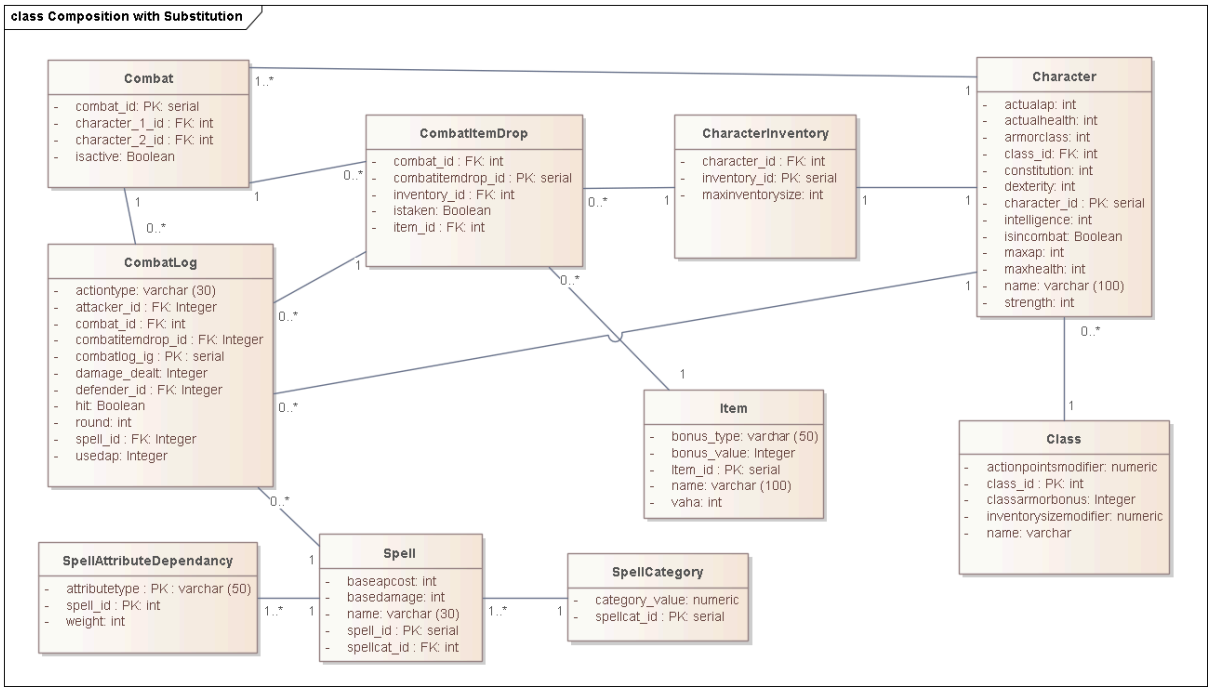
Zadanie 3

Dokumentácia

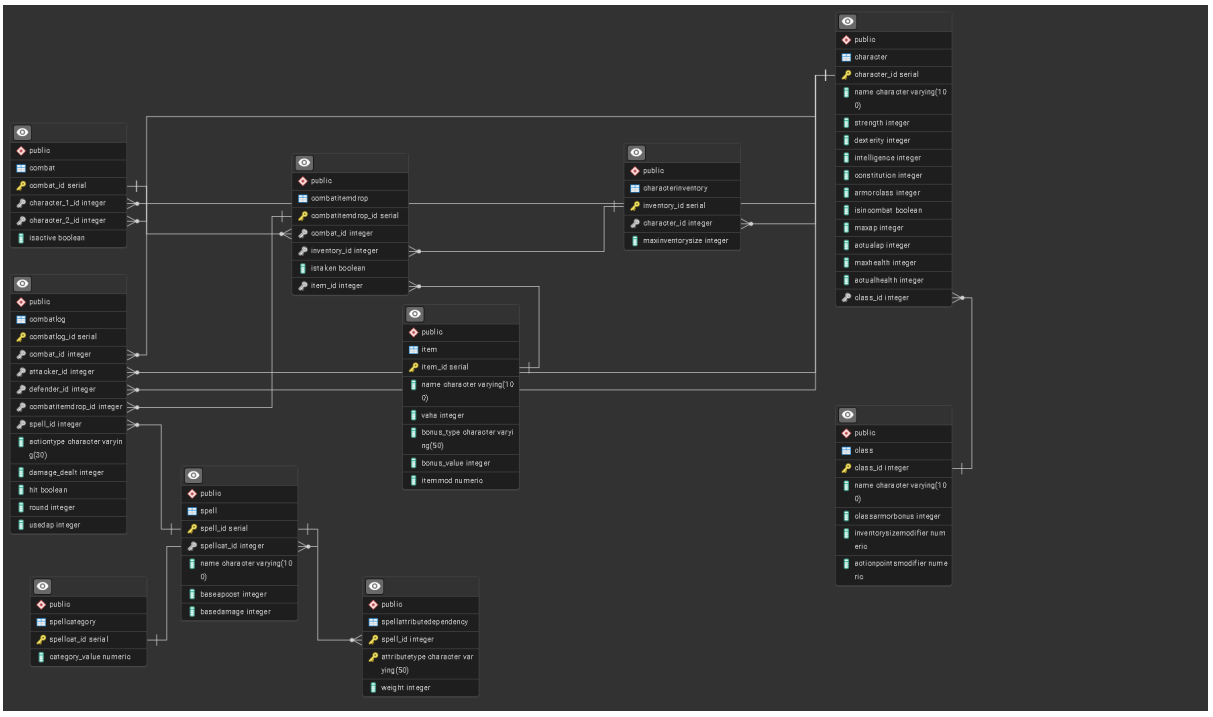
Databázové systémy
Fakulta informatiky a informačných technológií
Slovenská technická univerzita

Jozef Kušnier
ID Študenta : 120957
27.4.2025

Logicko-fyzické mapovanie modelu



Fyzický model



Podrobné vysvetlenie procesných tokov a prototypov postupov

1. sp_enter_combat (Vstup postavy do boja)

Postava môže vstúpiť do existujúceho alebo nového boja, ak nie je mŕtva a ešte nie je súčasťou iného boja. V prípade nového boja sa zároveň vygenerujú 4 náhodné predmety. Procedúra zabezpečí aktualizáciu tabuľky combat, nastaví postave v tabuľke character isincombat = true, doplní jej actualap a zaloguje vstup do combatlog.

