Table des matières

[Création de la table Combat 3](#_Toc437371534)

[Fonction 3](#_Toc437371535)

[fn\_BuyEquipment 3](#_Toc437371536)

[fn\_HealSpell 4](#_Toc437371537)

[fn\_MobStat 4](#_Toc437371538)

[fn\_NonObtainedQuest 5](#_Toc437371539)

[fn\_SpellDmg 5](#_Toc437371540)

## Création de la table Combat

--combat courant des personnages contient quelle personnage se bat contre quel monstre ainsi que sa vie

CREATE Table Combat

(

CombatID int IDENTITY(0,1),

CombatCharactersID int NOT NULL,

CombatMonsterID int NOT NULL,

CombatMonsterHP int,

CombatTurn BIT NOT NULL,

CONSTRAINT CCombatID PRIMARY KEY(CombatID),

CONSTRAINT CCombatCharactersID FOREIGN KEY(CombatCharactersID) REFERENCES Characters(CharactersID),

CONSTRAINT CCombatMonsterID FOREIGN KEY(CombatMonsterID) REFERENCES Mob(MobID),

)

# Fonction

## fn\_BuyEquipment

--achete et retourne une nouvelle piece d'équipement

CREATE FUNCTION fn\_BuyEquipment(@CharID int)

RETURNS @Table TABLE(EquipID int)

AS

BEGIN

DECLARE @EquipID int--id de l'équipement qu'il va acheter

DECLARE @CharGold int--or du personnage

SELECT @CharGold = Characters.CharactersGold FROM Characters WHERE CharactersID = @CharID

SELECT TOP 10 percent @EquipID = Equipment.EquipID FROM Equipment WHERE EquipID NOT IN(SELECT CharactersEquipment.CharactersEquipmentEquipmentID

FROM CharactersEquipment INNER JOIN Equipment ON CharactersEquipmentEquipmentID = EquipID

WHERE CharactersEquipmentCharactersID = @CharID) AND EquipValue <= @CharGold

ORDER BY EquipValue DESC

INSERT INTO @Table(EquipID) VALUES(@EquipID)

RETURN

END

## fn\_HealSpell

--donne le meilleur spell de soin du personnage donnée

CREATE FUNCTION fn\_HealSpell (@CharID int)

RETURNS @Table TABLE(SpellID int, SpellHeal int)

AS

BEGIN

DECLARE

@Heal int,--spell de heal qui heal le plus

@ID int;--id du spell

SELECT TOP 1

@Heal = Spell.SpellHeal,

@ID = Spell.SpellID

FROM (Spell INNER JOIN SpellQuantity ON SpellID = SpellQuantitySpellID) INNER JOIN Characters ON CharactersID = SpellQuantityCharactersID

WHERE SpellQuantityCharactersID = @CharID AND Spell.SpellHeal > 0

ORDER BY SpellHeal DESC

INSERT INTO @Table(SpellID,SpellHeal) VALUES(@ID,@Heal)

RETURN

END

## fn\_MobStat

--donne les stat du monstre donnée

CREATE FUNCTION fn\_MobStat(@MobID int)

RETURNS @\_Table TABLE(MobDps int,MobExp int,MobGold int)

AS

BEGIN

DECLARE

@Dps int,--dégat du monstre

@Exp int,--exp donnée par le monstre

@Gold int;--gold donnée par le monstre

SELECT

@Dps = Mob.MobDPS,

@Exp = Mob.MobExpReward,

@Gold = Mob.MobGoldReward

FROM Mob

WHERE MobID = @MobID

INSERT INTO @\_Table(MobDps,MobExp,MobGold) VALUES(@Dps,@Exp,@Gold)

RETURN

END

## fn\_NonObtainedQuest

--donne une quête non obtenue par le joueur donnée

--doit ajouter le order by chiffre random

CREATE FUNCTION fn\_NonObtainedQuest(@CharID int)

RETURNS @Table TABLE(QuestID int)

AS

BEGIN

DECLARE @Result int--id de la quête au hasard

SELECT TOP 10 percent @Result = QuestID

from Quest

where QuestID not in (select QuestStatusQuestID

from QuestStatus WHERE QuestStatusCharactersID = @CharID)

