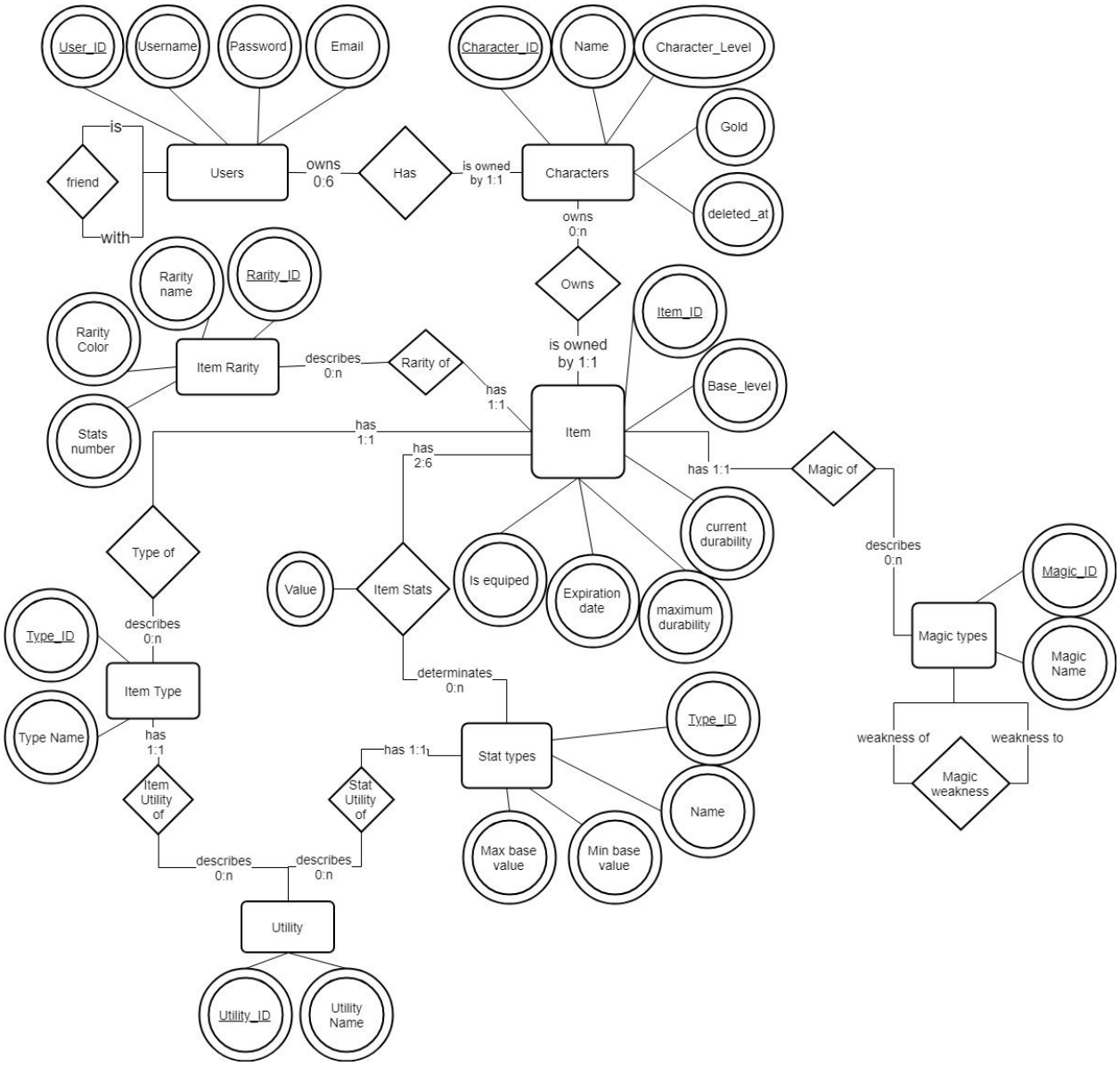
Prezentarea proiectului
RPG Items PSGBD
Lipan Radu-Matei A1,
Damian Andrei A1

1. Descrierea aplicatiei
2. Schema E/R
3. In ce forma normala este baza de date
4. Script de populare
5. Structuri de date utilizate (indexuri, views, tabele)
6. Lista de functii si proceduri

1. Descrierea aplicatiei

Aplicatia noastra consta intr-o baza de date ce stocheaza obiectele (items) dintr-un joc de tip RPG (Role Playing Game). Fiecare jucator va avea un cont. Fiecare caracter va avea un inventar de items (arme, armuri), iar fiecare item va avea diferite bonusuri, raritati, atribute. Jucatorii vor putea primi items generate dinamic (lootbox) , vor putea face tranzactii cu prietenii lor sau vor putea vizualiza statistici despre un caracter al acestora, statistici influentate de ce items are caracterul respectiv echipate.

2. Schema E/R

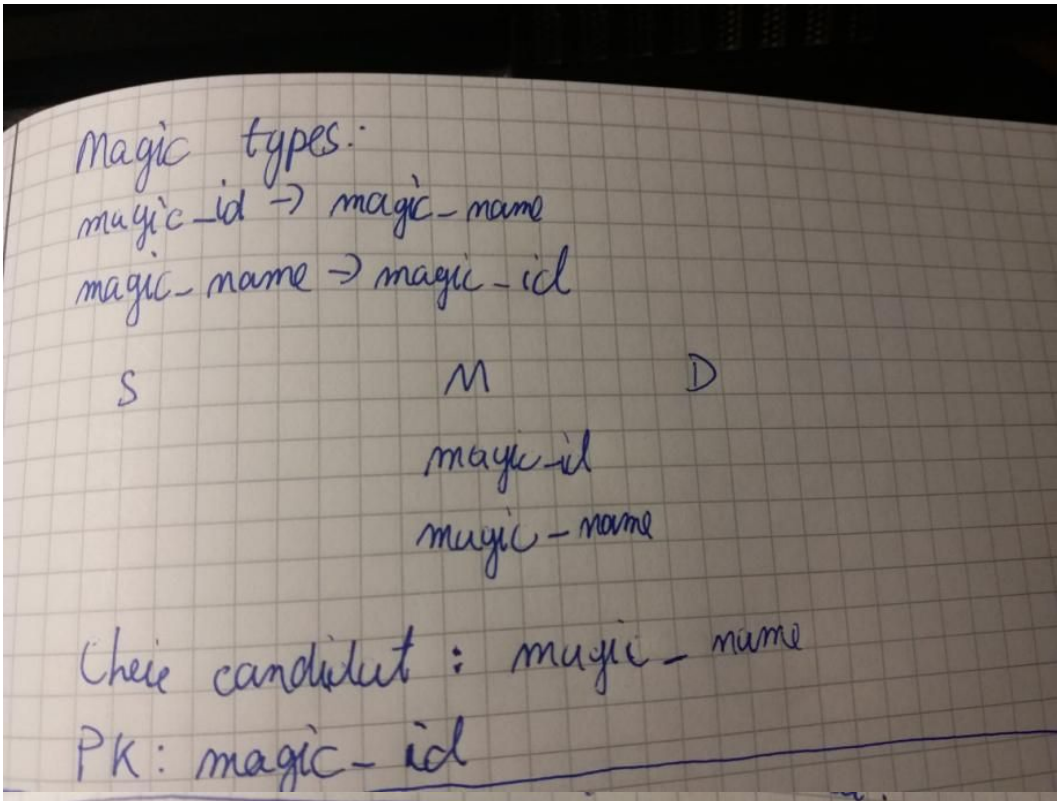

User_ID
Username
Password
Email
Character_ID
Name
Character_Level
Gold
deleted_at
is
friend
with
Users
owns
0:6
Has
is owned
by 1:1
Characters
owns
0:n
Owns
is owned
by 1:1
Rarity
name
Rarity_ID
Rarity
Color
Item Rarity
describes
0:n
Rarity of
has
1:1
Item_ID
Base_level
Stats
number
has
1:1
has
2:6
Item
has 1:1
Magic of
describes
0:n
Type of
describes
0:n
Value
Item Stats
Is equiped
Expiration
date
maximum
durability
current
durability
Magic_ID
Magic types
Magic
Name
Type_ID
Item Type
Type Name
has
1:1
determinates
0:n
Stat types
has 1:1
Type_ID
Name
Max base
value
Min base
value
weakness of
weakness to
Magic
weakness
Item
Utility of
Stat
Utility of
describes
0:n
describes
0:n
Utility
Utility_ID
Utility
Name

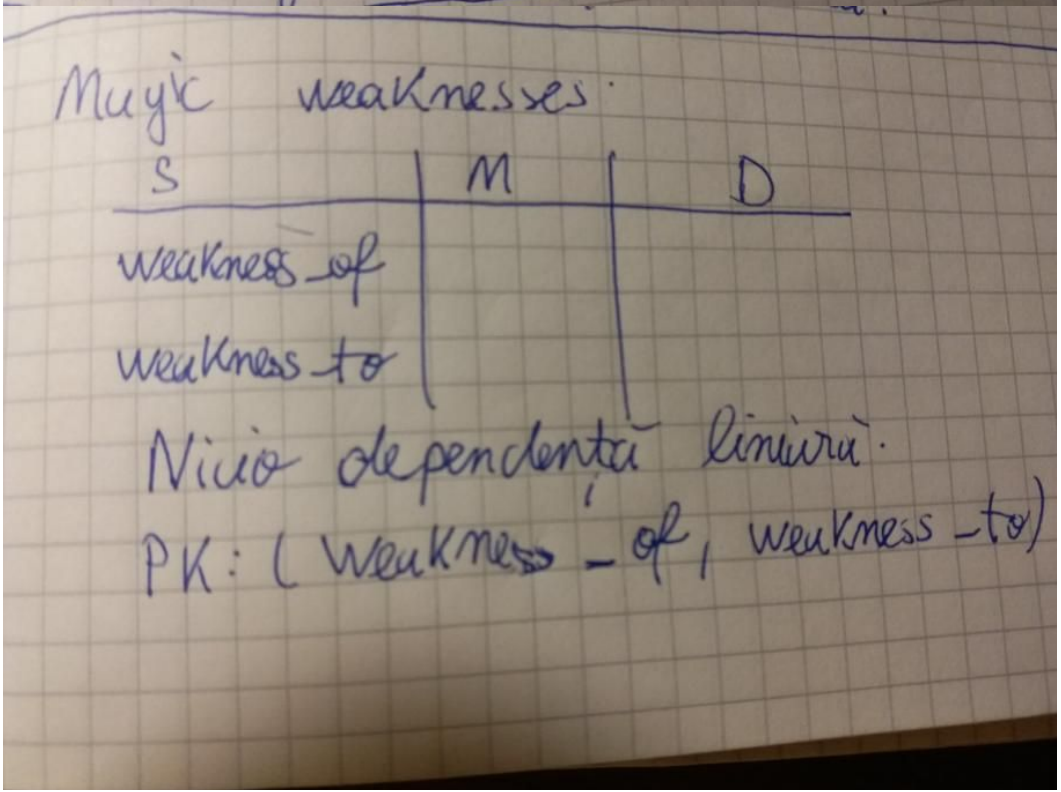
3. In ce forma normala este baza de date

3.1. 1NF

Baza noastra de date este in 1NF deoarece atributele sunt indivizibile si valoare fiecaruia este atomica.

3.2. Dependente liniare, chei candidat si chei primare:





Item type

type_id → utility_id, type_name

type_name → type_id, utility_id

S	M	D
	type_id type_name	utility_id

Chei candidat: type_name

PK: type_id

Utility

utility_id → utility_name

utility_name → utility_id

S	M	D
	utility_id utility_name	

Chei candidat: utility_n

PK: utility_id

Stat types:

type_id → utility_id, name, min_base_value, max_base_v

name → type_id, utility_id, min_base_value, max_base_

S	M	D
	~~type_id~~ name utility_id	min_base_value max_base_value type_id

Chei candidat: name

PK: utility_id

ITEM_RARITY

Rarity_ID → Name, color, stats_number

Name → Rarity_ID, color, stats_number

color → Name, Rarity_ID, stats_number

S	M	D
	Rarity_ID Name Color	stats_number

Chei candidat: Rarity_ID, Name, Color

PK: Rarity_ID

FRIENDS

Nu avem dependențe funcționale

S	M	D
User_ID1 User_ID2		

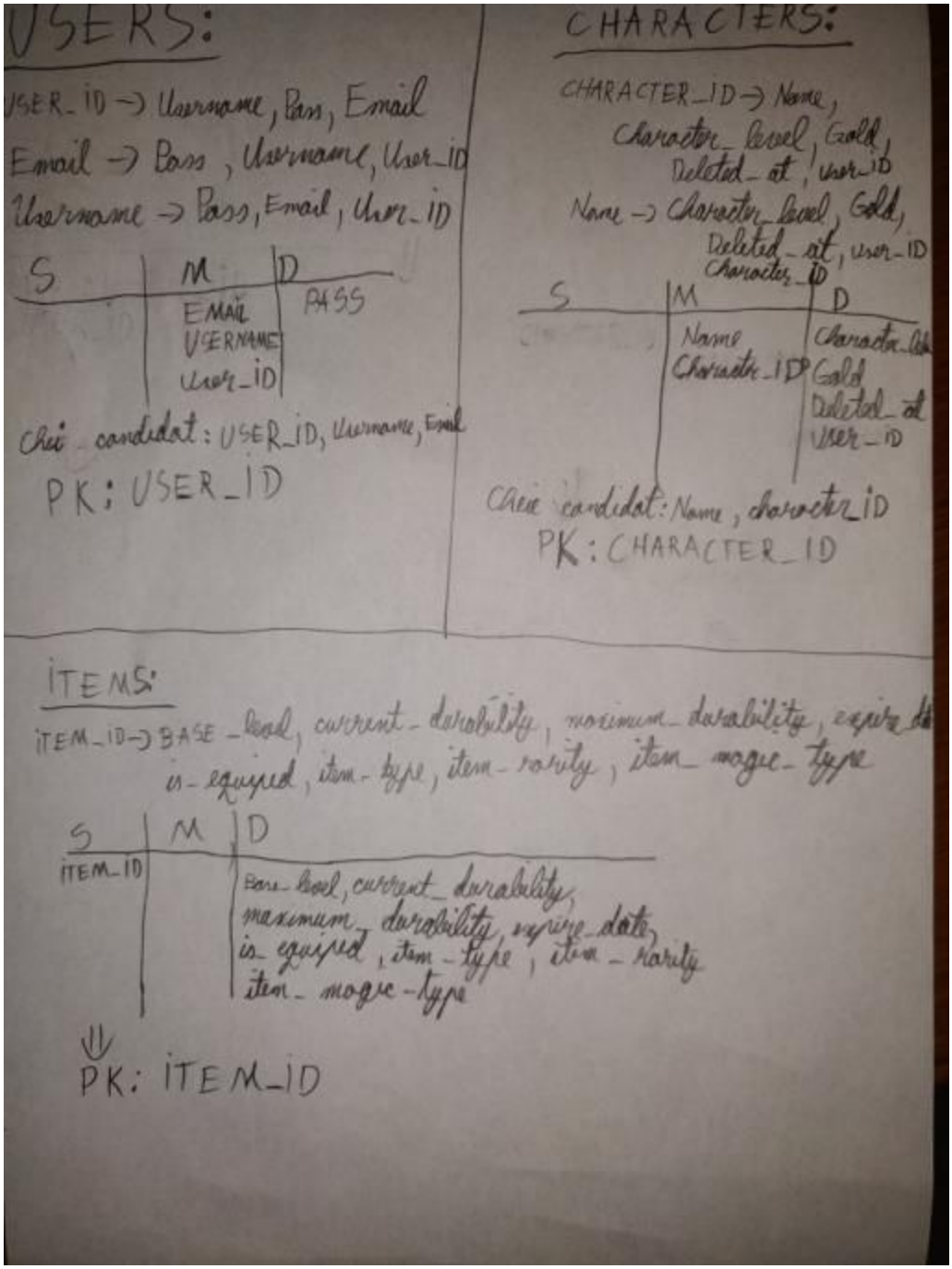
PK: (User_ID1, User_ID2)

ITEM_STATS

ITEM_ID, TYPE_ID → VALUE

S	M	D
ITEM_ID TYPE_ID		VALUE

PK(ITEM_ID, TYPE_ID)



*Pozele atat pentru normalizare cat si diagrama E/R sunt si pe git: https://github.com/xTachyon/rpg_items/tree/master/poze%20normalizare *

3.3. 2NF

O schema este in 2NF daca orice atribut neprim este dependent plin de orice cheie. Toate cheile candidat, cu exceptia cheilor primare pentru

rpg_item_stats, rpg_friends si rpg_magic_weaknesses sunt formate dintr-un singur atribut.

In cele trei tabele de mai sus cheia este formata din doua atribute, iar restul atributelor sunt dependente pline de catre aceste chei deoarece nu exista alte dependente functionale in tabelele respective de la atribute din interiorul acestor chei.

3.4. BCNF si 3NF

O schema este in BCNF daca este in 1NF si pentru orice dependenta functionala netriviala X -> A, X este supercheie. O schema ce este in BCNF este in 3NF.

In toate dependentele functionale din schema noastra, rolul lui X este luat chiar de o cheie candidat. O cheie candidat este supercheie, rezulta ca schema este in BCNF, deci este si in 3NF.

3.5. 4NF

O schema de relatie este in 4NF daca este in 1NF si pentru orice dependenta multivaluata netriviala X ->-> A, X este supercheie.

Pentru 4NF, nu am identificat dependente multivaluate netriviale in relatia noastra. Rezulta ca relatia noastra este in 4NF.

4. Script de populare

Script-ul de populare (rpg_tables_init.sql) lucreaza aproximativ 8-9 minute. Acesta creaza 25.000 de utilizatori, fiecare din acestia avand intre 2 si 6 caractere ducand la un numar de aproximativ 100.000 de caractere in total. Avem si 100.000 de relatii de prietenie intre utilizatori. Fiecarui caracter ii vor fi create intre 5 si 10 obiecte (items) random fiecare dintre acestea avand intre 2 si 6 atribute unice (item stats) create tot random. Astfel, ajungem sa avem aproximativ 1 milion de intrari in tabela rpg_items si 5 milioane de intrari in tabela rpg_item_stats. Restul tabelelor sunt umplute cu date stabilite in prealabil care ajuta la generarea acestor obiecte (items) dar si la mentinerea logicii jocului (ce tipuri de arme sau armuri exista, care sunt puternice impotriva carora etc).

5. Structuri de date utilizate (indexuri, views, tabele)

5.1. Indexuri:

Vom avea indexuri de tip B+ arbore pe rpg_item_stats(item_id), rpg_items(owner_id) si rpg_characters(user_id). Acestea sunt cheile straine care asigura legatura intre cele mai importante si mai populate tabele ale bazei de date. Ele vor aparea si in componenta celor mai frecvente interogari ale tabelei, aflarea datelor despre: caracterele unui utilizator, obiectele (items) ale unui caracter si atributele (stats) unui anumit obiect.

5.2. Views:

1. rpg_friends_view - selecteaza, intr-un format citibil de oameni, care utilizatori sunt prieteni cu care. De exemplu, spune in format text ca X e prieten cu Y, rezolvand id-urile utilizatorilor.
2. rpg_magic_weaknesses_view - afiseaza, in text, ce elemente sunt mai puternice sunt mai puternice decat altele;
3. rpg_items_view - selecteaza itemele din tabelul de iteme, rezolvand id-urile pentru utilitati, raritati, caracteristici si categorii;
4. rpg_stats_view - afiseaza, intr-un format citibil, id-ul itemului, numele caracteristicii si valoare acesteia.

5.3. Tabele:

1. rpg_users - contine utilizatorii aplicatiei, avand atributele username si parola, cu care se autentifica, si email;
2. rpg_friends - o relatie de prietenie intre doi utilizatori este reprezentata de o linie in aceasta tabela. Ordinea nu conteaza, (id1, id2) reprezentand aceeasi prietenie ca (id2, id1). Daca A este prieten cu B, atunci neaparat si B este prieten cu A;
3. rpg_characters - fiecare utilizator poate avea mai multe caractere de joc(maxim 6). Caracterele au un nume, un nivel si un numar de bani.
4. rpg_item_rarity - itemele au o diferita rarite, care semnifica cat de greu sunt de obtinut. Itemele care sunt mai greu de obtinut au si statusurile mai bune decat cele mai usor de obtinut. Raritatile sunt: Comun(Common), Rar(Rare), Epic, Legendar(Legendary).

5. rpg_magic_types - itemele au fiecare un element primordial pe care il reprezinta, iar cele mai slabe iteme sunt comune. Elemente: Foc(Fire), Apa(Water), Aer(Air), Earth(Pamant).
6. rpg_magic_weaknesses - unele elemente au avantaj contra altora. De exemplu, Apa bate Focul. Astfel, un item reprezentant de semnul Apei va fi mai puternic impotriva unui item reprezentat de Foc. In aceasta tabela se pastreaza cine bate pe cine.
7. rpg_item_utilities - itemele se impart in mai multe categorii: iteme de atac, iteme de aparare, etc. Acest tabel descrie aceasta caracteristica a unui item.
8. rpg_item_types - categorii mai pe larg in jurul la item_utilities. De exemplu sabiile, pumnalele, sulitele intra la iteme de atac, dar coiful, pantalonii, papucii intra la iteme de aparare.
9. rpg_stat_types - tipurile de caracteristici concret, valoarea minima si maxima a acestuia, tipul de utilitate si de item pe care se poate aplica bonusul. De exemplu, pentru un item de atac, acel item fiind o sabie, putem avea caracteristica Daune(Attack Damage), care are valoarea minima 1 si valoarea maxima 3.
10. rpg_items - un item care este detinut de un caracter. Fiecare item are o durabilitate curenta si o durabilitate maxima, durabilitatea scazand de fiecare data cand itemul este folosit. Itemele au o data de expirare, care poate fi null, in cazul in care itemul nu expira niciodata. De asemenea, itemul e descris de raritate, tipul de utilitate, tipul de element si daca e echipat sau nu.
11. rpg_item_stats - pe langa atributele cu care este descris fiecare item in tabela rpg_items, in aceasta tabela mai sunt descriere si caracteristicile itemelor(exemplu Daune, Aparare). Fiecare item are minim o caracteristica si maxim 5.

6. Lista de functii si proceduri

- procedure add_friend(User first, User second)
  Se apeleaza atunci cand un utilizator adauga un alt utilizator ca prieten, si al doilea utilizator accepta.

- function get_character_items(number character_id) -> ItemsArray
  Luand id-ul unui caracter, returneaza toate itemele acestuia.
- function get_all_characters(number user_id) -> CharactersArray
  Returneaza toate caracterele unui utilizator;
- procedure delete_all_removed_characters()

Cand sunt sterse, caracterele sunt de fapt marcate ca sterse, adica este completata coloana `deleted_at` cu data la care au fost sterse. La intervale regulate de timp chiar sunt sterse din baza de date, pentru a salva memorie(de exemplu, la un anumit interval de zile).

- function calculate_health(number character_id) -> number
  In functie de ce iteme de defensiva sunt echipate si de nivelul caracterului se calculeaza viata acestuia si se returneaza
- function generate_item(number character_id, number min_rarity) -> number

Genereaza un random item (de lootbox), cu o anumita raritate minima. Functia adauga itemul in inventarul caracterului dat ca parametru. Returneaza id-ul itemului generat.

Informatii despre ce ofera fiecare jucator: itemele si goldul.

- procedure do_trade(TradeInfo first, TradeInfo second)

Se apeleaza cand doi jucatori decid sa faca schimb de iteme si gold, primul caracter primind itemele si goldul oferit de al doilea, si invers.

  - type TradeInfo = (number character_id, ItemsArray items, number gold);
- function calculate_character_stats(number character_id) -> StatsReportArray

Calculeaza atributele caracterului(de exemplu, puterea de atac(daunele) si apararea), luand in considerare itemele echipate si nivelul acestuia.

  - type StatsReport = (number stat_type_id, number stat_value);
- function battle_characters(number first_character_id, number second_character_id) -> number

Simuleaza o batalie intre cele doua caractere pe baza itemelor pe care acestea le poseda. Se calculeaza puterea de atac si aparare a fiecarui

caracter si cel care ar reusi sa aduca primul punctele de viata ale oponentului la 0 castiga. Returneaza id-ul caracterului care ar castiga.

- function calculate_items_differences(Item first, Item second) -> StatsReportArray

Compara atributele, pentru doua iteme din aceeasi categorie(utilitate) si pune pentru fiecare atribut diferenta in StatsReportArray.

- procedure equip_item(number item_id)

Echipeaza un item. Daca un item de acelasi tip este deja echipat, atunci il dezechipeaza inaintea echiparii. De exemplu, vom putea avea doar un item cu utilitatea atac echipat la un moment dat.

- function upgrade_item(number item_id) -> bool

Imbunatateste atributele unui item(cu un procent mic si random). Itemul poate trece de la raritatea in care e la urmatoarea raritate, si poate creste si nivelul de baza al itemului. Un item legendar nu mai poate fi imbunatatit. Exista o sansa mare ca itemul sa fie distrus in proces. Returneaza daca itemul a supravietuit sau nu.

- procedure sell_item(number seller_character_id, number item_id)

Vinde un item cu un pret decis de aplicatie in functie de raritatea si nivelul acestuia. Caracterul primeste suma de bani pe care a fost vandut itemul.