Database design for the core of a tactical RPG system

Andrii Dokaniev (AIS id: 128365)

Contents

Úvod

Modelovanie a vytváranie schémy databázového systému je neoddeliteľnou prípravnou časťou pred vytvorením akéhokoľvek produktu, ktorý potrebuje spracovanie údajov. Správne navrhnutý model údajov umožňuje tímu vývojárov nielen pochopiť hlavné vzťahy medzi kľúčovými prvkami systému, ale aj poskytuje možnosť v budúcnosti vykonať zmeny v štruktúre vytvoreného projektu bez potreby úplne prepracovať logiku projektu. Čo zároveň umožňuje vlastníkovi produktu minimalizovať náklady na následné zmeny poskytovaného produktu alebo služby.

Cieľom tohto zadania je vytvoriť systém pre ťahový RPG súbojový systém, ktorý s podrobnou dokumentáciou systému umožní komukoľvek, kto si ju prečíta, pochopiť, ako je usporiadaná logika herného sveta, jeho vzťahy medzi objektmi a štruktúra údajov v tomto hernom svete.

Hlavnou súčasťou tohto sveta sú dobrodruhovia, ktorí sa stretávajú vo vzťahových súbojoch, vrhajú kúzla, zbierajú magické predmety a bojujú o prežitie. Existuje veľa rôznych typov tried dobrodruhov, a používateľ si môže zvoliť ten typ herného aktéra, ktorý mu najviac vyhovuje. Tento výber bude čiastočne ovplyvňovať jeho osud a frekvenciu používania jeho kúziel. Počas boja postava môže zbierať veci z bojiska do svojho batohu. Počet vecí a dostupné miesto v batohu sú však obmedzené, a preto hráč bude rozmýšľať, aká vec je pre neho viac užitočná.

Rôznorodé množstvo typov kúziel dáva hráčovi viac priestoru na premýšľanie, pričom obmedzený počet ich použití vytvára na bojisku najväčšie napätie. Avšak pre rôzne postavy nebude vždy vhodné použiť ten istý kúzelník, a preto si hráč bude musieť sám dôkladne premyslieť, akú mágiu bude najlepšie použiť.

Postavy systému budú bojovať o prežitie, kým všetky kolá súboja neskončia(časovo ohraničene podlá počtu hráčov), alebo kým nebude vyhlásený víťaz bitky.

Na preklad textu bol použitý prekladač DeepL a vstavanú funkciu Word na kontrolu a opravu textu.

Funkčná špecifikácia

Databázové systémy sa používajú na ukladanie dlhodobých dát, ktoré tam môžu zostať uložené na ochranu, zatiaľ čo server počas tohto obdobia môže pokojne pracovať so zvyšnými požiadavkami a dopytmi zo strany klienta. Preto je mimoriadne dôležité určiť, ktoré z dát potrebujeme mať v databázovom systéme a ktoré nie. Od tohto rozhodnutia závisí nielen výkon aplikácie, ale aj celková efektívnosť servera.

Pre databázový systém je potrebné navrhnúť štruktúru, ktorá minimalizuje počet vstupných dopytov a umožní serveru efektívne pracovať iba s dátami, ktoré sú skutočne nevyhnutné.

Na začiatku potrebujeme vytvoriť základnú tabuľku pre používateľa v systéme – **Users**, ktorá bude obsahovať základné informácie o používateľovi, údaje pre prihlásenie a ďalšie informácie potrebné na spravovanie používateľa.

Tabuľka **UsersCharacters** bola vytvorená tak, aby hráč mohol mať viacero postáv. Každá postava je úplne nezávislá, akoby hráč začínal novú hru. Zároveň má možnosť kedykoľvek prepínať medzi postavami podľa svojho výberu.

Pre vytvorený herný svet bolo rozhodnuté vytvoriť **Item Storage**, v ktorom hráč bude mať k dispozícii svoje predmety. Hráč môže ukladať predmety do tohto úložiska po skončení boja a brať si odtiaľ svoje veci do svojho **Inventory** a do sovieho **CharacterEquipment** do ďalšieho boja. Ak hráč zahynie v boji, stratí všetky veci, ktoré si vložil do inventára. To, čo hráč uložil predtým do **Item Storage**, tam však zostane. Predmety z tabuľky **CharacterEquipment** budú ovplyvňovať charakteristiky postavy.

Pri uložení do **Item Storage** musíme manipulovať s kapacitou v inventári (**inventory_limit**). Keď hráč zoberie nejaký predmet, musíme zvýšiť kapacitu, pridať predmet do inventára a vymazať záznam z **Item Storage**. Ak chce hráč uložiť predmet, vymažeme zoznam predmetu z inventára, odčítame hodnotu **inventory_limit** a vložíme zoznam predmetu do **Item Storage**.

Pre veci, ktoré sa nachádzajú na poli bojiska, bola vytvorená nová tabuľka. Aby sledovať aké veci boli vygenerovane na poli bojiska tabuľka bola pridaná do databázy. **Game Items**, ktorá zaznamenáva, ktoré veci boli vygenerované pre príslušný boj. Keď niektorý hráč zahynie na poli bojiska, všetky jeho veci, ktoré mal pri sebe, budú pridané do tohto zoznamu(veci zo **Inventory** a **CharacterEquipment** zmiznú). Ak hráč chce z bojiska zobrať nejaký predmet, musíme overiť, či má dostatok voľnej kapacity v inventári

(**inventory_limit**). Limit inventára sa aktualizuje pri každom vložení predmetu a kontroluje, či nepresahuje povolený limit(ak presiahne, tak postava nemôže zobrať predmet). Ak áno, odstránime príslušný predmet zo zoznamu **Game Items** a pridáme ho do hráčovho inventára.

Na každom kole hráč bude mat možnosť spraviť iba jeden ťah. Aby hra mala férový charakter, bolo rozhodnuté, že hráč počas boja bude mať na svojom ťahu výber z dvoch možností:

- Zobrať nejaký predmet,
- Vybrať súpera a zaútočiť naň kúzlom alebo zbraňou.