INSERT INTO @Table(QuestID) VALUES(@Result)

RETURN

END

## fn\_SpellDmg

--donne le spell qui fait le plus de dégat du personnage donnée

CREATE FUNCTION fn\_SpellDmg(@CharID int)

RETURNS @Table TABLE(SpellID int, SpellDmg int)

AS

BEGIN

DECLARE

@Dmg int,--spell de heal qui heal le plus

@ID int;--id du spell

SELECT TOP 1

@Dmg = Spell.SpellDmg,

@ID = Spell.SpellID

FROM (Spell INNER JOIN SpellQuantity ON SpellID = SpellQuantitySpellID) INNER JOIN Characters ON CharactersID = SpellQuantityCharactersID

WHERE SpellQuantityCharactersID = @CharID AND Spell.SpellDmg > 0

ORDER BY SpellDmg DESC

INSERT INTO @Table(SpellID,SpellDmg) VALUES(@ID,@Dmg)

RETURN

END

# Procédure Stockée

--fait les actions selon les différents états du personnage

CREATE PROCEDURE sp\_Action(@CharID int)

AS

BEGIN

if((SELECT COUNT(\*) FROM Combat WHERE CombatCharactersID = @CharID) > 0)

--si personnage est en combat

BEGIN

DECLARE @TableMonster Table(Dps int,Exps int,gold int)

--contient les stat du monstre du combat

DECLARE @Turn bit--tour du combat

DECLARE @MonsterID int--id du monstre dans le combat

DECLARE @Hp int--vie courant du personnage en train de se battre

Select @Hp = CharactersCurrentHP FROM Characters WHERE CharactersID = @CharID

Select TOP 1 @MonsterID = Combat.CombatMonsterID FROM Combat

SELECT TOP 1 @Turn = Combat.CombatTurn FROM Combat

INSERT INTO @TableMonster

SELECT \* FROM fn\_MobStat(@MonsterID)

if(@Turn = 0)

BEGIN--Attack du monstre

Set @Hp = @Hp - ((SELECT Dps FROM @TableMonster) \*

(SELECT CharactersLevel FROM Characters WHERE CharactersID = @CharID)) /2

if(@Hp < 0)—si vie dans le négatife le met a 0

BEGIN

SET @Hp = 0

END

UPDATE Characters SET CharactersCurrentHP = @Hp

WHERE CharactersID = @CharID

UPDATE Combat Set CombatTurn = 1

Where CombatCharactersID = @CharID

END

Else--attack du perso

Begin

DECLARE @Random int--chiffre random qui determine l'action du joueur(spell ou attack normal)

Set @Random = ROUND(RAND() \* 100,0)

DECLARE @MaxHp int--vie maximale du personnage

SELECT @MaxHp = Characters.CharactersMaxHP FROM Characters WHERE CharactersID = @CharID

if(@Hp <= ((0.50) \* @MaxHp))-- lance un spell de soin

BEGIN

DECLARE @HealSpell TABLE(SpellID int,SpellHeal int)

--spell de soin qui va être lancé

INSERT INTO @HealSpell

SELECT \* FROM fn\_HealSpell(@CharID)

UPDATE Characters SET CharactersCurrentHP = CharactersCurrentHP + (Select SpellHeal FROM @HealSpell)

WHERE CharactersID = @CharID

UPDATE SpellQuantity SET SpellQuantityQuantity = SpellQuantityQuantity - 1

WHERE SpellQuantityCharactersID = @CharID AND

SpellQuantitySpellID = (Select SpellID FROM @HealSpell)

END

Else if(@Random <= 30)--spell d'Attack

BEGIN

DECLARE @DmgSpell TABLE(SpellID int,SpellDmg int)

--spell de dégat qui va être lancé

INSERT INTO @DmgSpell

SELECT \* FROM fn\_SpellDmg(@CharID)

if((((SELECT CombatMonsterHP from Combat WHERE CombatCharactersID = @CharID) –

(SELECT SpellDmg FROM @DmgSpell)) <= 1) AND ((SELECT SpellDmg FROM @DmgSpell) is not null))

BEGIN

UPDATE Combat SET CombatMonsterHP = 1

WHERE CombatCharactersID = @CharID

END

ELSE

Begin

UPDATE Combat SET CombatMonsterHP = CombatMonsterHP –

(SELECT SpellDmg FROM @DmgSpell)

WHERE CombatCharactersID = @CharID

END

UPDATE SpellQuantity SET SpellQuantityQuantity = SpellQuantityQuantity - 1

WHERE SpellQuantityCharactersID = @CharID AND

SpellQuantitySpellID =

(Select SpellID FROM @DmgSpell)

END

ELSE--attack normal

BEGIN

DECLARE @EquipID int--id de l'équipement équipé par le perso

DECLARE @PersoDmg int--dommage effectué par le perso

Set @PersoDmg = 0

DECLARE CurDmgs CURSOR FOR SELECT CharactersEquipment.CharactersEquipmentEquipmentID

FROM CharactersEquipment

--curseur qui parcours l'équipement du perso

WHERE CharactersEquipment.CharactersEquipmentCharactersID

= @CharID

OPEN CurDmgs

FETCH NEXT FROM CurDmgs INTO @EquipID

WHILE @@FETCH\_STATUS = 0

BEGIN

DECLARE @EquipStat int--force de l'équipement

SELECT @EquipStat = Equipment.EquipStrength FROM Equipment WHERE EquipID = @EquipID

Set @PersoDmg = @PersoDmg + @EquipStat

FETCH NEXT FROM CurDmgs INTO @EquipID

END

CLOSE CurDmgs

DEALLOCATE CurDmgs

DECLARE @RaceStr int--force de la race du perso

DECLARE @CharStr int--force de base du perso

SELECT @CharStr = Characters.CharactersStrength FROM Characters WHERE CharactersID = @CharID

Set @PersoDmg = @CharStr + @PersoDmg

SELECT @RaceStr = Race.RaceStrength FROM

Race INNER JOIN Characters ON

CharactersRaceID = Race.RaceID

WHERE CharactersID = @CharID

SET @PersoDmg = @RaceStr + @PersoDmg

if(((SELECT CombatMonsterHP from Combat

WHERE CombatCharactersID = @CharID) -

@PersoDmg) <= 1 )

--si le monstre va mourir au prochain coup

BEGIN

UPDATE Combat SET CombatMonsterHP = 1

WHERE CombatCharactersID = @CharID

END

ELSE--ne meurt pas au prochain coup

BEGIN

UPDATE Combat SET CombatMonsterHP = (CombatMonsterHP - @PersoDmg)

WHERE CombatCharactersID = @CharID

END

END

UPDATE Combat Set CombatTurn = 0

WHERE CombatCharactersID = @CharID

END

END

ELSE--pas en combat

BEGIN

DECLARE @ItemCount int--quantité d'item du perso

DECLARE @ItemID int--id de l'item

SET @ItemCount = 0

DECLARE CurItem CURSOR FOR SELECT ItemQuantity.ItemQuantityItemID FROM ItemQuantity WHERE ItemQuantityCharactersID = @CharID

OPEN CurItem

FETCH NEXT FROM CurItem INTO @ItemID

WHILE @@FETCH\_STATUS = 0

BEGIN

DECLARE @Quantity int--quantité de l'item courant

SELECT @Quantity = ItemQuantityQuantity FROM ItemQuantity WHERE ItemQuantityItemID = @ItemID

SET @ItemCount = @ItemCount + @Quantity

FETCH NEXT FROM CurItem INTO @ItemID

END

CLOSE CurItem

DEALLOCATE CurItem

if(@ItemCount < ((0.80) \* 255))--si il peut encore continuer a se battre(inventaire pas encore plein)

BEGIN

Declare @MobID int--id du monstre du prochain combat

Declare @MobHp int--vie du monstre du prochain combat

DECLARE @MobTable TABLE(MobID int,MobHp int)

--contient les infos du prochain monstre

INSERT INTO @MobTable

SELECT \* FROM view\_RndMob

SELECT @MobID = MobID FROM @MobTable

SELECT @MobHp = MobHp FROM @MobTable

INSERT INTO Combat(CombatCharactersID,CombatMonsterID,CombatMonsterHP,CombatTurn) VALUES(@CharID,@MobID,@MobHp,1)

END

ELSE

BEGIN --plein donc retourne au village

DECLARE @QuestID int--id de la quête en cours

DECLARE @QuestItemID int--item de la quête en cours

SELECT @QuestID = QuestStatusQuestID FROM QuestStatus

WHERE QuestStatusCharactersID = @CharID AND

QuestStatusCompleted = 0

SELECT @QuestItemID = QuestItemID FROM Quest

WHERE QuestID = @QuestID

--Quest Item a rajouté dans table Quest

--accomplie la quête

if((SELECT COUNT(\*) FROM ItemQuantity WHERE ItemQuantityCharactersID = @CharID AND

ItemQuantityItemID = @QuestItemID) > 0)

BEGIN

UPDATE QuestStatus SET QuestStatusCompleted = 1

WHERE QuestStatusCharactersID = @CharID

DELETE FROM ItemQuantity

WHERE ItemQuantityCharactersID = @CharID AND

ItemQuantityItemID = @QuestItemID

SELECT @QuestID = QuestID FROM fn\_NonObtainedQuest(76)

INSERT INTO

QuestStatus(QuestStatusObtained,QuestStatusCompleted,

QuestStatusQuestID,QuestStatusCharactersID) VALUES(1,0,@QuestID,@CharID)

END

--vidage d'inventaire

DECLARE CurItem CURSOR FOR SELECT ItemQuantityItemID FROM ItemQuantity WHERE ItemQuantityCharactersID = @CharID

OPEN CurItem

FETCH NEXT FROM CurItem INTO @ItemID

WHILE @@FETCH\_STATUS = 0

BEGIN

DECLARE @ItemGold int--valeur de l'item

DECLARE @ItemQuantity int--quantité de l'item

SELECT @ItemQuantity = ItemQuantityQuantity FROM ItemQuantity

WHERE ItemQuantityItemID = @ItemID

SELECT @ItemGold = Item.ItemValue FROM Item

WHERE ItemID = @ItemID

UPDATE Characters SET CharactersGold = @ItemGold\*@ItemQuantity + CharactersGold

DELETE FROM ItemQuantity WHERE ItemQuantityItemID = @ItemID AND ItemQuantityCharactersID = @CharID

FETCH NEXT FROM CurItem INTO @ItemID

END

CLOSE CurItem

DEALLOCATE CurItem

--Achat d'équipement

DECLARE @NewEquip int--id du nouvel équipement

DECLARE @EquipType int--type du nouvel équipement

SELECT @NewEquip = EquipID FROM fn\_BuyEquipment(@CharID)

if(@NewEquip is not null)--si retourne un équipment

Begin

SELECT @EquipType = Equipment.EquipTypeID FROM Equipment

WHERE EquipID = @NewEquip

-- a changer donne pas équipement en fonction de sa classe donne n'importe quelle type d'équipement

if((SELECT Count(\*) FROM CharactersEquipment INNER JOIN Equipment ON CharactersEquipmentEquipmentID = EquipID

WHERE EquipTypeID = @EquipType) = 0)

BEGIN--ajoute le nouvel équipement

INSERT INTO CharactersEquipment(CharactersEquipmentEquipmentID,

CharactersEquipmentCharactersID) VALUES(@NewEquip,@CharID)

END

ELSE

BEGIN--change le vielle équipement par le nouveau

DECLARE @OldEquip int--id du dernier équipement

SELECT @OldEquip = Equipment.EquipID FROM CharactersEquipment

INNER JOIN Equipment ON CharactersEquipmentEquipmentID = EquipID

WHERE EquipTypeID = @EquipType

UPDATE CharactersEquipment SET CharactersEquipmentEquipmentID = @NewEquip

WHERE CharactersEquipmentEquipmentID = @OldEquip

END

END

END

END

END