INFOH303 - Projet de Système d'Inventaire pour un RPG - Partie 2

Bilal Vandenberge, Lucas Verbeiren, Ethan Van Ruyskensvelde, Rares Radu-Loghin

11 juin 2025

Méthode d'extraction des données

Pour remplir la base de données, nous avons développé un script `‘insert.py‘` qui :

- Crée les tables si elles n'existent pas.
- Lit et extrait les données des fichiers du dossier `‘data‘`.
- Vérifie la validité des données avant insertion (notamment si une donnée similaire a déjà été insérée).

Choix d'implémentation des données et gestion des erreurs

- Contraintes d'intégrités en SQL, à l'aide du mot-clé `CHECK`, pour empêcher d'ajouter des données incorrectes dans la base de données.
- En cas d'insertion ou modification introduisant des valeurs qui ne respectent pas ces assertions, le binding `Python MySQL` lance une exception que nous interceptons pour éviter que le programme ne plante.
- Afin de fournir un message d'erreur dans de tels cas, nous vérifions les contraintes, dans le code Python, avant de modifier/insérer dans la base de données.
- Suppression `CASCADE` lorsqu'un utilisateur supprime son compte afin de supprimer tous les tuples qui concernaient cet utilisateur.

Requêtes demandées

Les requêtes SQL suivantes permettent d'extraire différentes informations :

- Top 10 joueurs avec le plus d'or.
- Joueur avec le plus de personnages de la même classe.
- Quête avec la plus grosse récompense en or.
- Le PNJ possédant l'inventaire contenant les objets dont la valeur en or cumulée est la plus importante
- Le type d'objet (arme, armure, potion ou relique) le plus souvent offert en récompense de quêtes de niveau 5
- Les monstres avec les meilleures récompenses en valeur en or cumulée en fonction de leurs points de vie

Les 10 joueurs ayant le plus d'or

SQL :

```
SELECT p.Name, p.Money
FROM Players p
ORDER BY p.Money DESC LIMIT 10;
```

Algèbre relationnelle : Impossible car cela nécessite d'agréger les quêtes par niveau de difficulté et de sélectionner celles ayant la récompense maximale par groupe. Cela requiert des opérateurs comme GROUP BY qui n'existent pas en algèbre relationnelle.
Calcul tuple : Impossible en calcul relationnel tuple, car ce formalisme est basé sur la théorie des ensembles et n'inclut ni ordre, ni limite.

Le joueur ayant le plus de personnages de la même classe

SQL :

```
SELECT p.Name, c.Class, COUNT(*) AS nbCharacters
FROM Players p
JOIN Characters c ON p.ID = c.PlayerID
GROUP BY p.Name, c.Class
ORDER BY nbCharacters DESC LIMIT 1;
```

Algèbre relationnelle et Calcul tuple : Impossible car cela nécessite de compter le nombre de personnages que chaque joueur possède avec COUNT(*) en SQL, ce qui est impossible pour l'algèbre relationnelle et le calcul tuple.

La quête ayant la plus grosse récompense en or par niveau de difficulté

SQL :

```
WITH QuestGold AS (
SELECT q.Name AS QuestName, q.Difficulty, SUM(i.Price * r.Quantity)
AS TotalGold
FROM Quests q
JOIN Rewards r ON q.Name = r.QuestName
JOIN Items i ON r.ItemName = i.Name
GROUP BY q.Name, q.Difficulty
),
RankedQuests AS (
    SELECT *, ROW_NUMBER() OVER (PARTITION BY Difficulty ORDER BY
    TotalGold DESC) AS `rank`
    FROM QuestGold
)
SELECT QuestName, Difficulty, TotalGold
FROM RankedQuests
WHERE `rank` = 1;
```

Algèbre relationnelle et Calcul tuple : Impossible car cela nécessite de faire la somme (avec SUM en SQL) du prix des items * la quantité de ce qu'offre la quête, ce qui n'existe pas en algèbre relationnelle et en calcul tuple pour ce formalisme.

Le PNJ possédant l'inventaire contenant les objets dont la valeur en or cumulée est la plus importante

SQL :

```
SELECT npc.Name, SUM(i.Price * npcII.Quantity) AS TotalValue
FROM NPCs npc
JOIN NPCItemInventories npcII ON npc.Name = npcII.NPCName
JOIN Items i ON npcII.ItemName = i.Name
GROUP BY npc.Name
ORDER BY TotalValue DESC LIMIT 1;
```

Algèbre relationnelle et Calcul tuple : Impossible car cela nécessite de faire la somme (avec SUM en SQL) du prix des items * la quantité des items que possède le PNJ dans son inventaire, ce qui n'existe pas en algèbre relationnelle et en calcul tuple pour ce formalisme.

Le type d'objet (arme, armure, potion ou relique) le plus souvent offert en récompense de quêtes de niveau 5

SQL :

```
SELECT item.Type, COUNT(*) as apparitions
FROM Items item
WHERE item.Name in (
    SELECT reward.ItemName
    FROM Rewards reward
    WHERE reward.ItemName <> 'Or' AND
        reward.QuestName in (
        SELECT quest.Name
        FROM Quests quest
        WHERE quest.Difficulty = 5
    )
)
GROUP BY item.Type
ORDER BY apparitions DESC LIMIT 1;
```

Algèbre relationnelle et Calcul tuple : Impossible car cela nécessite de compter (avec COUNT en SQL) le nombre d'apparition des objets, ce qui n'existe pas en algèbre relationnelle et en calcul tuple pour ce formalisme.

Les monstres avec les meilleures récompenses en valeur en or cumulée en fonction de leurs points de vie

SQL :

```
SELECT (MonsterCost/MonsterHealth) AS Ratio, Name
FROM (
    SELECT SUM(ItemPrice * lootQuantity) as MonsterCost, MonsterName
    FROM
    (
        SELECT
            items.Name as ItemName,
            items.Price as ItemPrice,
            loot.MonsterName as MonsterName,
            loot.Quantity as lootQuantity
        FROM Items items
        JOIN MonsterLoots loot ON items.Name = loot.LootName
    ) as subrequest
    GROUP BY MonsterName
) as monsterHealthAndPrice
JOIN Monsters monsters on monsters.Name = MonsterName
ORDER BY Ratio DESC;
```

Algèbre relationnelle et Calcul tuple : Impossible car cela nécessite de faire la somme (avec SUM en SQL) du prix des items * la quantité du loot, ce qui n'existe pas en algèbre relationnelle et en calcul tuple pour ce formalisme. De plus, il faut trier le résultat du meilleur ratio par ordre décroissant, ce qui est également impossible en algèbre relationnelle et en calcul tuple.