Database design for the core of a tactical RPG system

Andrii Dokaniev (AIS id: 128365)

Contents

Zmeny oproti predchádzajúcemu návrhu 4
Kľúčové rozdiely procesov 4
1. **Čo sa stane, keď postavy zosielajú kúzlo vnútri boja? Ktoré entity/záznamy sa vytvoria? Aké atribúty sa menia? Čo sa musí skontrolovať?** 4
6. **Ako postava získa predmet a skontroluje limity inventára?** 5
Relačná štruktúra fyzického modelu 6
Funkcie a procedúry 8
sp_cast_spell 8
sp_rest_character 10
sp_enter_combat 10
sp_loot_item 10
f_effective_spell_cost 10
sp_reset_round 11
sp_initialize_combat_items 11
sp_init_test_data 12
sp_reset_test_data 12
sp_physical_attack 12
sp_drop_item 12
sp_wake_up_character 13
Views 14
v_combat_state 14
v_most_damage 14
v_strongest_characters 14
v_combat_damage 14
v_spell_statistics 15
v_player_statistics 15
v_most_spells_unlocked 15
v_most_items_spawned_in_combat 15
v_combat_log_statistics 15

Indexy 16
Akceptačne testy 18
References: 20

Zmeny oproti predchádzajúcemu návrhu

Boli zmenené atribúty jednotných polí, pridané nové atribúty pre používateľov a postavy, a odstránené nevyužiteľné atribúty. Boli opravené vzorce, aby herné predmety mali tiež nejaký vplyv.

Všetky rozdiely oproti pôvodnému návrhu z predchádzajúceho zadania boli zaznamenané v dokumentácii s ohľadom na rôzne fázy implementácie. Kľúčové rozdiely zahŕňajú napríklad úpravu vzorca na zásah alebo na **damage** bonusy spôsobené predmetmi, ako aj implementáciu logiky predmetov do systému.

Taktiež je možné vidieť tieto zmeny aj na príslušnej štruktúre databázy.

Kľúčové rozdiely procesov

Bola pridaná logika pre predmety postav.

1. **Čo sa stane, keď postavy zosielajú kúzlo vnútri boja? Ktoré entity/záznamy sa vytvoria? Aké atribúty sa menia? Čo sa musí skontrolovať?**

To by sme mohli počítať podlá vzorca:

$$Attack = d20 + AttackBonus + EquipmentModifiers.(\text{"}\mathbf{Intelligence}\text{ "})$$

Kde **AttackBonus** predstavuje **Intelligence**. $EquipmentModifiers$ je súčet modifikátorov **Intelligence.**

Na zásah hráča kúzlom musíme počítať podlá vzorca:

$$ArmorClass = 10 + (\frac{Dexterity}{2} + EquipmentModifiers.Type(\text{"armor"})) + ClassArmorBonus$$

Útok zasiahne, ak $Attack > ArmorClass$.

Ak kúzlo zasiahne súpera, musíme vypočítať aj spôsobené poškodenie (damage):

$$Damage = BaseDamage + \sum(\text{damage} * (1 + \frac{ConfiguredAttribute}{20})$$

Kde

- **BaseDamage** je základný damage kúzla.
- **ConfiguredAttribute** predstavuje atribút(atribút postavy na, ktorú bolo zaslane dane kúzlo), na ktorý konkréte kúzlo ma vplyv.

Fyzicky útok sme počítame inak.

AC pri zaslaní počítame rovnako,

$$Attack = Strength + EquipmentModifiers.Type(\text{"strength"})$$

Na rozdiel od kúziel tento útok vždy zasiahne cieľ, avšak môže spôsobiť cieľu výrazne menší **damage**.

$$TotalDamage = max(Attack - (AC * 0.2), 0)$$

Cena kúziel je definovaná nasledujúcim vzorcom:

$$EffectiveCost = BaseCost * Category.Modifier(1 - \frac{SelectedAttribute}{100})(1 - ItemModifiers) \quad (2)$$

Kde

- $BaseCost$
- $Category.Modifier$ – modifikátor klasu. Type (“action_points”)
- $SelectedAttribute$ — atribbut postavy(Intelligence)
- $ItemModifiers$ — modifikator, ktorý dáva predmet. Type (“action_points”)