Preto hráč bude premýšľať, či vezme predmet, alebo zaútočí na súpera.

Po skončení boja musíme obnoviť údaje o hráčoch (počet zdravia hráča). Túto úlohu prevezme server, aby sme nemuseli pre každého hráča ukladať hodnotu zdravia a vyhľadávať ju pri každej udalosti.

Keď hráč skončí hru, zahynie alebo stratí časť zdravia, musíme vedieť, kedy jeho zdravie obnoví na maximum. Na to slúži tabuľka **Sleep Characters**, v ktorej sú zaznamenané údaje s časom (timestamp), keď hráč obnoví svoje zdravie na maximum, a maximálna hodnota jeho zdravia. Preto, keď chce hráč ísť do boja, musíme skontrolovať, či jeho zdravie nie je 0. Od toho závisí, či mu dovolíme hrať, alebo nie.

Keď je hráč v systéme alebo keď chce ísť bojovať, musíme tiež skontrolovať, koľko času zostáva do úplného obnovenia zdravia. Na základe aktuálneho zdravia (**health**), maximálneho zdravia (**max_health**) a času do úplného obnovenia následne aktualizujeme hráčovo zdravie.

Čas do úplného obnovenia zdravia vypočítame podľa vzorca:

$$time\_to\_full\_health = \frac{max_{health} - current_{health}}{restoration\_rate} + current\_time$$

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

Vypočítame novú hodnotu zdravia pre hráča podľa vzorca:

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

Keď hra prejde na boj, musíme odstrániť čas na obnovu zdravia zo spiacich postáv (**Sleep Characters),** aby sme predišli kolíziám.

Keď hráč začne boj, systém bude čakať na pripojenie minimálneho počtu hráčov(3) potrebných na boj. Po dosiahnutí minimálneho počtu sa spustí časovač, ktorý odpočítava čas do začiatku kola, pričom ďalší hráči sa môžu stále pripojiť. Ak sa niektorý hráč odhlási a počet hráčov klesne pod minimum, časovač sa zastaví a systém bude vyžadovať pripojenie aspoň jedného ďalšieho hráča. Počas boja bude mať každý hráč na ťah 15 sekúnd. Ak hráč neurobí ťah, systém automaticky pokračuje k ďalšiemu hráčovi. Ak hráč vykoná ťah, zvyšné sekundy sa nečakajú a hra prejde na nasledujúceho hráča. Aby boj netrval nekonečne dlho alebo hráči zámerne neodkladali svoje ťahy, herný čas bude obmedzený podľa vzorca $n \times 3$, kde *n* je počet hráčov a 3 je počet minút na hráča.

Keď sa boj začne, systém vytvorí tabuľku **Game Items**, v ktorej sa budú nachádzať predmety dostupné na bojovom poli. Predmety budú generované náhodne s ohľadom na počet hráčov(**n**) a kvalitu predmetov:

- **Common** – Lower Bound (n * m) predmetov na hru, kde m je náhodne číslo od 50% do 150%.
- **Uncommon** – Lower Bound($n/2$), 50% na každý predmet.
- **Rare** – $\min(\frac{n*30\%}{100\%}, 2)$ predmetov na hru.
- **Epic** – 3% na predmet na hru.
- **Legendary** – 0.5% na predmet na hru.

Ďalej sa vytvorí tabuľka so zoznamom hráčov v boji **Combat_players**, ktorá v budúcnosti môže obsahovať informácie, ako napríklad počet zabití hráča, získané skúsenosti (**XP**) a ďalšie štatistické údaje o hráčovi.

V budúcnosti, ak bude potrebné pridať kúzlo na obnovenie zdravia, musíme niekde uchovávať maximálne zdravie hráča. Toto môžeme zabezpečiť buď na strane servera, alebo v databáze. Táto hodnota sa bude nachádzať v tabuľke **Sleep Characters**.

Dočasné hodnoty, ako napríklad **AP**, je vhodné uchovávať na strane servera, ktorý zabezpečí náhodné poradie hráčov v každom kole a obnoví **AP** na maximum po každom kole. Rovnako bude kontrolovať, aby počet AP neklesol pod 0 pri zosielaní kúzla. Ak sa tak stane, postava nebude môcť kúzlo zoslať.

Tabuľka **Characters_game_stats** obsahuje informácie o profile hráča, ako napríklad počet hier, počet víťazstiev a získané XP. Na realizáciu týchto štatistík však budú potrebné dodatočné manipulácie s databázou, čo v tomto článku uvádzať nebudeme. Počet hier a počet víťazstv by bolo najjednoduchšie spraviť, len že pre pocte XP. musíme navrhnúť ďalšiu logiku. Príkladom slúži pridanie nejakého poctu XP. podlá urovná postavy.

Keď hráč premiestni nejaký predmet zo svojho **Item Storages** do **Inventory**, musíme najprv skontrolovať **inventory_limit**. Okrem toho je potrebné prejsť tabuľku hráčovho inventára a overiť, či všetky typy predmetov sú v súlade s pravidlami a či hráč nemá kolíziu s existujúcimi predmetmi.

Keďže systém má rozširujúci charakter (môže sa rozšíriť v budúcnosti, pridávať typy predmetov, meniť popisy a základné charakteristiky), je vhodné rozdeliť organizáciu tabuľky o predmetoch na dve tabuľky. Máme základnú tabuľku **Items**, ktorá obsahuje informácie o predmete a jeho charakteristikách.

Každý **predmet** má svoju váhu a hráč musí premýšľať, čo si chce odniesť so sebou po skončení boja. Predmety, ktoré si hráč môže odniesť, budú rozdelené do jednotných tried (classov):

- **Zbrane**: meče, luky, dýky, sekery a magické palice...
- **Zbroje**: prilby, brnenia, štíty, plášte...
- **Magické predmety**: zvitky, magické knihy...