Ukážka tabuľky combat

	combat_id [PK] integer	character_1_id integer	character_2_id integer	isactive boolean
1	99	1	[null]	true

Ukážka tabuľky combatlog

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	1	99	1	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]

Ukážka tabuľky character

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	1	99	1	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]

Ukážka tabuľky combatitemdrop - 4 náhodné predmety na zemi, pri vytvorení combatu

	combatitemdrop_id [PK] integer	combat_id integer	inventory_id integer	istaken boolean	item_id integer
1	1	99	[null]	false	1
2	2	99	[null]	false	3
3	3	99	[null]	false	2
4	4	99	[null]	false	4

2. sp_rest_character (Odpočívajúca postava)

Postava mimo boja si môže doplniť život a AP späť na maximum. Procedúra kontroluje, či postava nie je mŕtva a nie je v boji. V opačnom prípade spraví výnimku.

pred použitím *sp_rest_character*

	character_id [PK] integer	name character varying (100)	actualhealth integer	maxhealth integer
1	3	Medivh	1	60

po použití *sp_rest_character*

	character_id [PK] integer	name character varying (100)	actualhealth integer	maxhealth integer
1	3	Medivh	60	60

3. sp_reset_round + f_check_and_reset_round

Funkcia *f_check_and_reset_round* sa volá automaticky po každom volaní *sp_cast_spell* procedúri. Sleduje, či niektorá z postáv má dostatok AP na vykonanie akcie. Ak nie, spustí procedúru *sp_reset_round*, ktorá nastaví obom postavám actualap = maxap a zaznamená do tabuľky combatlog akciu typu RESET_ROUND.

Ukážka tabuľky combatlog

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	1	99	1	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]
2	2	99	2	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]
3	3	99	2	1	[null]	1	SPELL	0	false	1	4
4	4	99	[null]	[null]	[null]	[null]	RESET_ROUND	[null]	[null]	2	[null]

4. sp_cast_spell (Útok kúzlom)

Postava vykoná pokus o útok na cieľové ID pomocou kúzla. Procedúra overí existenciu boja a či obe postavy sú jeho účastníkmi. Ak má útočník dostatok AP podľa *f_effective_spell_cost*, vypočíta sa hod kockou – vygeneruje sa náhodné číslo 1 až 20 (d20). Túto hodnotu porovnáme s armorclassom cieľa. Ak je výsledok viac ako armorclass postavy, kúzlo zasiahne. Následne vyberieme dominantný atribút a jeho hodnotu pre daný spell, z tabuľky spellattributedependency. Damage vypočítame ako :

```
v_scaling := (v_attr_value * v_attr_weight) / 20.0;

v_damage := v_base_damage * (1 + v_scaling);
```

Táto hodnota sa následne odpočíta z ActualHealth cieľovej postavy. Akcia sa zapíše to tabuľky combatlog. Následne sa volajú procedúry *f_check_and_reset_round* a *f_check_and_end_combat* .

Ukážka tabuľky combatlog

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	4	99	2	1	[null]	3	SPELL	28	true	1	4

5. f_check_and_end_combat (Vyhodnotenie konca boja)

Funkcia sa volá po po každom volaní *sp_cast_spell* procedúri. Ak má niektorá postava HP ≤ 0, boj sa ukončí. Nastaví sa combat.isactive = false, isincombat = false pre oboch a itemi automaticky padnú na zem prostredníctvom sp_drop_item.

Ukážka tabuľky combatlog - ukončenie boja

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	1	99	1	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]
2	2	99	2	[null]	[null]	[null]	JOIN	[null]	[null]	1	[null]
3	3	99	2	1	[null]	1	SPELL	0	false	1	4
4	4	99	[null]	[null]	[null]	[null]	RESET_ROUND	[null]	[null]	2	[null]
5	5	99	2	1	[null]	1	SPELL	48	true	2	4
6	6	99	2	1	[null]	1	SPELL	0	false	2	4
7	7	99	2	1	[null]	1	SPELL	48	true	2	4
8	8	99	[null]	[null]	[null]	[null]	RESET_ROUND	[null]	[null]	3	[null]
9	9	99	2	1	[null]	1	SPELL	48	true	3	4
10	10	99	[null]	[null]	[null]	[null]	GAME_END	[null]	[null]	3	[null]

6. sp_loot_item (Zodvihnutie predmetu)

Postava môže zdvihnúť predmet zo zeme. Procedúra overí, či je ešte voľný (istaken = false), či postava má dostatok miesta v inventári a následne priradí item k inventáru. Atribút postavy (strength, dexterity, intelligence, constitution) sa posilnia o hodnotu bonusu daného predmetu. Následne sa zavolá procedúra *sp_update_character_stats*, ktorá aktualizuje atribúty postavi, ktoré závisia od strength, dexterity, intelligence, constitution. Konkrétne sú to atribúty armorclass, maxap, inventorysize. Celý proces sa zaloguje do combatlog. V tabuľke combatiemdrop sa priradí k predmetu ID characteru, ktorý daný item zobral.

Ukážka tabuľky combatitemdrop pred volaním *sp_loot_item*

	combatitemdrop_id [PK] integer	combat_id integer	inventory_id integer	istaken boolean	item_id integer
1	1	99	[null]	false	4

Ukážka tabuľky combatitemdrop po volaní *sp_loot_item*

	combatitemdrop_id [PK] integer	combat_id integer	inventory_id integer	istaken boolean	item_id integer
1	1	99	1	true	4

Ukážka tabuľky combatlog po volaní *sp_loot_item*

	combatlog_id [PK] integer	combat_id integer	attacker_id integer	defender_id integer	combatitemdrop_id integer	spell_id integer	actiontype character varying (30)	damage_dealt integer	hit boolean	round integer	usedap integer
1	3	99	1	[null]	1	[null]	ITEM_PICKUP	[null]	[null]	1	[null]

Ukážka tabuľky character pred volaním *sp_loot_item*

	character_id [PK] integer	name character varying (100)	strength integer	dexterity integer	intelligence integer	constitution integer	armorclass integer	isincombat boolean	maxap integer	actualap integer	maxhealth integer	actualhealth integer	class_id integer
1	1	Arthas	8	2	1	7	15	true	3	3	100	100	1

Ukážka tabuľky character po volaní *sp_loot_item* -> inteligence sa pridala hodnota +3

	character_id [PK] integer	name character varying (100)	strength integer	dexterity integer	intelligence integer	constitution integer	armorclass integer	isincombat boolean	maxap integer	actualap integer	maxhealth integer	actualhealth integer	class_id integer
1	1	Arthas	8	2	4	7	15	true	6	3	100	100	1

Ukážka tabuľky item (pridáva buff +3 inteligence)

	item_id [PK] integer	name character varying (100)	vaha integer	bonus_type character varying (50)	bonus_value integer
1	3	Magic Staff	3	Intelligence	3

7. sp_drop_item (Zahodenie predmetu)

Postava môže odhodiť predmet zo svojho inventára. Overuje sa, či skutočne vlastní daný item (cez inventory_id). Po úspešnej kontrole sa bonus predmetu odpočíta od atribútov postavy (strength, dexterity, intelligence, constitution). Následne sa zavolá procedúra *sp_update_character_stats*, ktorá prepočíta a aktualizuje odvodené atribúty armorclass, maxap a inventorysize podľa nových základných hodnôt. Predmet sa v tabuľke combatitemdrop odpojí od inventára (inventory_id = NULL) a nastaví sa istaken = false, čím sa vráti naspäť na zem a je opäť dostupný na zdvihnutie.

Ukážky tabuľiek sú identické ako pri *sp_loot_item ,* no idú v opačnej postupnosti.

8. f_effective_spell_cost (Výpočet efektívneho nákladu kúzla)

Táto funkcia je volaná pri procedúre *sp_cast_spell* . Vypočíta efektívny AP cost kúzla.Základom je hodnota baseapcost, ktorá sa násobí koeficientom kategórie kúzla (category_value) z tabuľky spellcategory. Upravuje sa podľa schopností postavy.Pre každé

kúzlo sú určené relevantné atribúty (napr. Strength, Intelligence...), ku ktorým sú priradené váhy v tabuľke spellatributedependency. Podľa toho sa vypočíta súčet modifikátorov schopností.

v_sum_attr=∑(atribút × váha)/100

Tento súčet sa použije pri znížení nákladu, pretože čím viac je postava prispôsobená kúzlu, tým menej AP stojí. Konečný náklad sa vypočíta takto.

v_effcost := v_baseapcost * v_catmod * (1 - v_sum_attr) * (1 - v_itemmod)

9. sp_create_character (Vytvorenie novej postavy)

Používa sa na založenie novej postavy a jej inventára. Na základe vstupných atribútov (strength, dexterity, intelligence, constitution) a triedy sa vypočíta armorclass, maxap, inventorysize, ktoré sa následne nastavia do tabuľky character a characterinventory. Classarmorbonus, actionpointsmodifier a inventorysizemodifier sa nachádzajú v tabuľke class.

ArmorClass je : 10 + (dexterity/2) + classarmorbonus

MaxAP je : round((dexterity + intelligence) * actionpointsmodifier

MaxInventorySize je : round((strength + constitution) * inventorysizemodifier

Ukážka tabuľky character po volaní *sp_create_character*

	character_id [PK] integer	name character varying (100)	strength integer	dexterity integer	intelligence integer	constitution integer	armorclass integer	isincombat boolean	maxap integer	actualap integer	maxhealth integer	actualhealth integer	class_id integer
1	3	Medivh	1	3	10	4	12	false	20	20	60	60	3

Ukážka tabuľky characterinventory

	inventory_id [PK] integer	character_id integer	maxinventorysize integer
1	3	3	4

Ukážka tabuľky class

	class_id [PK] integer	name character varying (100)	classarmorbonus integer	inventorysizemodifier numeric	actionpointsmodifier numeric
1	3	Mage	1	0.8	1.5

10. sp_update_character_stats (Prepočet odvodených hodnôt)

Procedúra sa volá pri zmene atribútov postavy (cez lootovanie a dropovanie itemov). Na základe aktuálnych hodnot a class sa znovu vypočítajú armorclass, maxap a inventorysize, a aktualizujú sa zodpovedajúce tabuľky.

ArmorClass je : 10 + (dexterity/2) + classarmorbonus

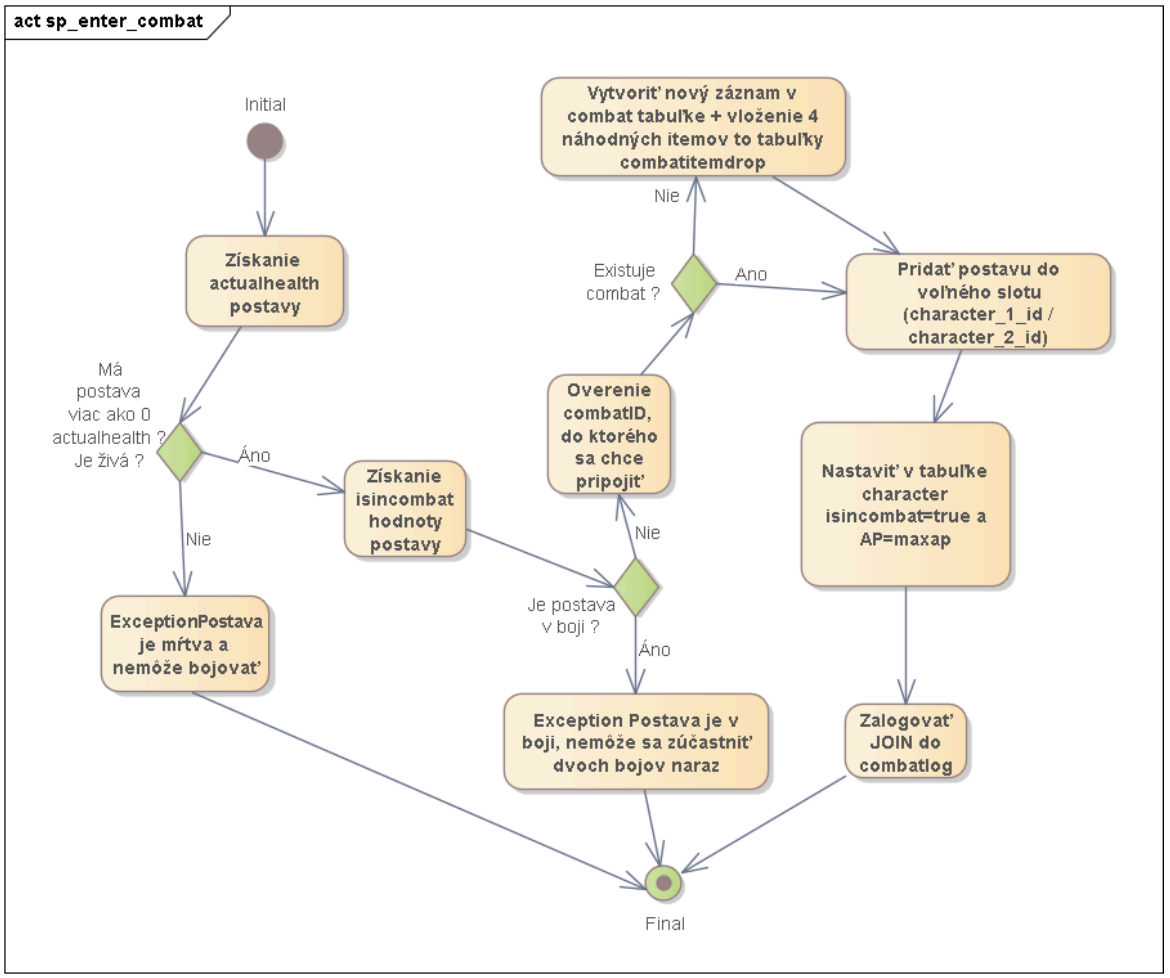
MaxAP je : round((dexterity + intelligence) * actionpointsmodifier

MaxInventorySize je : round((strength + constitution) * inventorysizemodifier

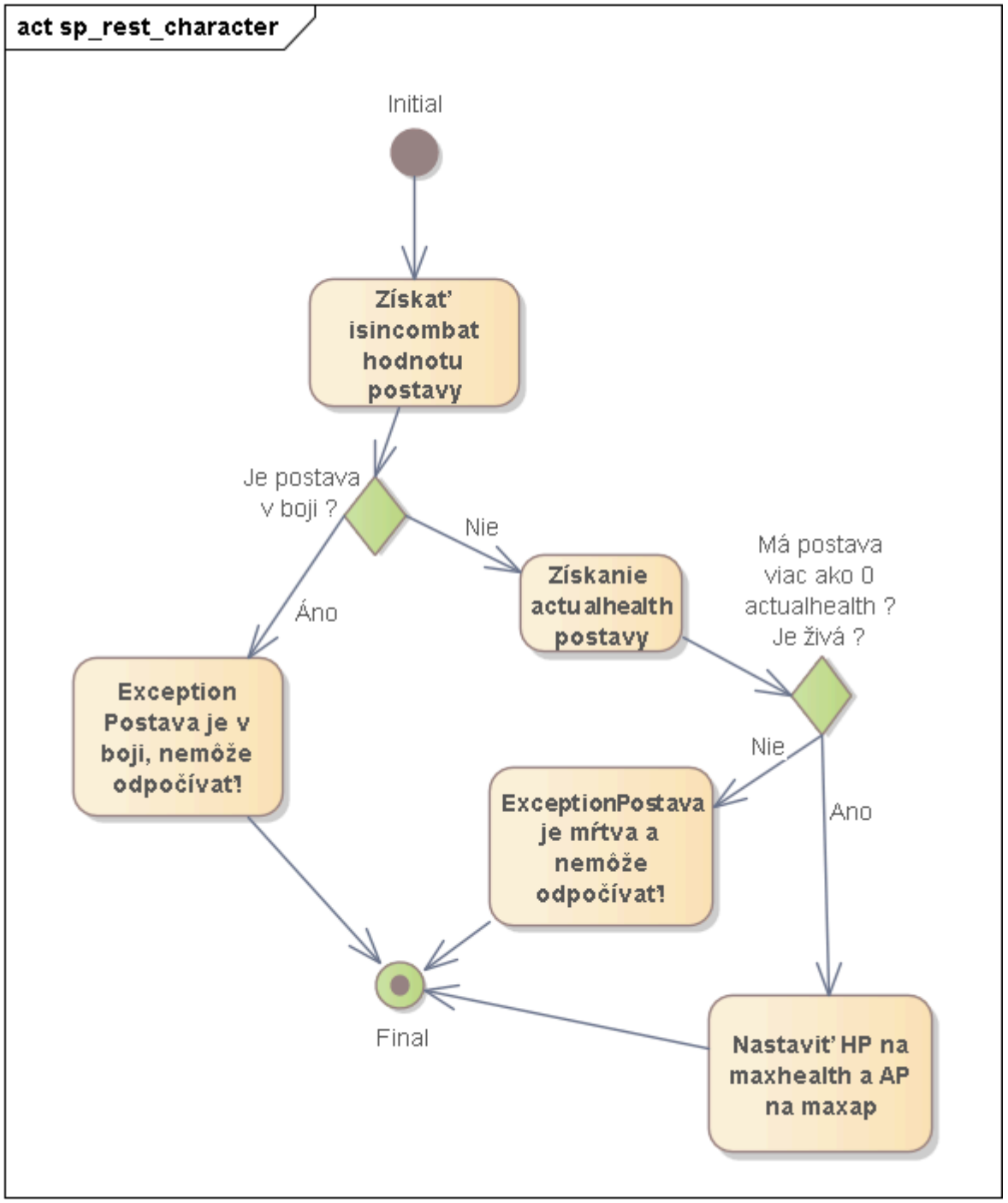
Ukážky tabuliek sú identické ako pri *sp_create_character*

Diagramy

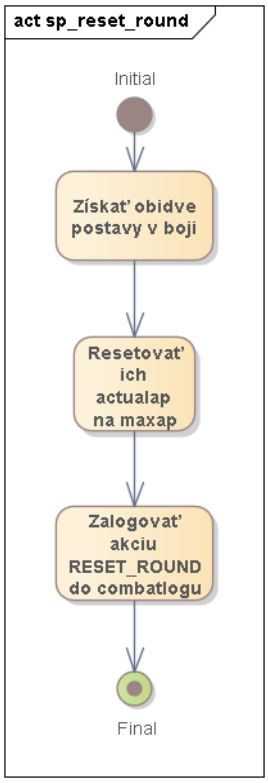
1. sp_enter_combat (Vstup postavy do boja)



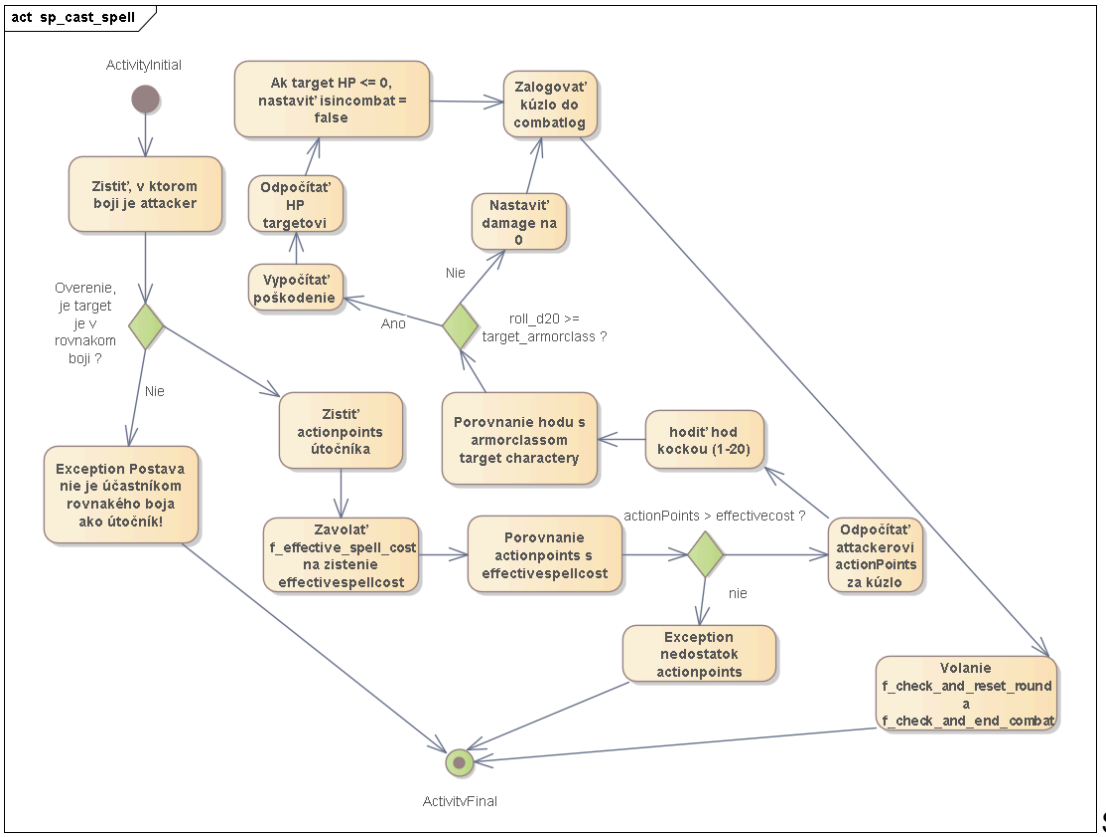
2. sp_rest_character (Odpočívajúca postava)



3. sp_reset_round

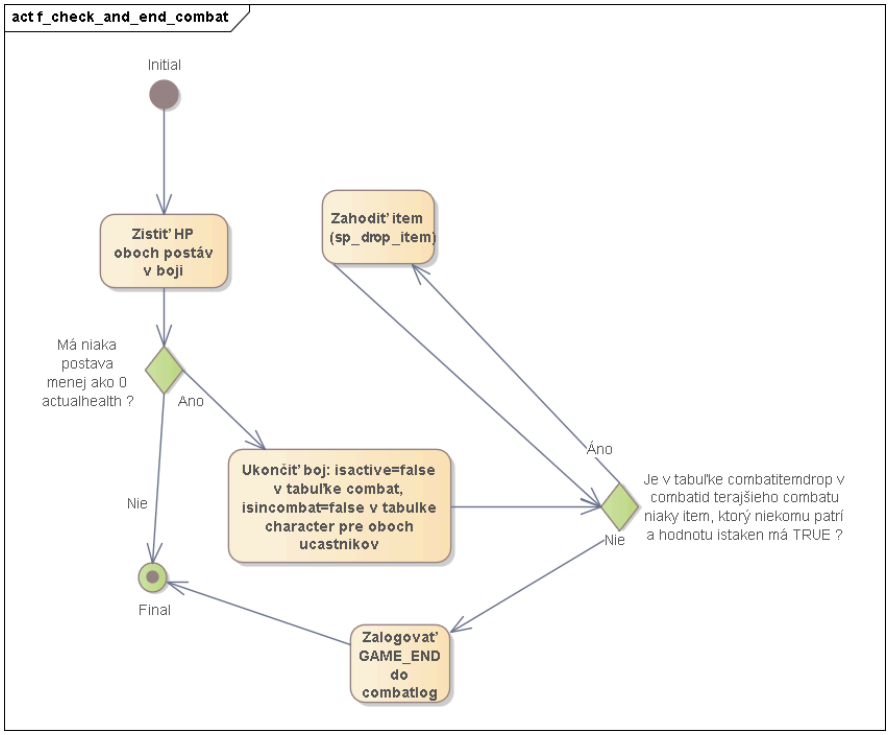


4. sp_cast_spell (Útok kúzlom)

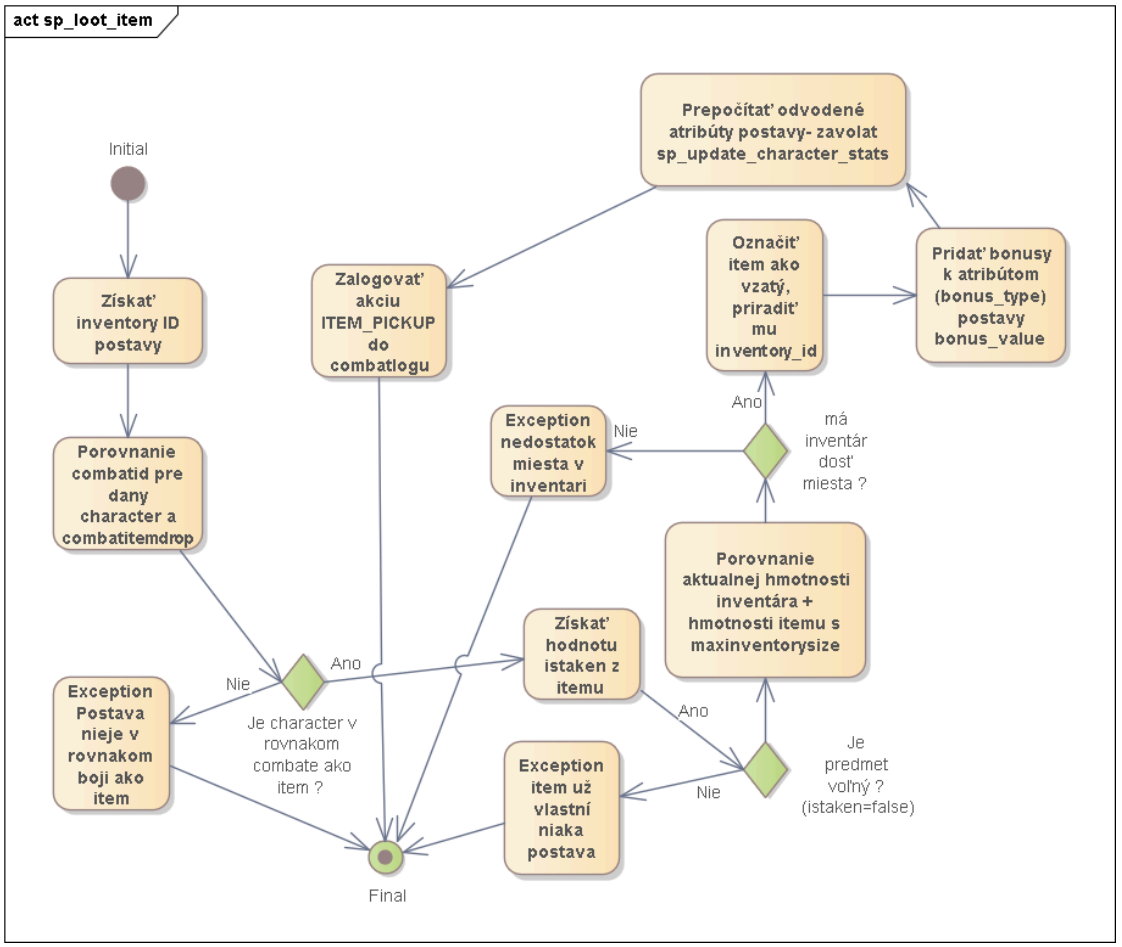


S

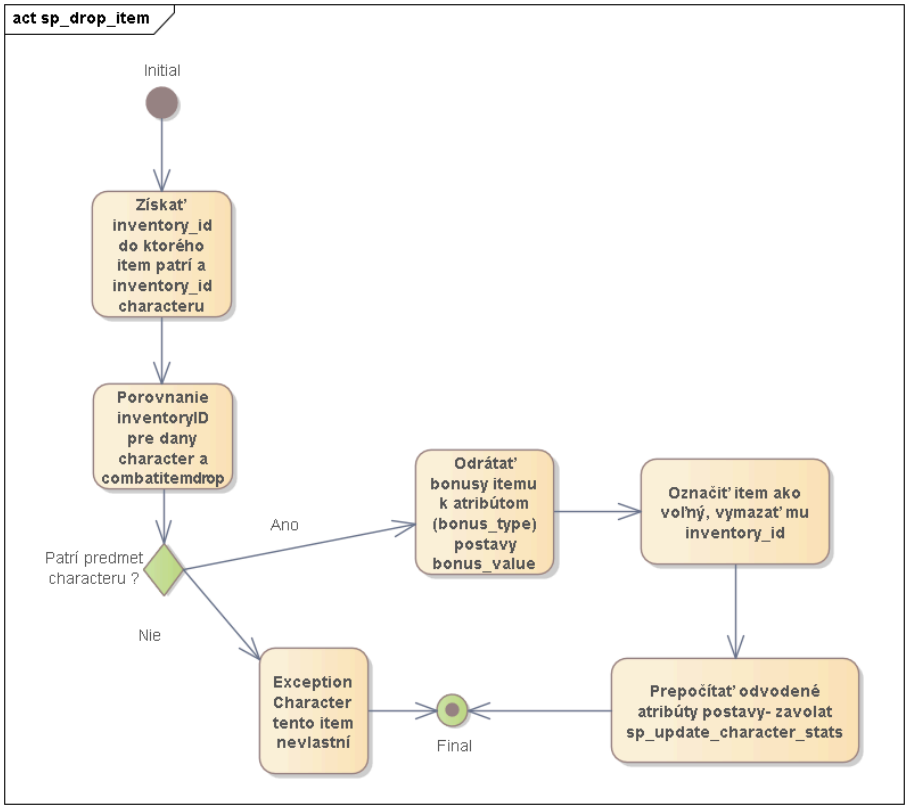
5. f_check_and_end_combat (Vyhodnotenie konca boja)



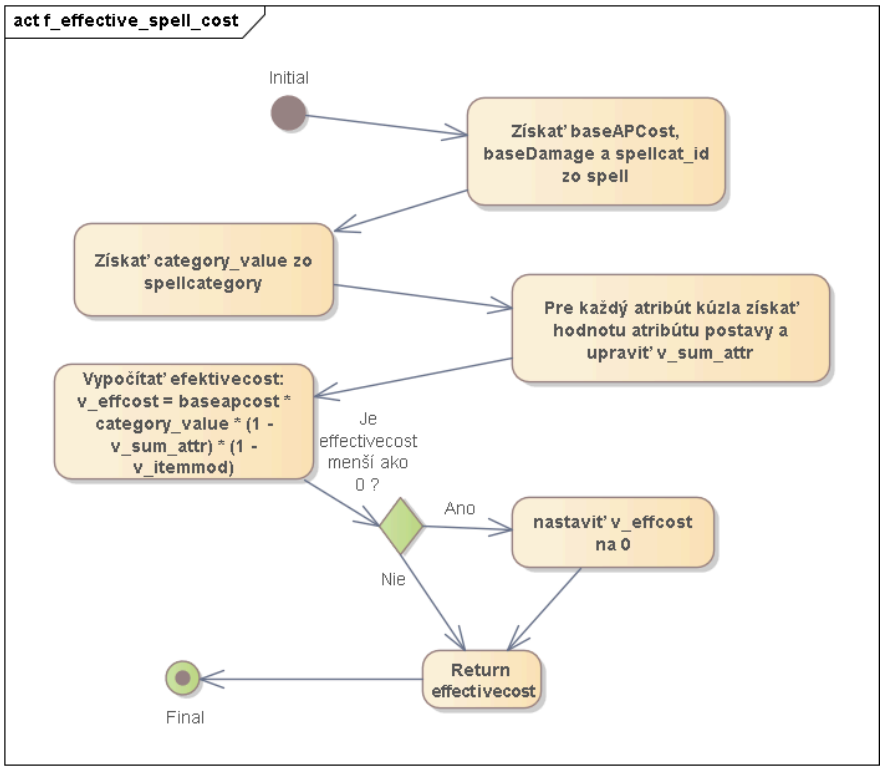
6. sp_loot_item (Zodvihnutie predmetu)



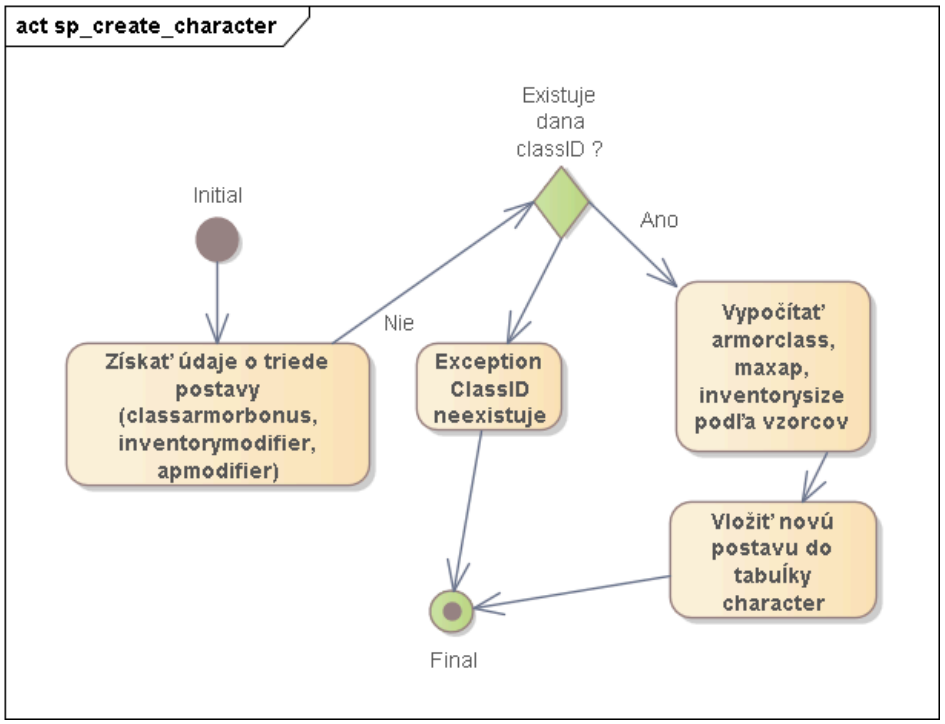
7. sp_drop_item (Zahodenie predmetu)



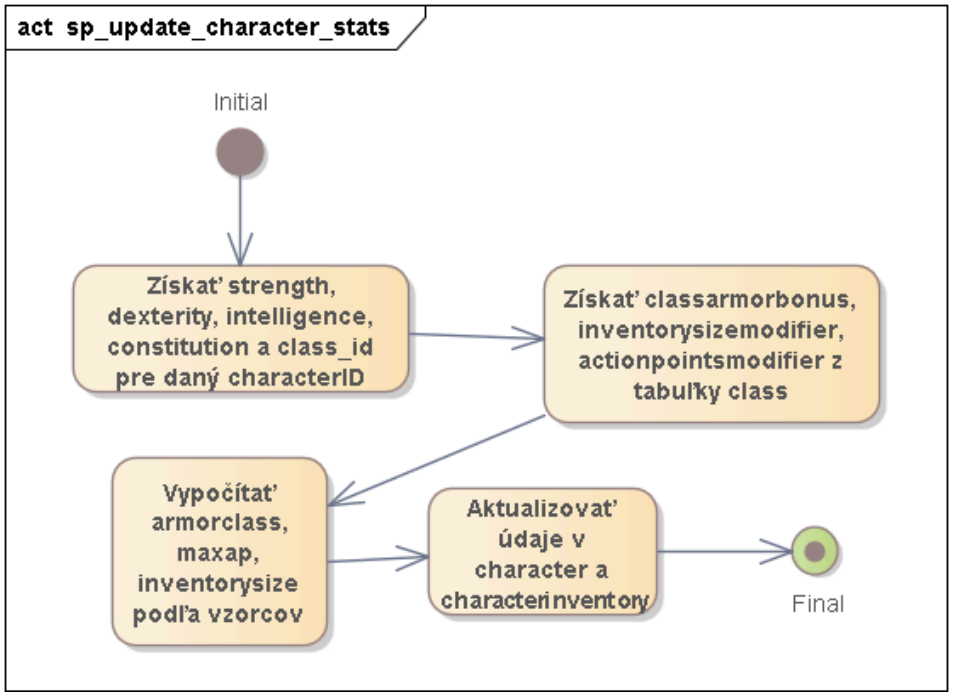
8. f_effective_spell_cost (Výpočet efektívneho nákladu kúzla)



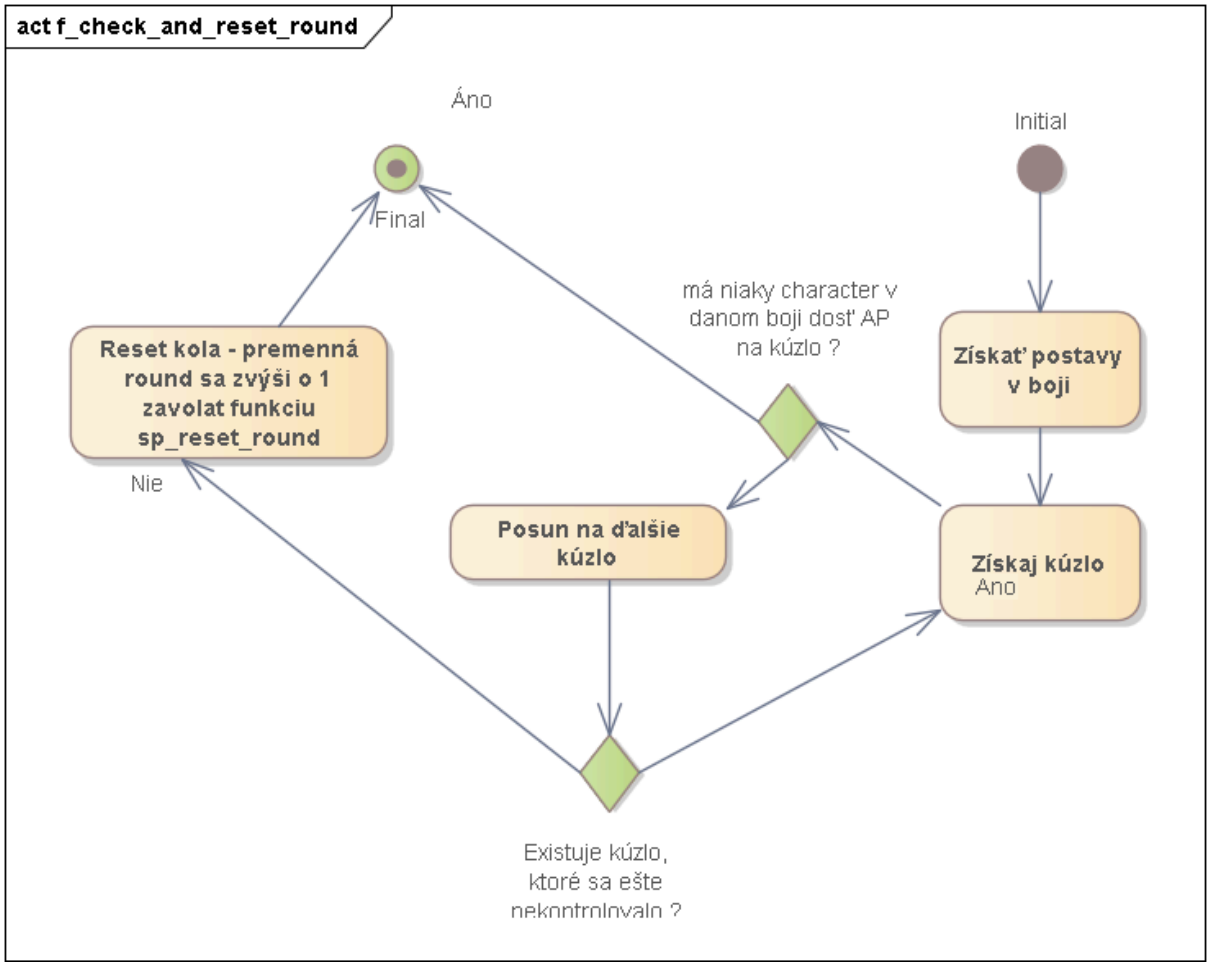
9. sp_create_character (Vytvorenie novej postavy)



10. sp_update_character_stats (Prepočet odvodených hodnôt)



11. f_check_and_reset_round



Zoznam vytvorených / navrhovaných indexov

1. Index na class_id v tabuľke character

Pre rýchle JOINY keď prerátavaš vlastnosti (sp_update_character_stats).

2. Index na character_id v tabuľke characterinventory

Pre rýchle lookupy inventáru (sp_loot_item, sp_drop_item, f_effective_spell_cost).

3. Index na character_1_id a character_2_id v tabuľke combat

Pre rýchle nájdenie, v ktorom boji je postava (sp_cast_spell, sp_enter_combat).

4. Index na combat_id v tabuľke combatitemdrop

Pre rýchle nájdenie itemov v boji (sp_loot_item, sp_drop_item).

5. Index na inventory_id v tabuľke combatitemdrop

Pre lookupy, keď zisťuješ, čo je v inventári (sp_loot_item, f_effective_spell_cost).

6. Index na combat_id v tabuľke combatlog

Pre zrýchlenie vyhľadávania akcií v boji (f_get_current_round, pohľady ako v_combat_state, v_most_damage).

7. Index na spellcat_id v tabuľke spell

Pre rýchly prístup ku kategórii kúzla (f_effective_spell_cost).

8. Index na spell_id v tabuľke spellattributedependency

Pre lookup atribútov kúzla (f_effective_spell_cost, sp_cast_spell)..

Rozdiely z pôvodného návrhu z prodchádzajúceho zadania

V predchádzajúcom návrhu sú fixne 3 spelly na kolo vs v novom zadaní je reset kola založený dynamicky na AP.V novom návrhu sa presnejšie zapisujú rôzne typy akcií do CombatLog.

V predchádzajúcom návrhu pri lootovaní sa hlavne kontroluje hmotnosť predmetov (váha), a aktualizuje sa inventár.

V novom návrhu lootovanie predmetu okamžite zvyšuje príslušný atribút postava tým dostáva buff (Strength, Dexterity, Intelligence, Constitution) a následne sa volá sp_update_character_stats, aby sa prepočítali odvodené vlastnosti (ArmorClass, MaxAP, InventorySize).

V pôvodnej databáze boli primárne kľúče ako CharacterID, ItemID, SpellID, CombatID, CombatItemDropID a InventoryID nastavené ako int. V aktuálnej databáze boli tieto kľúče zmenené na typ serial, čo znamená, že ich hodnoty sa generujú automaticky. Táto zmena zjednodušuje vkladanie nových záznamov a eliminuje riziko ručných kolízií ID.

Ďalšou zmenou je, že v tabuľke CombatLog sa stĺpec ActionType zmenil z typu integer na varchar(30). V pôvodnej verzii sa akcie zaznamenávali ako číselné hodnoty, zatiaľ čo teraz sú uložené ako textové názvy (napríklad „SPELL“, „ITEM_PICKUP“ alebo „JOIN“).

V tabuľke Item pribudli nové stĺpce bonus_type a bonus_value. Táto úprava umožňuje jednoduché pridávanie buffom z itemom postavám.

V rámci prepojení medzi tabuľkami došlo k spresneniu a rozšíreniu väzieb. Napríklad v CombatLog bol pridaný samostatný primárny kľúč combatlog_id, čo umožňuje lepšiu identifikáciu jednotlivých logov. Väzby medzi CombatItemDrop, CharacterInventory a Item sú teraz jednoznačne definované cez cudzie kľúče.

Pokyny na načítanie vzorových údajov a vykonanie akceptačných testov

1. Spustit celý súbor vytvorenie_databazy.sql . Tento súbor vytvorí prázdnu databázu, bez vložených hodnôt.
2. Spustit celý súbor index.sql
3. Spustiť celý súbor FunkcieAprocedury.sql . Su tam definovane funkcie, procedury aj pre view.
4. Spustit celý súbor VlozenieDoDatabazy.sql . Tento súbor vloží testovacie dáta do databázy. Tieto dáta sú pripravené na absolvovanie akceptačných testov. Tieto dáta nemožno upravovať, do robenia akceptačných testov, nakoľko testy rátajú s hodnotami, ktoré sme tam teraz vložili.
5. Otvoriť súbor useCase.sql .
6. Skript v súbore useCase.sql je nutné spúšťať riadok po riadku zhora nadol, tak ako je napísaný. Je potrebné ísť za radom, ako je kód písaný, keďže zmeny v databáze medzi testami na seba nadväzujú a väčšina testov ráta s predchádzajúcimi zmenami v databáze. Pre bližšie vysvetlenie, napríklad testy na procedúru *sp_drop_item* sa spoliehajú na to, že v predchádzajúcich testoch sme použil procedúru *sp_loot_item,* takže character ten item reálne má v inventári. Z tohoto dôvodu nie je možné preskakovať postupnosť testov a riadkov.

Príklad postupnosti testovania

krok 1

```
-- -- -- -- -- USE CASE: PRIDANIE POSTAV DO BOJA -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
--Pre funkcnost testov ich treba vykonavat v poradi, v ktorom su napisane. Databaza je nato pripravena (nieje pripravena na ir

-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```

krok 2

```
-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```

krok 3

```
-- -- -- -- -- USE CASE: PRIDANIE POSTAV DO BOJA -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
--Pre funkcnost testov ich treba vykonavat v poradi, v ktorom su napisane. Databaza je nato pripravena (nieje pripravena na in

-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```

krok 4

```
-- -- -- -- -- USE CASE: PRIDANIE POSTAV DO BOJA -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
--Pre funkcnost testov ich treba vykonavat v poradi, v ktorom su napisane. Databaza je nato pripravena (nieje pripravena na

-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```

krok 5

```
-- -- -- -- -- USE CASE: PRIDANIE POSTAV DO BOJA -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
--Pre funkcnost testov ich treba vykonavat v poradi, v ktorom su napisane. Databaza je nato pripravena (nieje pripravena na in

-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```

krok 6

```
-- -- -- -- -- USE CASE: PRIDANIE POSTAV DO BOJA -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
--Pre funkcnost testov ich treba vykonavat v poradi, v ktorom su napisane. Databaza je nato pripravena (nieje pripravena na in

-- Pozitívny test - pridanie postavy do boja a vytvorenie boju
-- OČAKÁVANÝ VÝSTUP: postava ID=1 sa pridá do boja 99, vznikne záznam v `combat`, `combatlog`, `combatitemdrop`
SELECT sp_enter_combat(99, 1);
SELECT * FROM combat;           -- kontrola či je postava s id_1 v boji a či sa boj vytvoril
SELECT * FROM combatlog;        -- kontrola logu vstupu do boja
SELECT * FROM combatitemdrop;   -- kontrola pridaných itemov do boja

-- Negatívny test - pridanie do boja postavi, ktora uz v boji je
-- OČAKÁVANÝ VÝSTUP: chyba - postava je už v boji = ERROR:  Postava 1 je už v boji!
SELECT sp_enter_combat(98, 1);
SELECT * FROM combat;           -- kontrola ze postava nieje v boji 98
```