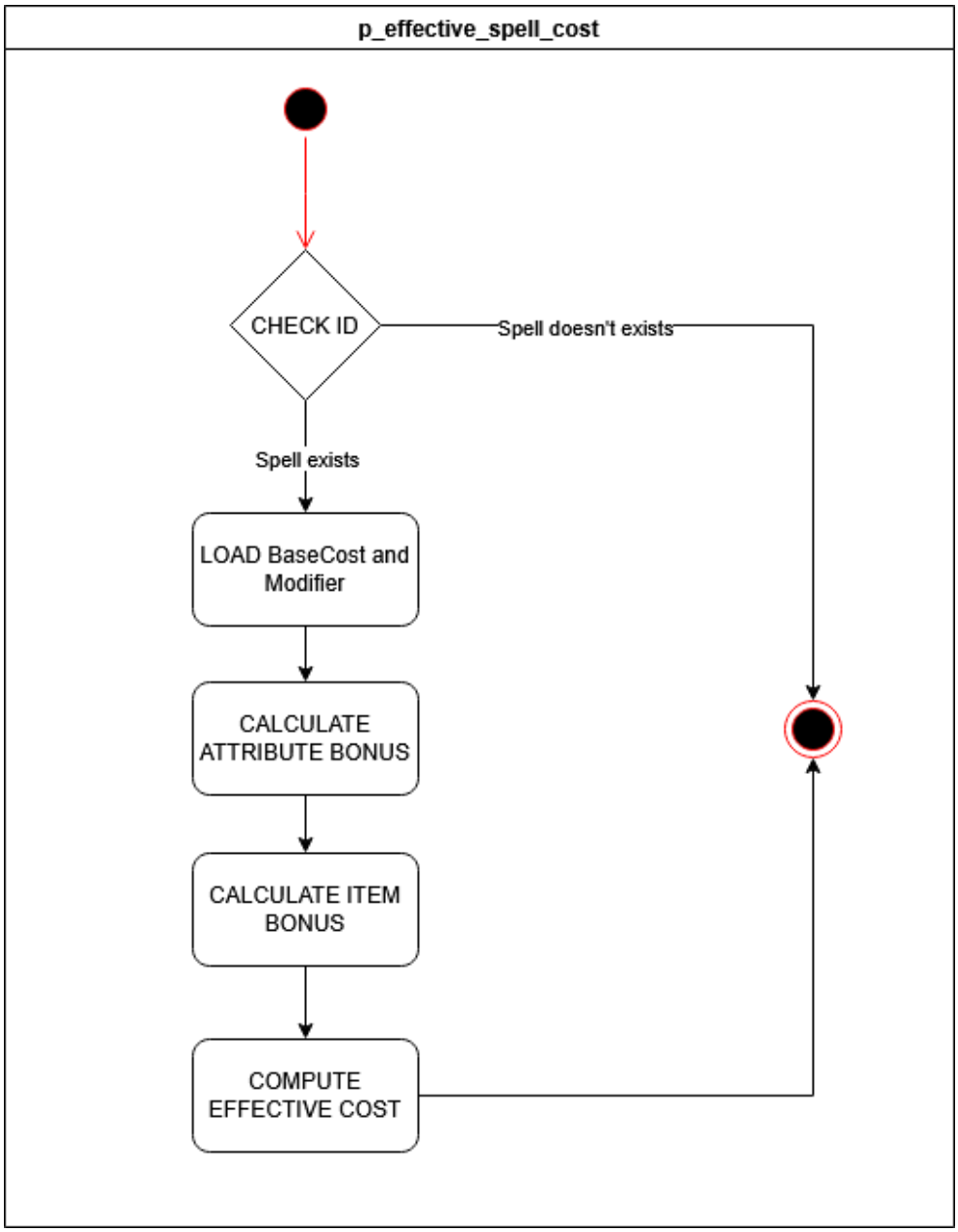
4. **Výpočet efektívnej ceny:** Vypočíta sa efektívna cena kúzla pomocou nasledujúceho vzorca:
   - **effective_cost** = spell_base_cost * spell_base_cost_modifier * (1 - attribute_modifier_sum) * (1 - item_modifier_sum)

Použitie

Funkciu p_effective_spell_cost je možné zavolať s ID kúzla a ID postavy (castera):

SELECT p_effective_spell_cost(201, 102); --- Tento príkaz vráti efektívnu cenu kúzla s ID 201 pre postavu s ID 102.

Diagram



Obrázok 6 Diagram aktivít UML - p_effective_cost

p_cast_spell

Popis

Táto funkcia simuluje zoslanie kúzla postavou (**character** - CASTER) na cieľ (**character** - TARGET) v rámci súboja. Najprv overí, či sú obe postavy (caster a target) nažive a prítomní v súboji. Následne vypočíta cenu akčných bodov (AP) potrebných na použitie kúzla a vykoná hod d20 kockou na určenie úspešnosti zásahu. Funkcia potom vypočíta poškodenie (alebo liečenie) a aplikuje ho na cieľ. Záznam o tejto udalosti sa uloží do tabuľky **combat_log**. Na záver funkcia aktualizuje život targetu, a v prípade smrti túto udalosť zaznamená.

Vstupné parametre:

- *selected_caster_id* (INTEGER): Identifikátor postavy, ktorá zosiela kúzlo. Tento ID zodpovedá primárnemu kľúču v tabuľke character.*selected_target_id* (INTEGER): Identifikátor postavy, ktorá je cieľom kúzla. Tento ID zodpovedá primárnemu kľúču v tabuľke character.
- selected_spell_id (INTEGER): Identifikátor kúzla, ktoré sa má zoslať. Tento ID zodpovedá primárnemu kľúču v tabuľke spell.
- *selected_combat_id* (INTEGER): Identifikátor súboja, v ktorom sa kúzlo zosiela. Tento ID zodpovedá primárnemu kľúču v tabuľke combat.

Výstupné hodnoty: *VOID*
Funkcia nevracia žiadnu explicitnú hodnotu. Efektom funkcie sú zmeny v tabuľkách **character** (zmena AP castera a zdravia targeta), **combat_participant** (potenciálna zmena stavu is_alive targeta) a pridanie záznamu do **combat_log**

Logika funkcie

1. **Overenie castera a cieľa:** Funkcia overí, či je postava zosielajúca kúzlo (caster) nažive a účastní sa daného súboja. Získa sa ID záznamu castera v tabuľke **combat_participant** a jeho aktuálne AP. Ak caster nie je platným účastníkom súboja, vyvolá sa výnimka. Následne sa overí, či je cieľ kúzla (target) nažive a č isa tiež účastní daného súboja. Získa sa ID záznamu targeta v tabuľke **combat_participant** a jeho hodnota brnenia. Ak cieľ nie je platným účastníkom súboja, vyvolá sa výnimka.

2. **Spracovanie ceny a odpočítanie AP:** Funkcia získa efektívnu cenu AP kúzla volaním p_effective_spell_cost s ID kúzla a ID castera, následne overí, či má caster dostatok AP; ak áno, odpočíta cenu AP od jeho aktuálnych AP v tabuľke character, inak vyvolá výnimku.

3. **Simulácia hodu kockou:** Simuluje sa hod 20-stennou kockou (d20) pomocou funkcie random() a výsledok sa uloží do premennej *roll_result*.

4. **Výpočet poškodenia kúzla:** Funkcia najprv zavolá funkciu **p_spell_damage** s ID kúzla a ID castera, aby získala základné poškodenie kúzla. Následne sa z premennej combat_target získa hodnota brnenia cieľa. Ak kúzlo nie je liečivé, aplikuje sa zníženie poškodenia na základe hodnoty brnenia cieľa: ak je brnenie väčšie alebo rovné poškodeniu, poškodenie sa nastaví na 0, inak sa poškodenie zníži o percentuálnu hodnotu vypočítanú ako (brnenie / 20).

5. **Logika liečenia:** Ak je kúzlo liečivé, zaznamená sa udalosť liečenia do **combat_log**, z tabuľky **character** sa získa maximálne zdravie cieľa, aktualizuje sa jeho aktuálne zdravie pričítaním hodnoty liečenia (s limitom na maximum cez funkciu LEAST) a funkcia sa následne ukončí pomocou RETURN.

6. **Kontrola úspešnosti zásahu a aplikácia poškodenia:** Ak kúzlo nie je liečivé, porovná sa jeho presnosť s výsledkom hodu kockou; ak je presnosť väčšia alebo rovná výsledku hodu, kúzlo zasiahne a aktuálne zdravie cieľa v tabuľke **character** sa zníži o hodnotu poškodenia, inak kúzlo minie.

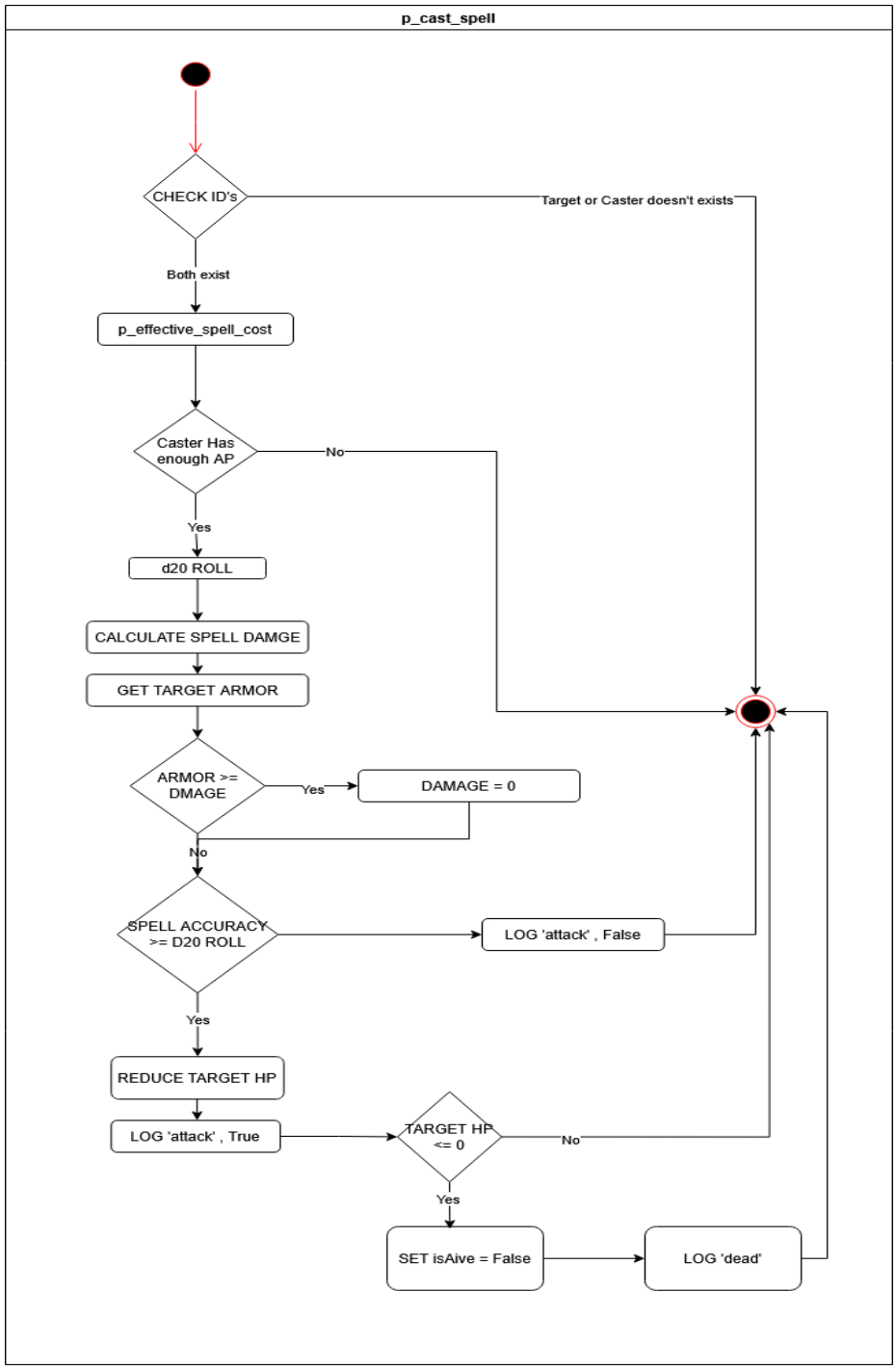
7. **Kontrola smrti cieľa:** Ak je po aplikovaní poškodenia aktuálne zdravie cieľa menšie alebo rovné 0, cieľ zomrel. Zaznamená sa udalosť smrti do **combat_log**, nastaví sa *is_alive* cieľa na FALSE v tabuľke combat_participant.

Použitie

Funkciu p_cast_spell je možné zavolať s ID castera, ID cieľa, ID kúzla a ID súboja:

SELECT p_cast_spell(102, 203, 304, 101); --- Tento príkaz simuluje zoslanie kúzla s ID 304 postavou s ID 102 na postavu s ID 203 v súboji s ID 101.

Diagram



Obrázok 7 Diagram aktivít UML - p_cast_spell

Zoznam navrhnutých indexov

- **idx_character_attr_pair**
  Index na tabuľke character_attribute pre dvojicu (character_id, attribute_id), aby sa zrýchlilo vyhľadávanie atribútov konkrétnej postavy.
- **idx_combat_participant_alive**
  Index na tabuľke combat_participant pre kombináciu (combat_id, is_alive, character_id), ktorý optimalizuje rýchle získavanie živých alebo mŕtvych účastníkov v konkrétnom boji.
- **idx_inventory_item_is_equipped**
  Index na tabuľke inventory_item pre trojicu (character_id, item_id, is_equipped), ktorý zrýchľuje vyhľadávanie vybavených (equipped) predmetov postavy.
- **idx_combat_item_is_taken**
  Index na tabuľke battleground_item pre kombináciu (combat_id, item_id, is_taken), ktorý umožňuje rýchle zistenie, ktoré predmety boli v boji už zdvihnuté alebo ešte dostupné.
- **idx_round_time**
  Index na tabuľke round pre stĺpce (start_time, end_time), ktorý urýchľuje dotazy pracujúce s časovým rozsahom kola.

Hlavné rozdiely oproti predchádzajúcej verzii

Úpravy logického modelu

1. **Nová vlastnosť "Action" pre itemy:** Každý predmet teraz obsahuje informáciu o tom, akú akciu ovplyvňuje — či zvyšuje spôsobené poškodenie, znižuje spotrebu akčných bodov alebo poskytuje liečenie.
2. **Časové údaje pre boje:** K tabuľke Combat boli pridané nové polia na zaznamenanie presného času začiatku a konca boja.
3. **Väzba medzi kolami a bojmi:** Model bol upravený tak, že každé kolo (Round) je priradené konkrétnemu boju (Combat), nie naopak.
4. **Rozšírenie stavov postáv a logov:** Pribudli nové stavy, napríklad postava môže byť v stave idle (neaktívna). Zároveň sa rozšírili typy udalostí v logoch (Combat_log) o záznamy ako login, round_start, round_end a winner.

Výpočty

Boli zavedené nasledujúce modifikácie vzorcov pre výpočty v systéme:

1. **Efektívne poškodenie**
   effective_damage := spell_base_damage + (((1 + attribute_modifier_sum/20) + multiplicative_modifier)/20) + (1 + item_modifier_sum)

2. **Efektívna cena kúzla**
   effective_cost := spell_base_cost * spell_base_cost_modifier * (1 - attribute_modifier_sum) * (1 - item_modifier_sum)

3. **Redukcia poškodenia brnením (čiastočná):**
   spell_damage := spell_damage - (armor_value / 20)

Pokyny na spustenie

1. Spustenie create_db.sql (vytvorenie tabuliek)
2. Spustenie všetkých súborov začínajúcich na p_ (vytvorenie procedúr)
3. Spustenie init_db.sql (naplnenie tabuliek)
4. Spustenie testov v tomto poradí
    a. test_p_rest.sql
    b. test_p_enter_combat.sql
    c. test_p_loot_item.sql
    d. test_p_cast_spell.sql
    e. test_p_reset_round.sql
5. Vytvorenie views (súbory začínajúce na v_)