Z pohľadu herného sveta sú takéto tabuľky, ako **Items** a **Spells**, pomerne malé a neobsahujú veľa informácií ani záznamov, čo nám umožňuje zlepšiť organizáciu tabuľky s podporou ďalších modifikácií v budúcnosti. Zároveň to odstraňuje duplicitu, zmenšuje hlavnú tabuľku a pridanie nového typu bude ľahko realizovať. Okrem toho, **Items_type** môže poskytovať spoločné charakteristiky a opis predmetov. Takisto **Item** ukazuje, či ide o **Equipment** alebo nie. Pre **Equipment** bola vytvorená nová tabuľka, ktorá obsahuje predmety s rôznymi kategóriami. Pre lepšie manažovanie rôznych kategórií bola vytvorená tabuľka **Equipment_Categories**, ktorá určuje, do ktorej kategórie patrí daný equipment. **EquipmentModifiers** určuje, na ktoré atribúty pôsobí daný equipment.

V budúcnosti, ak budeme aktualizovať charakteristiky, nebudeme to musieť robiť cez inštancie **Items** – stačí zmeniť jej hlavný typ.

Podobnú situáciu máme aj s tabuľkou **Spell** a jej podtabuľkami. Tabuľka **Spell_Categories** definuje kategóriu kúziel, ktorá pridáva základné charakteristiky do všetkých kúziel, ktoré majú daný typ. Hráč môže používať kúzla, ktoré patria do rôznych kategórií. Tieto kúzla ovplyvňujú rôzne štatistiky hráča pri vykonaní útoku, ktoré je nastavene v tabuľke **SpellConfigurations**.

Postava bude mať na začiatku základný počet kúziel (**CharacterSpells**). Počet týchto kúziel sa bude časom zvyšovať.

Ako návrh na realizáciu je vhodné definovať základný počet rôznych kúziel a pridať možnosť získavať počas boja rôzne zvitky, ktoré hráčovi odomknú nové kúzlo alebo poskytnú určitú

kompenzáciu v prípade duplicity. Tieto kúzla hráč nemôže vidieť počas boja, kým zvitok neotvorí. Je však potrebné zaviesť aj úrovne týchto zvitkov, aby hráč vedel, či sa oplatí počas boja vziať zvitok so sebou alebo nie. Na to bude slúžiť atribút quality v predmete.

Na to v vytvorenom modeli slúži tabuľka **Character_spells**, ktorá umožňuje rozdeliť kúzla pre hráčov a zároveň poskytuje podporu ďalšieho rozvoja v budúcnosti.

Tabuľka **Classes** má v podstate podobný význam ako ostatné vyššie uvedené. Definuje konkrétne postavy v tom hernom svete. **ClassModifiers** určuje, ktoré charakteristiky ovplyvňuje základný typ postavy.

Characters

Každá **Character** patrí do určitej **Class**, hráč si musí sám zvoliť, za aký typ postavy chce hrať. Od toho závisia aj základné štatistiky postavy a typy **Spells**, ktoré pre jeho postavu viac vyhovujú. A Postava má definované základné atribúty podľa zvoleného **Class** rozlíšene, k týmto atribútom patria:

- Strength
- Dexterity
- Constitution
- Intelligence
- Health

Tieto postavové atribúty ovplyvňujú aj ďalšie, ako napríklad:

- Action Points(AP)
- Armor Class(CP)

AP (Action Points) určujú, koľko rôznych kúziel môže postava použiť počas boja. Je to energia potrebná na kúzlenie počas boja. Rôzne kúzla majú rôznu cenu použitia, ktorá závisí od viacerých faktorov, ako napríklad typ postavy a jej charakteristiky.

Maximálny počet **Action Points** je definovaný nasledovným vzorcom:

$$MaxActionPoints = (Dexterity + Intelligence)ClassModifier \quad (1)$$

Keďže rôzne typy postáv majú odlišné charakteristiky, pre niektorých je lacnejšie použiť určité kúzlo. Cena kúziel je definovaná nasledujúcim vzorcom:

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

Kde

- $BaseCostCategory.Modifier$ – zakladna cena kuzla,

- $SelectedAttribute$ – atribbut postavy(Intelligence, ...)
- $ItemModifiers$ – modifikator, ktorý dáva predmet.

Kde **ClassModifier** je unikátna hodnota, ktorá patrí rôznym Classom. Táto hodnota poskytuje bonus k Maximálnemu počtu **AP**.

Na základe **CP** (Armor Class) system určujú, ci kúzlo trafilo do cieľa alebo nie. Robí sa to tak že,

- Systém hodí kockou d20 (vygeneruje náhodné číslo od 1 do 20)
- Pridá sa bonus na základe atribútov útočníka (napr. **Intelligence** pre kúzlo alebo **Strength** pre útok).
- Výsledok sa porovná s **AC** cieľa.
- Ak dôjde k zásahu, vypočíta sa **Damage** a aplikuje sa na cieľovú postavu. Ak hodnota presiahne **AC** cieľa, útok je úspešný a spôsobuje poškodenie.

Ako vypočítame AC? Každá postava počíta AC na základe svojej Dexterity a bonusu z triedy podľa vzorca:

$$ArmorClass = 10 + (Dexterity/2) + ClassArmorBonus \quad (3)$$

Posledným atribútom hráča je **Inventory Limit**, ktorý určuje, koľko predmetov môže postava umiestniť do svojho inventára počas boja. Ak hráč chce pridať predmet do inventára, treba najskôr overiť, či je tam voľné miesto. Maximálna hodnota váhy predmetov v inventári je určená na základe **Strength** a **Constitution** postavy:

$$MaxInventoryWeight = (Strength + Constitution)ClassInventoryModifier \quad (5)$$

Combat

Keď sa začne boj (combat) alebo keď postava vstúpi do combatu, všetky zúčastnené postavy sú uzamknuté do ťahovej štruktúry, kde je poradie vygenerované v ľubovoľnom poradí. Každá postava v boji môže používať kúzla, pokiaľ má k dispozícii AP, a zbierať predmety z bojiska. Ak postava zahynie, opustí boj a všetky jej predmety budú presunuté na bojové pole.

Počas boja **Armor Class** (AC) sa vypočíta podľa vyššie uvedeného vzorca, pričom systém použije náhodné číslo (vygenerované od 1 do 20). Na základe tohto čísla určí, či postava zasiahla cieľ. Čím je postava skúsenejšia alebo silnejšia v príslušnom atribúte, tým väčší

účinok má jej kúzlo alebo úder. Počet naneseného poškodenia (damage) sa vypočíta podľa nasledujúceho vzorca:

$$Damage = BaseDamage(1 + ConfiguredAttribute/20) \ (4)$$

Kde **BaseDamage** je určené hodnotou s tabuľke **Spells** a **ConfiguredAttribute** predstavuje atribút spojený s konkrétnym kúzlom, ktorý ovplyvňuje jeho účinnosť.

Combat Logs

Každá akcia vykonaná počas kola sa zaznamenáva do bojového denníka. Kto konal? Akú akciu vykonal? Aké je číslo kola? Aká hra? Zaznamenávajú sa aj výsledky(**EventMsgType**), ako napríklad:

- Hit
- Miss
- DamageDealt
- ItemPicked

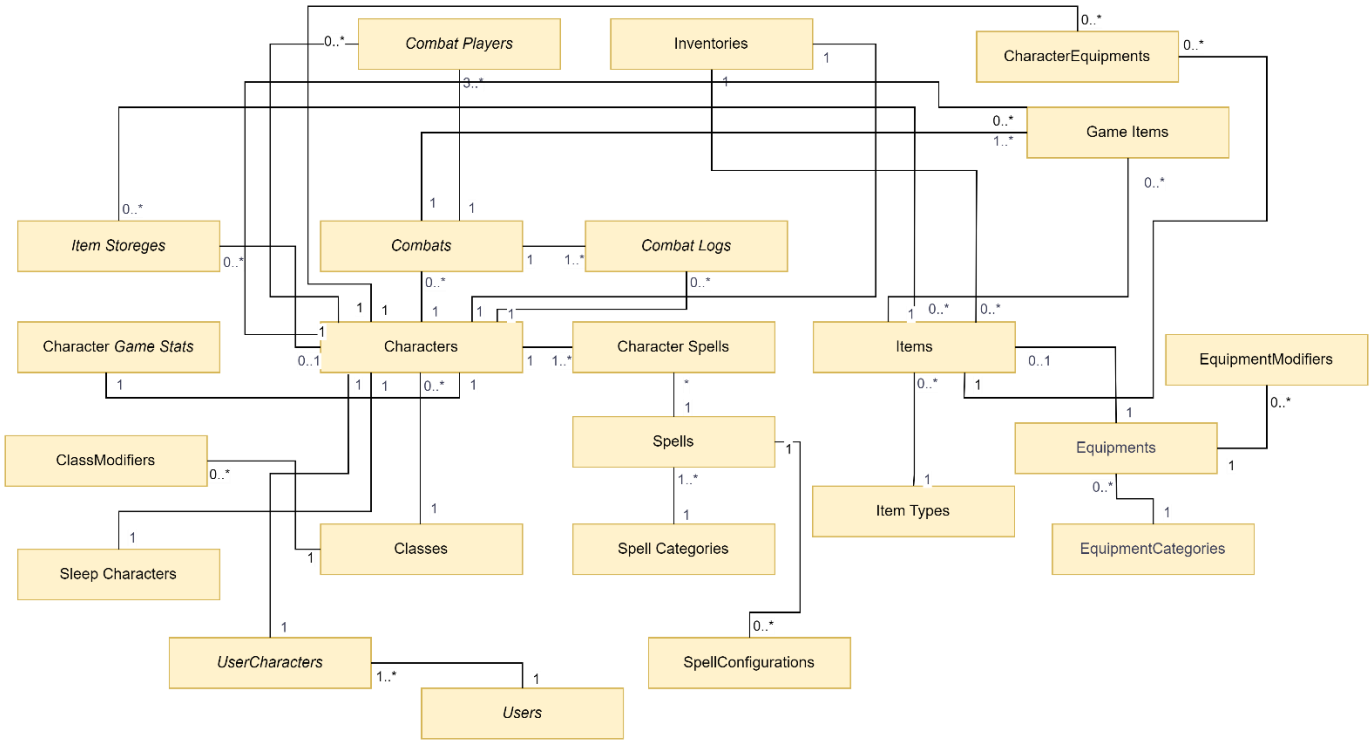
K akciám patria:

- Attack
- ItemPickup

DamageType vyzerá nasledovne:

- Physical
- Magical

Logicky E-R model



1. Users — UserCharacters
   - ❖ Type of relationship:
     - One to many (1:1..*).
   - ❖ Description:
     - Jeden používateľ môže vytvoriť viac **svojich** postav pre sebe(postavy sú nezávislé od seba. Používateľ si môže vybrať, za ktoré postavy chce hrať, podľa toho, ktoré sa mu viac páčia)
     - Každá vytvorená používateľom postava patriť jednému používateľu, ktorý ju vytvoril.
2. UserCharacters — Characters
   - ❖ Type of relationship:
     - One to one (1:1).
   - ❖ Description:
     - Jedená používateľská postava patri do jednej postavy v systéme.
     - **Pre každú** vytvorenú používateľom postavu systém vytvorí nezávislý blok.
3. Characters — Classes
   - ❖ Type of relationship:
     - Many to one (0..*:1).

- ❖ Description:
  - Viacero postav v systéme môže mat ten istý class.
  - **Pre každú** postavu v systéme musí byt zadefinovaný **iba jeden** class, do ktorého patri postava.

4. Classes — ClassModifiers
   - ❖ Type of relationship:
     - One to many (1:0..*).
   - ❖ Description:
     - **Pre každý** class postavy **môže** existovať viacero rôznych modifikátorov pre rozlične parametre classa.
     - **Rôzne** typy modifikátorov so špecifickými hodnotami **definujú** doplnkové charakteristiky classa.
5. Characters — Sleep Characters
   - ❖ Type of relationship:
     - One to one (1:1).
   - ❖ Description:
     - **Pre každú** postavu **existuje** pravé jeden zoznam informácií obsahujúci koľko bude postava oddychovať.
6. Characters — Character *Game Stats*
   - ❖ Type of relationship:
     - One to one (1:1).
   - ❖ Description:
     - **Pre každú** postavu **existuje** pravé jeden zoznam štatistik danej postavy.
     - **Každý** zoznam štatistik patri určitej postave.

M:N medzi Characters *a Spells,* Character Spells je medzitabuľka, ktorá definuje pre každého používateľa rôzne dostupne kúzla. Obsahuje kľúčové atribúty CharacterID a SpellID. CharacterID odkazuje na tabuľku Characters a SpellID na tabuľku Spells.

7. Characters — Character Spells
   - ❖ Type of relationship:
     - One to many (1:1..*).
   - ❖ Description:
     - Každá postava môže **mať** rôzny počet kúziel(minimálne jedno základne), ktoré sú dostupne pre danú postavu.
8. Character Spells — Spells
   - ❖ Type of relationship:
     - Many to one (0..*:1).

- ❖ Description:
  - **Každý** zoznam postavy je spojený s určitým kúzlom, ktoré je k dispozície danej postave.

9. Spells — Spell Categories
   - ❖ Type of relationship:
     - Many to one (1..*:1).
   - ❖ Description:
     - Spell Categories **definuje** kategórii pre príslušný spell.
     - Každé kúzlo patri do jednej z dostupných kategórii kúziel.
10. Spells — SpellConfigurations
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Každé kúzlo môže mať svoje konfigurovateľne atribúty.
      - Rôzne unikátne atribúty patria pre jedno kúzlo.

M:N medzi Character a Items. Item Storages je medzitabuľka, ktorá umožňuje pre každého používateľa mat rôzny počet veci na chránenie. Ma kľúčové atribúty ItemID a CharacterID. CharacterID odkazuje na tabuľku Characters ItemID na tabuľku Items.

11. Characters — Item Storages
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - **Pre kazdu** postavu existuje storage v ktorom postava môže mat rôzny počet predmetov.
12. Item Storages — Items
    - ❖ Type of relationship:
      - Many to one (0..*:1).
    - ❖ Description:
      - Item Storages obsahuje rôzne veci pre rôzne postavy.
13. Items — Item Types
    - ❖ Type of relationship:
      - Many to one (1..*:1).
    - ❖ Description:
      - Existuje veľa predmetov, ktoré patria iba do jednej kategórii.
14. Items — Equipments
    - ❖ Type of relationship:
      - One or zero to one (0..1:1).

- ❖ Description:
  - Predmet môže patriť medzi vybavenie, ale nemusí.

15. Equipments — EquipmentCategories
    - ❖ Type of relationship:
      - Many to one (0..*:1).
    - ❖ Description:
      - Každé equipment patrí iba do jednej kategórie.
      - Do jednej kategórie môže patriť viac zariadení.
16. Equipments — EquipmentModifiers
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Jeden equipment môže mat viacero modifikátorov pre rôzne štatistiky.

M:N medzi Characters a Combats. Combat Players je medzitabuľka, ktorá umožňuje pre každú hru viest zoznam hráčov. Obsahuje kľúčové hodnoty CharacterID a CombatID. CharacterID odkazuje na tabuľku Characters a CombatID na tabuľku Combats.

17. Characters — Combat
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Každá persona ma viacero zoznamov hier v ktorých ona sa zúčastnila a zvíťazila(pole winnerID).
18. *Combats — Combat Players*
    - ❖ Type of relationship:
      - One to many (1:3..*).
    - ❖ Description:
      - Pre každú hru mame minimálne 3 hráča, ktoré môžu sa zúčastniť v hre.
19. *Characters — Combat Players*
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Pre každú hru, každí hráč môže obťahovať zoznam hráčov v *Combat Players*.
20. *Combats* — Game Items
    - ❖ Type of relationship:
      - One to many (1:1..*).

- ❖ Description:
  - **Každá** hra ma viacero predmetov na poli.

21. *Combats — Combat Logs*
    - ❖ Type of relationship:
      - One to many (1:1..*).
    - ❖ Description:
      - Každá hra ma svoj bojový log v ktorom zaznamenane rôzne udalosti hry.
22. Characters — Game Items
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Jedna postava môže priniesť alebo uniesť rôzne Idem z boja, je to zaznamenane v Game Items.

M:N medzi Characters a Items. CharacterEquipments a Inventory je medzitabuľka, ktorá umožňuje pre každú postavu mat so sebou rôzne veci. Ty tabuľky spájame pomocou CharacterID a ItemID. CharacterID odkazuje na tabuľku Characters a ItemID na tabuľku Items.

- ❖ Many to many (M:N)
  - Každá postava môže vlastniť od nuly do viacerých predmetov.
  - Každý predmet môže patriť viacerým postavám.

Realizovaná cez medzitabuľku **Inventories**.

23. Characters — CharacterEquipments
    - ❖ Type of relationship:
      - One to many (1:0..*).
    - ❖ Description:
      - Jedna postava môže mať v svojich equipments viacero rôznych typov zbrane a iných equipments.
24. Characters — Inventories
    - ❖ Type of relationship:
      - One to many (1:1).
    - ❖ Description:
      - Jedna postava ma iba jeden inventory, ktorej obsahuje viacero položiek Items tejto postavy.
25. Characters — *Combat Logs*
    - ❖ Type of relationship:

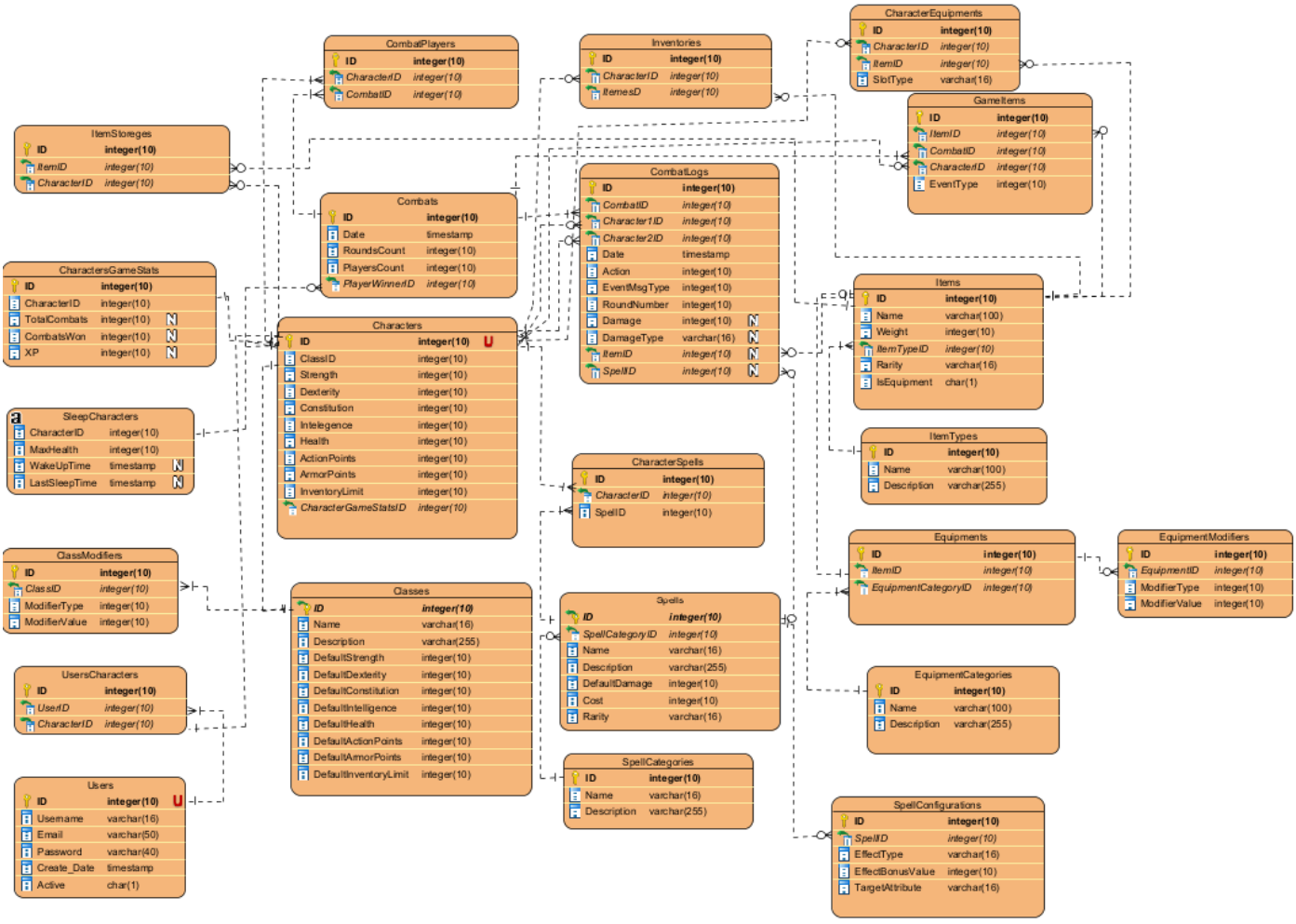
- One to many (1:0..*).
- ❖ Description:
  - Každý hráč môže byt zaznamenaný vo viacerých udalostiach v rôznych hrach.

26. Inventories — Items
    - ❖ Type of relationship:
      - Many to one (0..*:1).
    - ❖ Description:
      - Inventories môže obsahovať viacero rôznych Items rôznych hráčov.
27. CharacterEquipments — Items
    - ❖ Type of relationship:
      - One to many (0..*:1).
    - ❖ Description:
      - Postava môže mat viacero equipments, len tie equipments musia obsahovať rôzne Items, a Items rôznych typov.

M:N medzi Combats a Items. Game Items a Inventory je medzitabuľka, ktorá umožňuje pre každú hru mať rôzny počet predmetov, predmety môžu byt rovnaké. Kľúčové atribúty CombatID a ItemID. CombatID odkazuje na tabuľku *Combats* a ItemID na tabuľku *Items*.

28. Game Items— Items
    - ❖ Type of relationship:
      - One to many (0..*:1).
    - ❖ Description:
      - Postava môže mat viacero equipments, len tie equipments musia obsahovať rôzne Items, a Items rôznych typov.

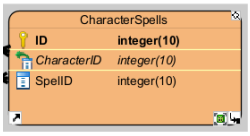
Relačná štruktúra fyzického modelu



Kľúčové procesy

V tej to časti znázorníme klubové procesy systému, ktoré boli opísane v hlavnej časí.

1. **Čo sa stane, keď postavy zosielajú kúzlo vnútri boja? Ktoré entity/záznamy sa vytvoria? Aké atribúty sa menia? Čo sa musí skontrolovať?**
   - Postava ma svoje otvorene **Spells**, ktoré je zaznamenane v tabuľke CharacterSpells. Postava môže zoslať kúzlo len vtedy, ak je kúzlo dostupné pre túto postavu. Musíme skontrolovať, či hráč ma tento kúzlo otvorene.



Character Spells 1

Budem poväzovať, že hráču stačí počet AP na zvolene kúzlo, v opačnom prípade nemôže zaslať dane kúzlo. Potrebný počet **Action Points**(AP) vypočítame podlá nasledujúceho vzorca:

$$EffectiveCost = BaseCostSpell.Modifier(1 - \frac{SelectedAttribute}{100})(1 - ItemModifiers)$$

Kde

- $\boldsymbol{BaseCostSpell}$ berieme z tabuľky **Spells,** ako základnú cenu kúzla.
- $\boldsymbol{SelectedAttribute}$ – Intelligence z tabuľky **Characters**.
- $\boldsymbol{ItemModifiers}$ – modifikator, ktorý dáva predmet na daný atribút hráča, s tabuľky **EquipmentModifiers**(napríklad Magická palica dáva bonus k Intelligence). $\boldsymbol{ItemModifiers}$ môže byť viacero, alebo vôbec nebyť. Keby ich bolo viacero, tak vzorec $(1 - ItemModifiers)$ vyzeral nasledovne: $max((1 - ItemModifiers\_1 + ItemModifiers_2 + \cdots + ItemModifiers\_n), 0.1)$.

Počas boja hráč je zaznamenaný v tabuľke **Combat_players.**

Pri každej akcie(po každom ťahu hráča) počas boja do **CombatLogs** sa pridá záznam obsahujúci:

- ID hry.
- ID hráča, ktorý zosiela kúzlo.
- ID hráča, na ktorého bolo kúzlo zoslané.
- Dátum tejto akcie.
- Typ akcie: **Attack**.
- **EventMsgType**, ktorý bude ukazovať výsledok akcie.
- Aktuálne číslo kola.

- **Damage**, ak hráč trafil cieľ.
- **DamageType** bude **Magical**.
- **ItemID** bude prázdne.
- **SpellID** bude odkazovať na zoslané kúzlo.

Na základe **CP** (Armor Class) system určujú, ci kúzlo trafilo do cieľa alebo nie. Robí sa to tak že,

- Systém hodí kockou d20 (vygeneruje náhodné číslo od 1 do 20)
- Pridá sa bonus na základe atribútov útočníka (***napr. Intelligence pre kúzlo*** alebo **Strength** pre útok).
- Výsledok sa porovná s **AC** cieľa.
- Ak dôjde k zásahu, vypočíta sa **Damage** a aplikuje sa na cieľovú postavu. Ak hodnota presiahne **AC** cieľa, útok je úspešný a spôsobuje poškodenie.

To by sme mohli počítať podlá vzorca:

$$Attack = d20 + AttackBonus$$

Kde **AttackBonus** predstavuje **Intelligence** pre kúzlo alebo **Strength** pre fyzický útok.

Na zásah hráča kúzlom musíme počítať podlá vzorca:

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

Každá postava počíta AC na základe svojej Dexterity a bonusu z triedy **ClassModifiers**.

Útok zasiahne, ak $Attack > ArmorClass$.

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

$$Damage = BaseDamage(1 + ConfiguredAttribute/20)$$

Kde **BaseDamage** je určené hodnotou z tabuľke **Spells**(DefaultDamage) a **ConfiguredAttribute** z tabuľky **SpellConfigurations** predstavuje atribút spojený s konkrétnym kúzlom, ktorý ovplyvňuje jeho účinnosť, atribút berieme z tabuľky **Characters**(napr. Intelligence).

Atribúty menia:

- U hráča, ktorý zoslal kúzlo, sa musí odpočítať **AP** (Action Points) z tabuľky **Characters** podlá hodnoty **EffectiveCost**.
- Ak hráč trafil cieľ, u porazeného hráča sa z tabuľky **Characters** zo stĺpca Health odpočíta hodnota spôsobeného **damage**.

Skontrolovať sa musí:

- Či hráč ma tento kúzlo otvorene.
- Či má hráč dostatok **AP** na zoslanie kúzla.
- Či hráč trafil cieľ.
- Absolútny počet zdravia hráča.
- Skontrolovať, či porazený hráč zahynul na základe jeho hodnoty **Health**, ak **Health ≤ 0**, hráč je považovaný za mŕtveho a všetky ho veci zo **Inventory** a **CharacterEquipment** musíme dať na hernú plochu.

Ak hráč zahynie, všetky predmety z jeho inventára (**Inventory** a **CharacterEquipment**) sa presunú do tabuľky **Game_Items** s uvedením aktuálneho **CombatID**. Do **CombatLogs** sa pridá záznam s EventMsgType = Death. A pre všetky veci, ktoré postava zobrala v **Game Items(**je zaznamenaný CharacterID) budú prepísane bez **CharacterID.**

2. Ako postava odpočíva a obnovuje zdravie?

Ak hráč zahynie alebo nemá plný počet zdravia, je potrebné obnoviť zdravie. Na tento účel slúži tabuľka **SleepCharacters**. V tabuľke **SleepCharacters** sú zaznamenané nasledujúce údaje:

- Maximálne zdravie hráča.
- Čas do úplného obnovenia zdravia.
- **CharacterID**, identifikátor postavy.
- Čas poslednej aktualizácie.

Keď je hráč v systéme alebo keď chce ísť bojovať, musíme tiež skontrolovať, koľko času zostáva do úplného obnovenia zdravia. Na základe aktuálneho zdravia (**health**), maximálneho zdravia (**max_health**) a času do úplného obnovenia následne aktualizujeme hráčovo zdravie.

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.

Keď hra prejde na boj, musíme odstrániť čas na obnovu zdravia zo spiacich postáv (**Sleep Characters),** aby sme predišli kolíziám. Hráč počas boja nemôže odpočívať.

Čo sa aktualizuje/zapisuje:

- V tabuľke **Characters**: Aktualizuje sa pole **Health**.
- V tabuľke **Sleep_Characters**: Aktualizuje sa **LastSleepTime**, alebo sa záznam odstráni.

3. Čo sa stane, keď vstúpi do boja?

Viac informácií, ktoré sa týkajú času boja, minimálneho počtu hráčov a ďalších aspektov nesúvisiacich s relačným modelom, je opísaných v hlavnej časti.

Vytvorí sa zoznam hry **Combats** obsahujúci:

- Data – dátum hry.
- RoundCount –počet kôl.
- PlayersCount –počet hráčov.
- PlayerWinnerID – id víťazného hráča.

Postavy sa pridajú do tabuľky **Combat_players** so záznamami:

- CombatID – aktuálny boj.
- CharacterID – postava.

Vygeneruju sa predmety na pole boja **GameItems**:

- ItemID – id predemta.
- CombatID – aktuálny boj.
- CharacterID – postava.
- EventType – udalosť (napr. hráč zobral ten predmet, alebo zahynul a s neho vypadol predmet)

Hráč prestane oddychovať, zmažeme WakeUpTime a LastSleepTime z tabuľky SleepCharacters.

Každý hráč bude mat k dispozícii iba tie spell, ktoré ma v tabuľke CharacterSpells.

4. Ako systém spracuje zoslanie kúzla (so všetkými výpočtami a šancami na zásah)?

Bolo opísané, ako v opise systému, tak aj v odpovedi na otázku č. 1.

Pre stručnosť:

- Kontroluje sa prítomnosť AP, aby hráč mohol zoslať dane kúzlo.

- Vypočíta sa šanca na zásah (d20 + bonus atribútu proti ArmorClass cieľa).
- Ak je zásah úspešný, vypočíta sa poškodenie a aktualizuje sa hodnota Health cieľa.
- Všetky akcie sa zaznamenávajú do CombatLogs.

Hráč má svoje kúzla uložené v tabuľke **CharacterSpells**. Ak zošle kúzlo, tak z tabuľky **Spells** zoberieme hodnoty **Cost** a **DefaultDamage**, ktoré sú potrebné na výpočty, uvedené už predtým. Po spojení s tabuľkou **SpellConfigurations** zistíme, od ktorých atribútov kúzlo závisí a v akom rozsahu.

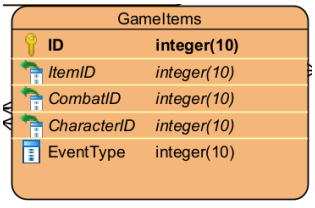
5. Ako sa pridávajú predmety do boja?

Keď sa boj začne, systém vytvorí tabuľku **Game Items**, v ktorej sa budú nachádzať predmety dostupné na bojovom poli. Predmety budú generované náhodne s ohľadom na počet hráčov v danej hre(CombatPlayers. CombatID = CombatID, označíme ako **n)** a kvalitu predmetov:

- **Common** – Lower Bound (n * m) predmetov na hru, kde m je náhodne číslo od 50% do 150%.
- **Uncommon** – Lower Bound($n/2$), 50% na každý predmet.
- **Rare** – $\min(\frac{n*30\%}{100\%}, 2)$ predmetov na hru.
- **Epic** – 3% na predmet na hru.
- **Legendary** – 0.5% na predmet na hru.

Táto pravdepodobnosť zabezpečí, že v každej hre bude minimálne jeden predmet na každého hráča, pričom najvzácnejšie predmety sa budú objavovať iba zriedka.

Predmety sa pridávajú do **GameItems** po zactaku boja a tiež v prípade, že niektorý hráč odíde, jeho veci sa presunú do GameItems.



- **ItemID**: Identifikátor predmetu pridaného do hry.
- **CombatID**: Identifikátor súboja.
- **CharacterID**: Identifikátor postavy, ktorá je spojená s predmetom(ak postava zoberie predmet, alebo ak opustí hru, predmet tejto postavy sa presunie na bojové pole).
- **EventType**: Typ udalosti spojený s predmetom alebo hracom a premetom(indikuje, či predmet pochádza od hráča, ktorý opustil hru, alebo či ho hráč zdvihol...).

6. Ako postava získa predmet a skontroluje limity inventára?

Postava môže získať predmet, ktorý je uložený v tabuľke **Game_Items** a je dostupný v aktuálnej hre, ak bude mať dostatok miesta v **Inventory**. **Game_Items** ma stĺpec **EventType,** ktorý určuje, ci **item** bol prinesený hracom(hráč odišiel z hry a zahynul) alebo ci daný predmet bol zobraný hracom. Tie políčka meníme počas udalostiach súvislých s tímy predmetmi v boji.

Na každom kole hráč bude mat možnosť spraviť iba jeden ťah. Aby hra mala férový charakter, bolo rozhodnuté, že hráč počas boja bude mať na svojom ťahu výber z dvoch možností:

- Zobrať nejaký predmet,
- Vybrať súpera a zaútočiť naň kúzlom alebo zbraňou.

Inventory Limit určuje koľko predmetov môže postava umiestniť do svojho inventára počas boja. Ak hráč chce pridať predmet do inventára, treba najskôr overiť, či je tam voľné miesto. Maximálna hodnota váhy predmetov v inventári je určená na základe **Strength** a **Constitution** postavy:

$$MaxInventoryWeight = (Strength + Constitution)ClassInventoryModifier$$

Kde:

- **Strength** a **Constitution** sa berú z tabuľky **Characters**.
- **ClassInventoryModifier** z tabuľky **ClassModifiers**.

Hmotnosť predmetu sa berie z tabuľky **Items** podľa daného **ItemID**. Hráč môže vziať predmet, ak počet predmetov pre jeho postavu v **Inventory** a **CharacterEquipment** spolu s pripočítanou hmotnosťou nového predmetu nepresiahne jeho **MaxInventoryWeight**. Musíme skontrolovať hmotnosť inventára s pripočítaným predmetom a **MaxInventoryWeight.**
Keď hráč zoberie predmet, tak v tabuľke **Game Items** pre tento predmet bude prepísan CharacterID postavy. Po skončení boja, všetky označene veci budú pridane do **Inventory** postavy.
Keď hráč zoberie predmet tak do **CombatLogs** sa pridá zoznam:

- Character1ID – id postavy.
- Action: "ItemPickup".
- EventMsgType: "ItemPicked".
- ItemID: id predmeta.

Záver

Navrhnutý dátový systém poskytuje štruktúrovanú štruktúru údajov pre RPG hru, opisuje mechaniky a pravidlá, ktorými sa hra riadi. Relačná štruktúra logického modelu znázorňuje koncepcie a poskytuje flexibilitu so zameraním na rozšírenie v budúcnosti.

References: