Groupe n°1: Projet Sur Base de donnée d'un Jeu Vidéo

Fanny GOUPIL[1], Fabien CABALLERO[2], Ari ÇILINGIROGLU[3], and Kaan Egemen SEN[4]

[1]n° etudiant : 21811054
[2]n° etudiant : 22009817
[3]n° etudiant : 21508927
[4]n° etudiant : 21611799

December 19, 2021

Contents

1 Introduction

Nous avons conçu une base de données en MYSQL sur phpmyadmin pour un jeu vidéo.
Dans ce jeu, un joueur doit affronter des "monstres" sur des maps.
Il doit vaincre tous les ennemis sur une map pour pouvoir avoir accès à un coffre dans lequel il peut avoir de nouveaux équipements qu'il peut equiper si il a le niveau requis. Nous avons eu l'idée de concevoir cette base de données pour ensuite l'utiliser dans un projet personnel qui consiste à créer ce jeu.

On s'intéresse au stockage des informations liées au jeu:
Lorsqu'un joueur joue au jeu, il choisit un Personnage caractérisé par: un numéro personnage, un nom, un level (niveau), des points de vie et une race.
Lorsque les points de vie sont à 0, le personnage meurt et le joueur perd (le score obtenu sur la tentative en cours est mis à 0).
Un joueur peut faire 0 ou plusieurs tentatives, une tentative étant caractérisé par une idTentative et un score obtenu.
Une tentative se déroule sur une Map caractérisée par un identifiant de map, un nombre d' ennemis, un type de zone et une difficulté.

Pour cette tentative, le joueur rencontre des ennemis qui sont d'autres personnages. Il peut les combattre. Lorsqu'il en bat un, le niveau de son ennemi est ajouté au sien et son score pour cette tentative est augmenté de la difficulté de la map.
A chaque fois, que chacun d'eux prend des dégâts, les points de vie des personnages sont mis à jour. Le joueur a, à sa disposition, 2 équipements (une armure et une arme) caractérisé par un identifiant d'équipement, une rareté, un niveau requis et un type.
Une armure possède des points de défense.
Une arme possède des points d'attaque.
Un équipement peut être possédé par plusieurs personnages.
Un joueur ne peut posséder qu'une seule armure et qu'une seule arme .
Lors d'un combat entre 2 personnages, si l'armure a ses points de défense strictement supérieur aux points d'attaque de l'arme de l'ennemi alors le personnage subissant l'attaque perd en points de vie:

$$ptsDeVieEnleve = pointsAttaque - (ptsDefense - ptsAttaque)$$

Si ptsDeVieEnleve est un résultat négatif. On enlève 0 points de vie.

2 Dictionnaire de Donnée

Entité Equipement

Attribut	Type	Longueur	Nature	Description
idEquipement	int	2^{32}	obligatoire et unique,	clé primaire table Equipement
rarete	Chaine de caractères	50	facultatif	rareté d'un équipement, définit par "normal","rare","épique" ou "légendaire"
niveauRequis	Numérique	999	facultatif	niveau requis pour utiliser l'équipement. Par défaut à 0
typeEquipement	Chaine de caractères	30	facultatif	type de l'équipement

Entité Arme

Attribut	Type	Longueur	Nature	Description
ptsAttaque	Numérique	99	facultatif	point d'attaque d'une arme. Par défaut 1.

Entité Armure

Attribut	Type	Longueur	Nature	Description
ptsDefense	Numérique	99	facultatif	point de défense d'une armure. Par défaut 1.

Entité Personnage

Attribut	Type	Longueur	Nature	Description
idPersonnage	int	2^{32}	obligatoire et unique	clé primaire table Personnage. NOT NULL
nom	Chaine de caractères	50	facultatif	nom du personnage
niveau	Numérique	999	facultatif	niveau du personnage. Par défaut 0.
pointsDeVie	Numérique	99	facultatif	Points de vie du personnage. Par défaut 99.
race	Chaine de caractères	50	facultatif	Race du personnage parmi :'Goblin', 'Elfe', 'Troll','Humain'

Entité Map

Attribut	Type	Longueur	Nature	Description
idMap	int	2^{32}	obligatoire et unique	clé primaire table Map
nbEnnemi	Numérique	99	facultatif	nombre d'ennemi total sur la map
typezone	Chaine caractères	50	facultatif	type de zone (exemple: plage)
difficulte	Numérique	9	facultatif	niveau de difficulté de la map. Par défaut 1.

Entité Tentative

Attribut	Type	Longueur	Nature	Description
idTentative	int	2^{32}	obligatoire et unique	clé primaire table Tentative
ScoreObtenu	Chaine caractères	3	facultatif	score obtenu lors d'une tentative. Par défaut 0.

Entité Coffre

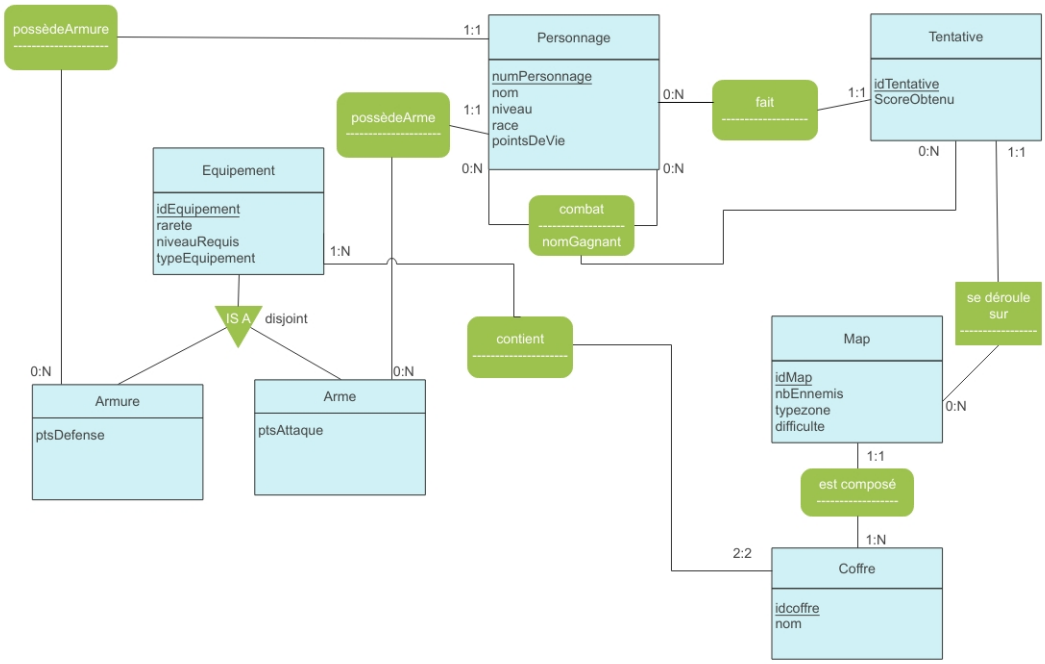
Attribut	Type	Longueur	Nature	Description
idCoffre	int	2^{32}	obligatoire et unique	clé primaire table Coffre
nom	Chaine de caractères	50	facultatif	nom du coffre

Association Combat

Attribut	Type	Longueur	Nature	Description
nomGagnant	Chaine de caractères	6	facultatif	Spécifie gagnant du combat soit par 'Joueur' ou 'Ennemi'.

3 Création

3.1 Schéma Entité Association



3.2 Modèle Relationnel

Equipement(idEquipement, rarete, niveauRequis, typeEquipement)
Armure(#idArmure, ptsDefense)
Arme(#idArme, ptsAttaque)
Personnage(numPersonnage, nom, niveau, race, pointsDeVie, #idArmure, #idArme)
Coffre(idcoffre, nom, #Equipement1, #Equipement2)
Map(idMap, nbEnnemis, typezone, difficulte, #idcoffre)
Tentative(idTentative,ScoreObtenu,#idPersonnage, #idMap)
Combat(#idJoueur, #idEnnemi,#idTentative,nomGagnant)

3.3 Schéma physique

```
create table Equipement(
    idEquipement int AUTO_INCREMENT,
    rarete  varchar(50) check (rarete in ("normal","rare","épique","légendaire")),
    niveauRequis numeric(3,0),
    typeEquipement varchar(30),
    constraint pk_equipement primary key (idEquipement)
);

create table Armure(
    idArmure int,
    ptsDefense numeric(2,0),
    constraint pk_Armure primary key (idArmure),
    constraint fk_idEquipement_Equipement_Armure foreign key (idArmure) references
    Equipement(idEquipement)


);
create table Arme(
    idArme int,
    ptsAttaque numeric(2,0),
    constraint pk_Arme primary key (idArme),
    constraint fk_idEquipement_Equipement_Arme foreign key (idArme) references
    Equipement(idEquipement)

);
create table Personnage(
    numPersonnage int AUTO_INCREMENT,
    nom varchar(50),
    niveau numeric(3,0),
    race varchar(50) check (race in ("Goblin","Elfe","Troll","Humain")),
    pointsDeVie numeric(2,0),
    idArmure int,
    idArme int,
    constraint pk_personnage primary key (numPersonnage),
    constraint fk_idArmure_Armure foreign key (idArmure) references Armure(idArmure),
    constraint fk_idArme_Arme foreign key (idArme) references Arme(idArme)

);

create table Coffre (
    idcoffre int AUTO_INCREMENT,
    nom varchar(50),
    Equipement1 int,
    Equipement2 int,
    constraint pk_coffre primary key (idcoffre),
    constraint fk_Equipement1_Equipement foreign key (Equipement1) references Equipement
    (idEquipement),
    constraint fk_Equipement2_Equipement foreign key (Equipement2) references Equipement
    (idEquipement)
);

create table Map(
    idMap int AUTO_INCREMENT,
    nbEnnemis numeric(2,0),
    typezone varchar(50),
    difficulte numeric(1,0),
    idcoffre int,
    constraint pk_map primary key (idMap),
    constraint fk_idcoffre_coffre foreign key (idcoffre) references Coffre(idcoffre)
);

```

```
58  create table Tentative (
59      idTentative int AUTO_INCREMENT,
60      ScoreObtenu numeric(5,0),
61      idPersonnage int,
62      idMap int,
63      constraint pk_tentative primary key (idTentative) ,
64      constraint fk_idPersonnage foreign key (idPersonnage) references Personnage(
        numPersonnage),
65      constraint fk_idMap foreign key (idMap) references Map(idMap)
66
67  );
68
69  create table Combat(
70      idJoueur int,
71      idEnnemi int,
72      idTentative int,
73      nomGagnant varchar(6) check (nomGagnant in ('Joueur','Ennemi')),
74      constraint pk_combat primary key (idJoueur,idEnnemi,idTentative),
75      constraint fk_idJoueur_Personnage foreign key (idJoueur) references Personnage(
        numPersonnage),
76      constraint fk_idEnnemi_Personnage foreign key (idEnnemi) references Personnage(
        numPersonnage),
77      constraint fk_idTentative_Tentative foreign key (idTentative) references Tentative(
        idTentative)
78  );
```

4 Requêtes

4.1 Group by

Affichage du nombre de personne d'une même race ayant le meme type d'équipement

```
SELECT typeEquipement,race, COUNT(P.numPersonnage) as nbPersosDeCetteRaceAyantCetteArme
FROM Personnage P, Equipement E
WHERE P.idArme=E.idEquipement
GROUP BY typeEquipement,race;
```

4.2 Division

Un personnage est sélectionné s'il n'existe aucun coffre pour lequel il y a à l'intérieur l'arme de ce personnage.

```
SELECT *
FROM Personnage
WHERE NOT EXISTS (SELECT *
                  FROM Coffre
                  WHERE NOT EXISTS (SELECT *
                                    FROM Arme
                                    WHERE Coffre.Equipement1!=Arme.idArme
                                          AND Coffre.Equipement2!=Arme.idArme
                                          AND Personnage.idArme=Arme.idArme ));
```

4.3 Sous requête

Liste des personnes qui ont leur armure de rareté "rare".

```
SELECT * FROM Personnage
WHERE idArmure IN (SELECT idArmure FROM Equipement,Armure
                   WHERE Equipement.idEquipement=Armure.idArmure
                   AND Equipement.rarete="rare");
```

4.4 Sous requête corrélative

Affiche les personnages qui n'ont jamais combattu en tant que joueur

```
SELECT * FROM Personnage
WHERE NOT EXISTS( SELECT * FROM Combat
                  WHERE Combat.idJoueur=Personnage.numPersonnage);
```

5 Procédures, Fonctions, Triggers

5.1 Triggers

5.1.1 Reinitialisation Points de Vie

Ce trigger est appelé avant qu'un combat soit inséré.
A la fin du combat *(lorsque les points de vie d'un des personnages sont à 0)*, on doit reinitialiser les points de vie des 2 personages si ils ont pris des dégats. *(On reinitialise à 99).*

```
CREATE TRIGGER `ReinitialisationPtsVie` BEFORE INSERT ON `Combat`
FOR EACH ROW BEGIN
    DECLARE v_ptvChar integer;

    SELECT pointsDeVie INTO v_ptvChar
    FROM Personnage
    WHERE numPersonnage = new.idJoueur;

    IF(v_ptvChar < 99) THEN
        UPDATE Personnage SET pointsDeVie=99 WHERE numPersonnage = new.idJoueur;
    END IF;

    SELECT pointsDeVie INTO v_ptvChar
    FROM Personnage
    WHERE numPersonnage = new.idEnnemi;

    IF(v_ptvChar < 99) THEN
        UPDATE Personnage SET pointsDeVie=99 WHERE numPersonnage = new.idEnnemi;
    END IF;
END
```

5.1.2 Niveau Requis

Trigger qui s'exécute avant la mise a jour d'un personnage.
On vérifie que le niveau du personnage est supérieur ou égal au niveau requis de l'équipement. Si cette condition est vérifiée, on ajoute l'équipement, sinon, on affiche le message 'niveau pas assez haut'.

```
CREATE TRIGGER `NiveauRequisEquipement` BEFORE UPDATE ON `Personnage`
FOR EACH ROW BEGIN
    DECLARE v_nivRecq integer;
    SELECT niveauRequis into v_nivRecq
    FROM Equipement
    WHERE Equipement.idEquipement=new.idArmure;

    IF v_nivRecq >  new.niveau THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'niveau pas assez haut';
    END IF;
END
```

5.2 Procedure

Cette procédure fait un classement des scores par race. Un score d'une race est obtenu en faisant la somme de toutes les tentatives d'une race.

Exemple:
Goblin 3000 (1er goblin Tentative 1 score: 300 + 1er goblin Tentative 2 score: 350 + ...)
Troll 2900
Elfe 2000
Humain 600

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `classementScores`()
BEGIN
    SELECT race,sum(ScoreObtenu) as TotScore
    FROM Tentative,Personnage
    WHERE Tentative.idPersonnage=Personnage.numPersonnage
    GROUP BY race
    ORDER BY sum(ScoreObtenu) DESC;
END
```

5.3 Fonction

Le joueur attaque l'ennemi *(voir introduction pour la gestion des dommages)*.
L'ennemi attaque le joueur *(idem)* et ça jusqu'a ce que l'un des deux meurt. Le combat est ajouté une fois terminé avec le bon attribut nomGagnant:

- Si c'est le joueur qui gagne le niveau de l'ennemi est ajouté à celui du joueur et son score est incrémenté de la difficulté de la map pour cette tentative.
- Si le joueur perd son score est mis à 0 pour cette tentative

```
CREATE DEFINER=`admin`@`localhost` FUNCTION `combattre`(`idJoueur` INT, `idEnnemi` INT,
    `idT` INT) RETURNS varchar(6) CHARSET utf8mb4
DETERMINISTIC CONTAINS SQL SQL SECURITY DEFINER
BEGIN
    DECLARE
    pdvJ NUMERIC(2, 0);
    DECLARE pdvE NUMERIC(2, 0);
    DECLARE nivE NUMERIC(3, 0);
    DECLARE ptsAt NUMERIC(2, 0);
    DECLARE ptsDef NUMERIC(2, 0);
    DECLARE ptsEnleve NUMERIC(3, 0);
    DECLARE difficulteMap NUMERIC(1, 0);

    WHILE 1=1 DO
        #Joueur attaque Ennemi
        SELECT pointsDeVie,ptsDefense,niveau INTO pdvE, ptsDef, nivE
        FROM Personnage,Armure
        WHERE numPersonnage = idEnnemi AND Personnage.idArmure = Armure.idArmure;

        SELECT ptsAttaque INTO ptsAt
        FROM Personnage,Arme
        WHERE numPersonnage = idJoueur AND Personnage.idArme = Arme.idArme;

        IF ptsAt < ptsDef THEN
            SET ptsEnleve = ptsAt -(ptsDef - ptsAt);
```

```
25          ELSE
26              SET ptsEnleve = ptsAt;
27          END IF;
28
29
30          IF ptsEnleve <=0 THEN
31                      SET ptsEnleve=0;
32          ENDIF;
33
34          UPDATE Personnage SET pointsDeVie = pdvE - ptsEnleve WHERE numPersonnage =
        idEnnemi;
35
36          IF pdvE - ptsEnleve <= 0 THEN
37              INSERT INTO Combat VALUES(idJoueur,idEnnemi,idT,'Joueur');
38              UPDATE Personnage SET niveau = niveau + nivE WHERE numPersonnage = idJoueur;
39
40              SELECT difficulte INTO difficulteMap
41              FROM Tentative, Map
42              WHERE Tentative.idTentative = idT  and Map.idMap=Tentative.idMap;
43
44              UPDATE Tentative SET scoreObtenu = scoreObtenu + difficulteMap
45              WHERE idTentative = idT;
46
47              RETURN 'Joueur';
48
49          ELSE
50              # Ennemi attaque Joueur
51              SELECT pointsDeVie, ptsDefense INTO pdvJ, ptsDef
52              FROM Personnage,Armure
53              WHERE numPersonnage = idJoueur AND Personnage.idArmure = Armure.idArmure;
54
55              SELECT ptsAttaque INTO ptsAt
56              FROM Personnage, Arme
57              WHERE numPersonnage = idEnnemi AND Personnage.idArme = Arme.idArme;
58
59              IF ptsAt < ptsDef THEN
60                  SET ptsEnleve = ptsAt -(ptsDef - ptsAt);
61              ELSE
62                  SET ptsEnleve = ptsAt;
63              END IF;
64
65              IF ptsEnleve <=0 THEN
66                      SET ptsEnleve=0;
67              END IF;
68
69              UPDATE Personnage SET pointsDeVie = pdvJ - ptsEnleve
70              WHERE numPersonnage = idJoueur;
71
72              IF pdvJ - ptsEnleve <= 0 THEN
73                  INSERT INTO Combat
74                  VALUES(idJoueur,idEnnemi,idT,'Ennemi');
75
76                  UPDATE Tentative SET scoreObtenu = 0
77                  WHERE idTentative = idT;
78                  RETURN 'Ennemi';
79              END IF;
80          END IF;
81      END WHILE;
82  END
```

6 Tests

6.1 Test Procédure classementScores

```
1 CALL classementScores();
```

6.2 Test Fonction combattre

```
1 SELECT combattre(10,5,1);
```

6.3 Test Trigger points de vie

```
1 SELECT numPersonnage,pointsDeVie
2 FROM Personnage
3 WHERE numPersonnage=5 OR numPersonnage=10;
```

6.4 Test Trigger niveau requis

```
1 UPDATE Personnage SET idArmure=18 WHERE numPersonnage=1;
```