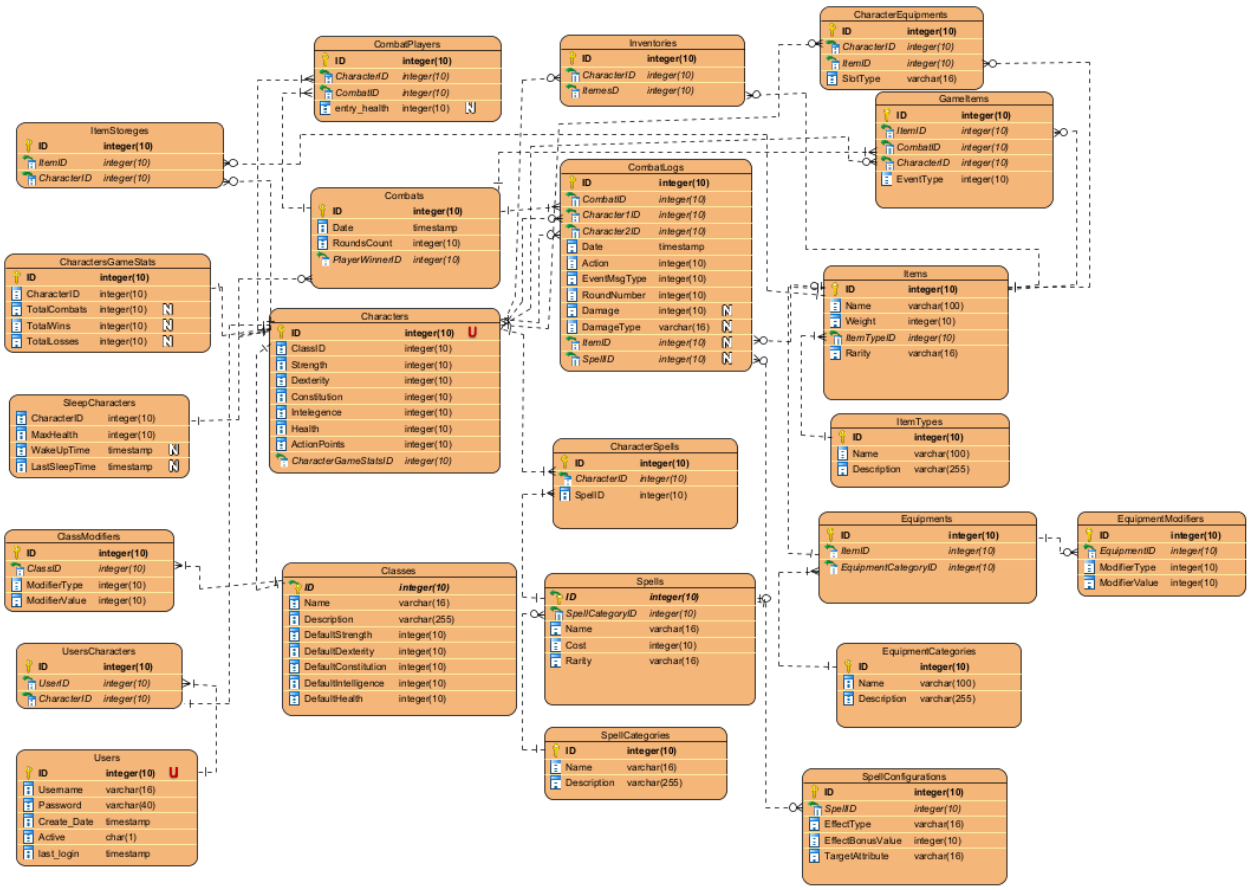
6. **Ako postava získa predmet a skontroluje limity inventára?**

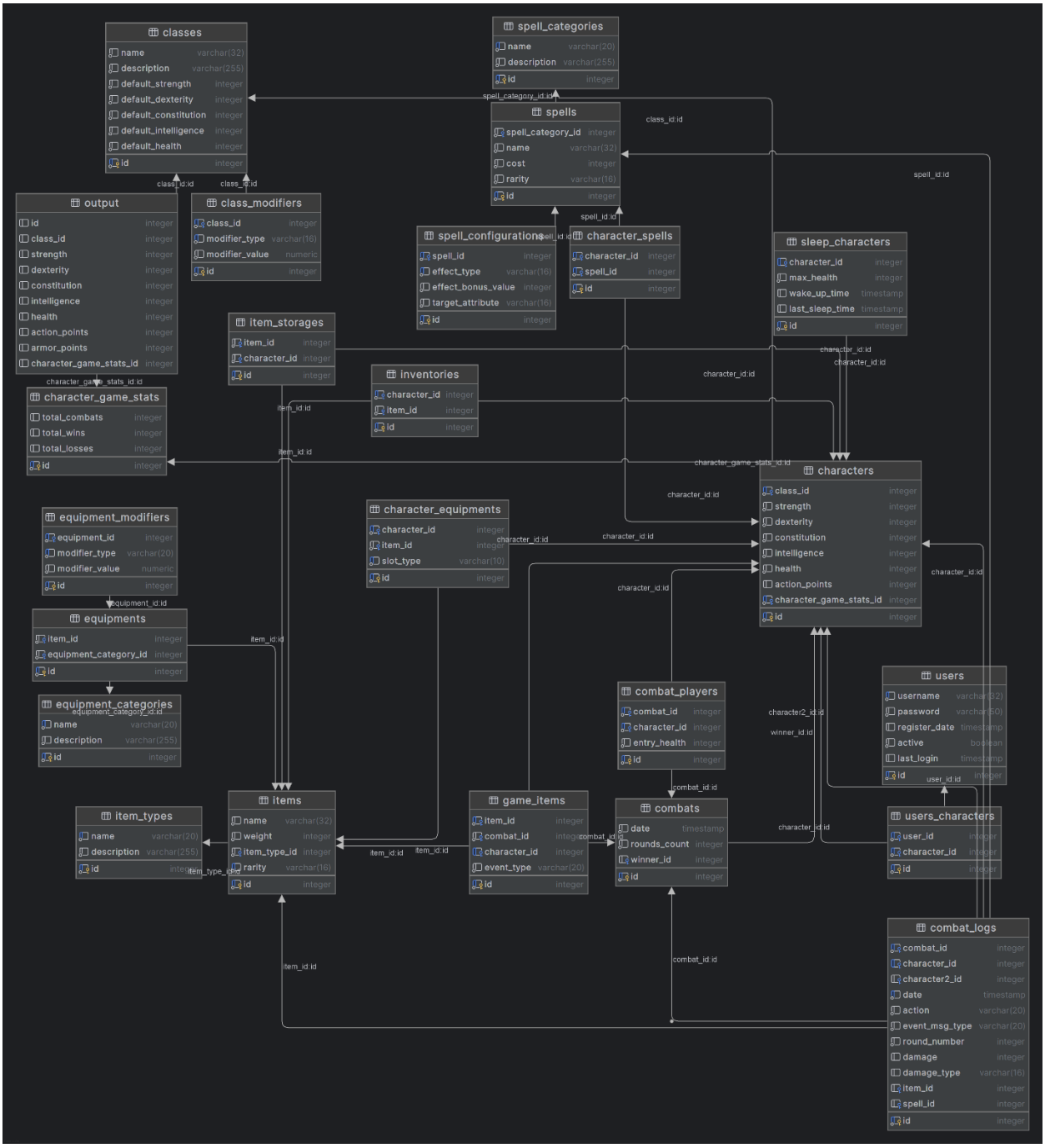
$$MaxInventoryWeight = (Strength + Constitution)ClassModifiers.Type(\text{Inventory}) + EquipmentModifiers.Type(\text{"Inventory"})$$

Kde:

- **Strength** a **Constitution** sa berú z tabuľky **Characters**.
- **ClassModifiers** z tabuľky **ClassModifiers**.
- $EquipmentModifiers$ je súčet modifikátorov z tabuľky **CharacterEquipments**

Relačná štruktúra fyzického modelu




classes
name varchar(32)
description varchar(255)
default_strength integer
default_dexterity integer
default_constitution integer
default_intelligence integer
default_health integer
id integer
spell_categories
name varchar(20)
description varchar(255)
id integer
spells
spell_category_id integer
name varchar(32)
cost integer
rarity varchar(16)
id integer
output
id integer
class_id integer
strength integer
dexterity integer
constitution integer
intelligence integer
health integer
action_points integer
armor_points integer
character_game_stats_id integer
class_modifiers
class_id integer
modifier_type varchar(16)
modifier_value numeric
id integer
spell_configurations
spell_id integer
effect_type varchar(16)
effect_bonus_value integer
target_attribute varchar(16)
id integer
character_spells
character_id integer
spell_id integer
id integer
sleep_characters
character_id integer
max_health integer
wake_up_time timestamp
last_sleep_time timestamp
id integer
item_storages
item_id integer
character_id integer
id integer
inventories
character_id integer
item_id integer
id integer
character_game_stats
total_combats integer
total_wins integer
total_losses integer
id integer
characters
class_id integer
strength integer
dexterity integer
constitution integer
intelligence integer
health integer
action_points integer
character_game_stats_id integer
id integer
character_equipments
character_id integer
item_id integer
slot_type varchar(10)
id integer
equipment_modifiers
equipment_id integer
modifier_type varchar(20)
modifier_value numeric
id integer
equipments
item_id integer
equipment_category_id integer
id integer
equipment_categories
name varchar(20)
description varchar(255)
id integer
combat_players
combat_id integer
character_id integer
entry_health integer
id integer
users
username varchar(32)
password varchar(60)
register_date timestamp
active boolean
last_login timestamp
id integer
item_types
name varchar(20)
description varchar(255)
id integer
items
name varchar(32)
weight integer
item_type_id integer
rarity varchar(16)
id integer
game_items
item_id integer
combat_id integer
character_id integer
event_type varchar(20)
id integer
combats
date timestamp
rounds_count integer
winner_id integer
id integer
users_characters
user_id integer
character_id integer
id integer
combat_logs
combat_id integer
character_id integer
character2_id integer
date timestamp
action varchar(20)
event_msg_type varchar(20)
round_number integer
damage integer
damage_type varchar(16)
item_id integer
spell_id integer
id integer

Funkcie a procedúry

sp_cast_spell

Táto procedúra slúži na útok na súpera pomocou zoslania kúzla. Ako vstupne parameter berie:

- p_caster_id – ID hráča, ktorý zosiela kúzlo
- p_target_id – ID hráča, na ktorého bol prevedený útok
- p_spell_id – ID kúzla

Procedúry nevracajú hodnoty, je to vidno z časti `RETURNS VOID AS $$`.

Ďalej overujeme, či postava má potrebný počet Action Points na zoslanie kúzla. Ak nie, vyvoláme výnimku:

```
RAISE EXCEPTION 'Caster does not have enough AP for casting this spell.';
```

Ďalej nasleduje podobné overenie, či hráč má k dispozícii dané kúzlo a či hráč, na ktorého útočíme, je ešte živý alebo ci hráč nechce spraviť samovraždu.

Ak boli všetky overenia splnené, môžeme pokračovať výpočtami na aktualizáciu AP príslušnej postavy, ktorá zosiela kúzlo.

```
UPDATE characters
SET action_points = action_points - f_effective_spell_cost(p_spell_id,
p_caster_id)
WHERE id = p_caster_id;
```

V danej časti je použitá funkcia, ktorej návratová hodnota je efektívna cena kúzla. Viac o tejto funkcii bude uvedené v nasledujúcich kapitolách popisu procedúr a funkcií.

Po dokončení aktualizácie AP kontrolujeme, či zoslané kúzlo zasiahlo cieľ. Robíme to tak, že vygenerujeme hod d20 (náhodné číslo od 1 do 20) a pripočítame ho k sile zásahu podľa vzorca:

$$d20.roll + (caster.intelligence + (SUM(caster.equipments.modifier\_type('intelligence'))))$$

Kde prepočítavame celkový počet inteligencie postavy. K ďalším možnostiam patrí pridanie bonusu na zásah podľa triedy postavy. Je to viac komplikovanejšie, preto tá možnosť bola vynechaná ako alternatíva k ďalšej aktualizácii.

Nasleduje výpočet Armor Pointov konečného cieľa kúzla. Na výpočet tejto hodnoty použijeme vzorec:

$$ArmorClass = (10 + (Dexterity/2) +$$

$$+ SUM(target.equipments.modifier\_type('armor')) * \\ * class\_modifier.modifier\_type('armor')$$

A overíme, či hráč trafil, porovnaním hodnoty Armor cieľa so silou zásahu.

$$hit = attack.attack\_strength >= target\_ac.armor\_points$$

Ak výsledok sa rovná ($hit == FALSE$), tak hovoríme že postava netrafila ciele,

($damage = 0$) a ideme zapĺňať combat log. Inak pristúpime k výpočtu poškodenia, ktoré spôsobilo zoslané kúzlo. Výpočet je určený podlá vzorca:

$$damage = BaseDamage + SUM(EffectBonusValue * (1 + TargetAttribute/20))$$

EffectBonusValue predstavuje silu efektu aplikovanú na konkrétny atribút (**TargetAttribute**). Medzi možné atribúty patria napríklad:

```
CASE target_attribute
    WHEN 'strength' THEN strength
    WHEN 'dexterity' THEN dexterity
    WHEN 'constitution' THEN constitution
    WHEN 'intelligence' THEN intelligence
    WHEN 'health' THEN health
END / 20::NUMERIC
```

Tabuľka **Spell Configurations** obsahuje stĺpec **effect_type**, ktorý môže byť **'base_damage'** alebo **'damage'**, Na základe tohto atribútu určujeme, či ide o základné poškodenie $BaseDamage$, alebo bonusový efekt kúzla, ktorý ovplyvňuje konkrétny atribút ciele.

Po nasledujúcej aktualizácii počtu zdravia hráča zapíšeme do *combat log* výsledok danej udalosti. Keďže zdravie nemôže prekročiť zápornú hranicu, musíme to ošetriť:

```
UPDATE characters
SET health = GREATEST(health - (SELECT total_damage FROM damage), 0)
WHERE id = p_target_id
RETURNING health
```

A zaplniť výsledok udalosti, kde podľa zdravia ciele, ktoré zostalo určime aký výsledok udalosti zapísať(event_msg_type):

```
CASE
    WHEN (SELECT health FROM health_update) = 0 THEN 'kill'
    WHEN (SELECT is_hit FROM hit) = TRUE THEN 'success'
    ELSE 'failure'
END
```

Do *combat_log* sa rovnako zapíšu informácie o ID súboja, ID prvej postavy, ID cieľa, dátume, type akcie, aktuálnom kole, type poškodenia, celkovom poškodení, ID predmetu a ID kúzla.

sp_rest_character

Táto procedúra je určená na odpočívanie mimo boja. Obnovuje zdravie hráča(**p_character_id**) na maximum. Maximálna hodnota zdravia je uložená v tabuľke **sleep_characters** a aktuálne v **characters**. Po spojení týchto tabuliek môžeme aktualizovať zdravie.

Aktualizujeme maximálny počet **AP** pomocou vzorca:

$$MaxActionPoints = (Dexterity + Intelligence)ClassModifier$$

V tomto vzorci **Dexterity** a **Intelligence** predstavujú príslušné charakteristiky postavy, zatiaľ čo **ClassModifier** je hodnota určujúca, akým tempom bude rásť AP na základe charakteristík postavy ($Dexterity + Intelligence$).

Informácie o odpočívaní nemusíme zapisovať do ***combat log***.

sp_enter_combat

Procedúra, ktorá pridá hráča zadaného ako parameter (**p_character_id**) do príslušného boja (**p_combat_id**).

Zapíše hráča do relácie **combat_players** s informáciami o **ID aktuálnej hry**, **ID postavy** a **počte zdravia** pri vstupe do boja. Aktualizuje **AP** hráča a zapíše do ***combat log*** informácie o tom, ktorý hráč vstúpil do boja.

sp_loot_item

Procedúra určená na pridanie hráča do inventára predmetu z bojovej oblasti.

Najprv overí, či je daný predmet k dispozícii (je voľný). Ak áno, vypočíta maximálny počet miest v inventári, určí aktuálnu váhu inventára, vezme váhu predmetu, ktorý chceme zobrať, a pripočíta ju k aktuálnej váhe.

Pri nasledujúcom porovnaní s maximálnou váhou zistí, či môže postava pridať predmet do inventára, alebo nie.

Ak má postava voľné miesto, pridáme nový predmet do inventára a aktualizujeme stav predmetu v tabuľke predmety danej hry.

Výsledok pridania predmetu zapíšeme do **combat_log** s id **predmeta a akciou 'loot_item'**

f_effective_spell_cost

Funkcia slúži na výpočet ceny kúzla (**p_spell_id**) pre určitú postavu (**p_caster_id**).

Vracia hodnotu typu **NUMERIC**, ktorá reprezentuje cenu kúzla. Tato hodnota je vypočítaná podľa vzorca:

$$EffectiveCost = BaseCost * Category.Modifier(1 - SelectedAttribute/100)(1 - ItemModifiers)$$

Na zatáčku je vypočítane bonusy pre **AP** od **equipment** postavy(napr. magická palica a iné.). Tato hodnota je vypočítaná ako suma takých modifikátorov predmetov $\boldsymbol{ItemModifiers}$**.** Hodnotu $\boldsymbol{BaseCost}$ zoberieme z tabuľky **spells** zo stĺpca **cost**. $\boldsymbol{SelectedAttribute}$ je atribút postavy – **intelligence.** A $\boldsymbol{Category.Modifier}$ je vypočítaný ako suma všetkých modifier_value, kde modifier_type = 'action_points' pre daného hráča. Je ošetrene aby $\boldsymbol{Category.Modifier}$ nebol null, v takom prípade to bude hodnota 1.

sp_reset_round

Procedúra na prechod na nasledujúce kolo v boji. Aktualizuje aktuálne kolo o jednu úroveň.

$$rounds\_count = rounds\_count + 1$$

Taktiež pre všetkých hráčov nastaví **AP** na maximálnu hodnotu pomocou vzorca:

$$action\_points = (dexterity + intelligence) * class\_modifiers.modifier\_value$$

Kde $\boldsymbol{modifier\_value}$ je hodnota, ktorá určuje ako rýchlo bude rast **AP** podľa určitých charakteristík postavy ($\boldsymbol{dexterity + intelligence}$)

Po aktualizácie kola, zaznamená to do **combat_log** v formáte:

$$ID, CombatID, null, null, CURRENT\_TIMESTAMP, reset\_round, success, R\_num, null, null, null, null$$

Kde R_{num} je aktuálne aktualizovane kolo.

sp_initialize_combat_items

Procedúra na vygenerovanie predmetov do bojovej plochy podľa nasledovných pravdepodobností:

- **Common** – Lower Bound (n * m) krát predmetov na hru, kde m je náhodne číslo od 50% do 150%.
- **Uncommon** – 50% na každý predmet, Lower Bound($n/2$) krát.
- **Rare** – 30% na každý predmet na hru, $\min\left(\frac{n*30\%}{100\%}, 2\right)$ krát.
- **Epic** – 3% na predmet na hru, n krát.

- **Legendary** – 0.5% na predmet na hru, n krát.

sp_init_test_data

Procedúra na inicializáciu testovacích zoznamov.

sp_reset_test_data

Procedúra na odstránenie zoznamov zo všetkých tabuliek a ich opätovnú inicializáciu.

sp_physical_attack

Procedúra pre postavu, ktorá chce útočiť fyzicky.

Vzorec pre damage:

$$damage = GREATEST(strength + SUM(equipment.modifier\_value(\text{'}strength\text{'})) - armor * 0.2, 0)$$

Aktualizujeme zdravie:

$$GREATEST((health - damage), 0)$$

A dame to do **combat_log:**

```
VALUES (
    (SELECT combat_id FROM combat_players WHERE character_id =
p_attacker_id),
    p_attacker_id,
    p_target_id,
    CURRENT_TIMESTAMP,
    'attack',
    CASE
        WHEN (SELECT health FROM health_update) = 0 THEN 'kill'
        WHEN (SELECT d FROM total_damage) = 0 THEN 'miss'
        WHEN (SELECT d FROM total_damage) > 0 THEN 'success'
        ELSE 'failure'
    END,
    (
        SELECT rounds_count
        FROM combats
        WHERE id = (SELECT combat_id FROM combat_players WHERE character_id =
p_attacker_id)
    ),
    (SELECT d FROM total_damage),
    'physical'
);
```

sp_drop_item

Procedúra na vyhadzovanie veci z inventára postavy v boji. Taktiež pridá informácie do **combat_log**.

sp_wake_up_character

Procedúra na aktualizáciu zdravia hráča na základe nasledujúcich vzorcov.

Čas do úplného obnovenia zdravia vypočítame podľa vzorca:

$$WakeUpTime = \frac{max_{health} - current_{health}}{restoration\_rate} + current\_time$$

Kde $restoration\_rate$ je hodnota, ktorá určuje, ako často sa zdravie aktualizuje(napr. $\frac{1}{min}$).

Vypočítame novú hodnotu zdravia pre hráča podľa vzorca:

$$new\_health = min\,(max\_health, current\_health + restoration\_rate \cdot (current\_time - last\_sleep\_time))$$

- restoration_rate: rýchlosť obnovy (napr. 1 jednotka zdravia za minútu).
- current_time: Aktuálny čas.
- last_sleep_time: Čas, kedy bol stav odpočinku naposledy aktualizovaný.

Taktiež aktualizuje AP postavy.

Views

Príklady všetkých views sa zobrazia po ich spustení, po vykonaní testu zo súboru test_log.sql.

v_combat_state

Zobrazuje aktuálne kolo, zoznam aktívnych postáv a ich zostávajúce AP.

	character_game_stats_id	action_points
1	1	1

V danej hre sa zúčastnili postavy s ID 1, 2 a 4. V danom momente zostala iba jedna živá postava, ktorá má ešte určitý čas do skončenia.

	id	class_id	strength	dexterity	constitution	intelligence	health	action_points
1	3	3	7	12	6	8	70	60
2	1	1	10	5	10	5	29	1
3	4	4	8	6	12	8	0	17
4	2	2	4	8	5	12	0	41

v_most_damage

Zoraďuje postavy podľa celkového spôsobeného poškodenia vo všetkých bojoch.

	character_id	total_damage
1	2	84
2	4	80
3	1	70

v_strongest_characters

Zoznam postáv zoradených podľa súhrnných ukazovateľov výkonu (napr. spôsobené poškodenie, zostávajúce zdravie).

	character_id	damage	remaining_health
1	2	84	0
2	4	80	0
3	1	70	12

v_combat_damage

Sumarizuje celkové poškodenie spôsobené v každej bojovej relácii.

	combat_id	total_damage
1	1	234

v_spell_statistics

Štatistika používania kúziel a spôsobeného poškodenia.

	spell_id	total_damage	total_casts	average_damage
1	2	75	3	25
2	4	60	4	15

v_player_statistics

Štatistika hráčov: počet hier a počet víťazstiev.

	character_id	total_combats	combats_won
1	4	1	0
2	2	1	0
3	1	1	1

v_most_spells_unlocked

Ukazuje počet otvorených kúziel pre rôzne postavy.

	spell_id	name	rarity	total_unlocked
1	4	Dark Wave	uncommon	2
2	1	Fireball	common	2
3	3	Lightning Bolt	epic	1
4	2	Ice Spike	rare	1

v_most_items_spawned_in_combat

Ukazuje počet rôznych predmetov, ktoré boli vygenerované na ihrisku.

	item_id	name	rarity	total_generated
1	1	Iron Sword	common	3
2	3	Leather Armor	uncommon	1
3	4	Magic Ring	epic	1

v_combat_log_statistics

Štatistika postáv, ktorá bola zaznamenaná v aktuálnom boji pomocou combat_log.

	character_id	total_actions	total_attacks	total_casts	total_kills	total_loots	total_drops
1	1	15	11	1	2	1	1
2	2	11	5	3	0	1	1
3	4	9	4	4	0	0	0

Indexy

Pri vytváraní indexov bolo zohľadnené, ako často sa pristupuje k jednotlivým stĺpcom a aké operácie sú v systéme najčastejšie.

Boli vytvorené indexy pre hľadania classa pomocou class_id:

- `CREATE INDEX idx_characters_class_id ON characters (class_id);`

Ten index bode použitý pri hľadania classa pomocou class_id, čo spôsobí zrýchleniu hľadania zoznamov. Pre ukladanie štruktúry pre indexovanie budeme potrebovať použiť pamäť navyše. Taktiež operácie nad zoznamami môžu byt pomalšie. Preto musíme zvážovať, ktoré indexy budú použiteľne.

Nasledovné indexy boli skontrolované pomocou EXPLAIN ANALYZE alebo EXPLAIN na optimalizáciu výkonu dotazov v databáze. Táto analýza umožňuje identifikovať efektivitu indexov.

Indexy na overenie kúziel pre postavu, hľadanie kúziel postavy:

- CREATE INDEX IDX_CHARACTER_SPELLS_CHARACTER_ID ON CHARACTER_SPELLS (CHARACTER_ID);
- CREATE INDEX IDX_CHARACTER_SPELLS_SPELL_ID ON CHARACTER_SPELLS (SPELL_ID);

Hľadanie equipment postavy:

- CREATE INDEX IDX_CHARACTER_EQUIPMENTS_CHARACTER_ID ON CHARACTER_EQUIPMENTS (CHARACTER_ID);

Hľadanie modifikátorov pre equipments:

- CREATE INDEX IDX_EQUIPMENT_MODIFIERS_EQUIPMENT_ID ON EQUIPMENT_MODIFIERS (EQUIPMENT_ID);
- CREATE INDEX IDX_EQUIPMENT_MODIFIERS_MODIFIER_TYPE ON EQUIPMENT_MODIFIERS (MODIFIER_TYPE);

Hľadanie modifikátorov pre postavu:

- CREATE INDEX IDX_CLASS_MODIFIERS_CLASS_ID ON CLASS_MODIFIERS (CLASS_ID);
- CREATE INDEX IDX_CLASS_MODIFIERS_MODIFIER_TYPE ON CLASS_MODIFIERS (MODIFIER_TYPE);

Spell configuration:

- CREATE INDEX IDX_SPELL_CONFIGURATIONS_SPELL_ID ON SPELL_CONFIGURATIONS (SPELL_ID);

- CREATE INDEX IDX_SPELL_CONFIGURATIONS_EFFECT_TYPE ON SPELL_CONFIGURATIONS (EFFECT_TYPE);

Hľadanie postav v boje:

- CREATE INDEX IDX_COMBAT_PLAYERS_COMBAT_ID ON COMBAT_PLAYERS (COMBAT_ID);
- CREATE INDEX IDX_COMBAT_PLAYERS_CHARACTER_ID ON COMBAT_PLAYERS (CHARACTER_ID);

Predmety na bojisku, hľadanie predmetu pomocou ID a podľa ID hráča:

- CREATE INDEX IDX_GAME_ITEMS_ITEM_ID ON GAME_ITEMS (ITEM_ID);
- CREATE INDEX IDX_GAME_ITEMS_CHARACTER_ID ON GAME_ITEMS (CHARACTER_ID);

Pre inventár konkrétneho hráča a výpočet váhy inventára:

- CREATE INDEX IDX_INVENTORIES_CHARACTER_ID ON INVENTORIES (CHARACTER_ID);

Pre vyber predmetov podľa ich rarity:

- CREATE INDEX IDX_ITEMS_RARITY ON ITEMS (RARITY);

Pre hľadanie zoznamu oddychu postáv:

- CREATE INDEX IDX_SLEEP_CHARACTERS_CHARACTER_ID ON SLEEP_CHARACTERS (CHARACTER_ID);

Pre prácu zo zoznamami logov(hľadanie a usporiadanie):

- CREATE INDEX IDX_COMBAT_LOGS_COMBAT_ID ON COMBAT_LOGS (COMBAT_ID);
- CREATE INDEX IDX_COMBAT_LOGS_CHARACTER_ID ON COMBAT_LOGS (CHARACTER_ID);
- CREATE INDEX IDX_COMBAT_LOGS_DATE ON COMBAT_LOGS (DATE);

Akceptačne testy

Boli vytvorené akceptačné testy pre každú funkciu alebo procedúru. Tieto testy vychádzajú z jedného navrhnutého dátového modelu. Pre spustenie jednotných testov potrebujete mäť vytvorené tabuľky zo súboru $\boldsymbol{init.sql}$. Taktiež potrebujete vytvoriť základne zoznamy pre tabuľky spustením sobora $\boldsymbol{test\_data.sql}$, ktorý definuje procedúry pre testy. A po ktorom je možne spustiť jednotne testy.

Testy pre väčšinu procedúr boli vytvorené v priečinku **functions_tests**, ktorý zahŕňa nasledujúce testy(všetky testy používajú **test_data.sql**:

- f_effective_spell_cost.sql
  - V tomto test testujeme správne fungovanie a vypočítanie ceny kúzla.
  - Pri nesprávnej požiadavke, ako napríklad neexistujúce kúzlo, systém vyhodí chybu.
- sp_cast_spall.sql
  - V tomto teste je zaznamenané úplné testovanie zoslania kúziel prvého hráča a druhého hráča spolu s popisom a výpočtami v zdrojovom kóde.
  - Vráťte chybu, ak postava neexistuje, kúzlo neexistuje alebo nie je k dispozícii pre danú postavu. Test zároveň kontroluje počet AP pre danú postavu a hráča, na ktorého je kúzlo zoslané.
- sp_enter_combat.sql
  - Testuje správne priradenie hráčov do boja a logovanie v combat_logs.
- sp_loot_item.sql
  - Testuje správne pridelenie predmetu do inventára hráča, odstránenie z bojovej plochy a zaznamenanie zoznamu v **combat_logs.**
  - Vráti chybu, ak daný predmet neexistuje na bojovej ploche alebo nie je k dispozícii. Taktiež kontroluje limit inventára pre postavu a spracuje chyby zrušením transakcie a vyhodením výnimky.
- sp_reset_round.sql
  - Iba aktualizuje príslušný round v tabuľke **kombats** a zaznamená udalosť do tabuľky **combats_log**. Testuje existenciu očakávaných údajov po vykonaní danej procedúry.
- sp_rest_character.sql
  - Otestuje správnu aktualizáciu zdravia a AP postavy.

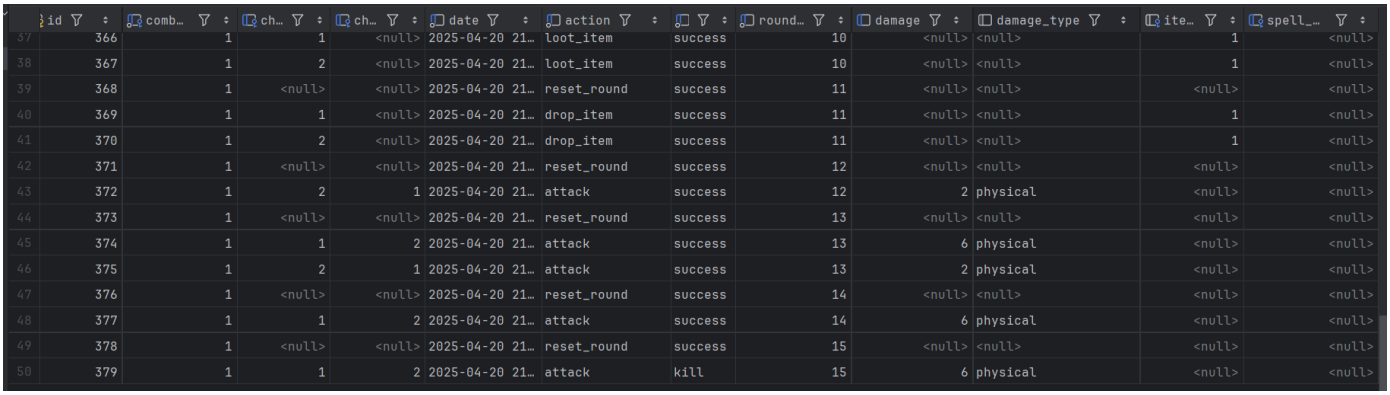
Tie testy vyhodia výnimky ak nastane chyba počas funkcii alebo pri overenie výsledku.

Pre odstránenie tabuliek alebo zoznamov z nich je potrebne spustiť skripty z $reset\_tabels.sql$.

Opis týchto testovacích skriptov je poskytovaný v samotnom kóde príslušných testov.

Ďalší test je $test\_log.sql$, v ktorom funkcia simuluje **combat** medzi 3 postavami a všetky akcie je zapísane do **combat_log.** Testovací scenár je viditeľný cez komentáre zanechané v kóde. V tomto teste sa testuje boj s loot_item a drop_item. Po skončení boja je možné pozrieť sa do **combat_log** a iných tabuliek. Taktiež po skončení tohto testu bola overená funkčnosť views a zaznamenaná vyššie.

Príklad combat_logs po skončení boja:



	id	comb…	ch…	ch…	date	action		round…	damage	damage_type	ite…	spell_…
37	366	1	1	<null>	2025-04-20 21…	loot_item	success	10	<null>	<null>	1	<null>
38	367	1	2	<null>	2025-04-20 21…	loot_item	success	10	<null>	<null>	1	<null>
39	368	1	<null>	<null>	2025-04-20 21…	reset_round	success	11	<null>	<null>	<null>	<null>
40	369	1	1	<null>	2025-04-20 21…	drop_item	success	11	<null>	<null>	1	<null>
41	370	1	2	<null>	2025-04-20 21…	drop_item	success	11	<null>	<null>	1	<null>
42	371	1	<null>	<null>	2025-04-20 21…	reset_round	success	12	<null>	<null>	<null>	<null>
43	372	1	2	1	2025-04-20 21…	attack	success	12	2	physical	<null>	<null>
44	373	1	<null>	<null>	2025-04-20 21…	reset_round	success	13	<null>	<null>	<null>	<null>
45	374	1	1	2	2025-04-20 21…	attack	success	13	6	physical	<null>	<null>
46	375	1	2	1	2025-04-20 21…	attack	success	13	2	physical	<null>	<null>
47	376	1	<null>	<null>	2025-04-20 21…	reset_round	success	14	<null>	<null>	<null>	<null>
48	377	1	1	2	2025-04-20 21…	attack	success	14	6	physical	<null>	<null>
49	378	1	<null>	<null>	2025-04-20 21…	reset_round	success	15	<null>	<null>	<null>	<null>
50	379	1	1	2	2025-04-20 21…	attack	kill	15	6	physical	<null>	<null>

Keď hráč zahynie, je potrebné odstrániť všetky jeho predmety z inventára a vybavenia.

References: