420-KBA, TP no2

1. **Pondération : 15%.**
2. **Date de de remise : le 11 octobre minuit, sauf pour les modèles de données.**
3. **Modalité de remise : boite de remise.**
4. **Ce travail doit se faire en individuel ; cependant vous pouvez vous entraider.**
5. **Vous pouvez poser des questions cependant je ne ferais pas le travail à votre place.**

Objectifs

Ce travail vise l'atteinte des objectifs suivants :

1. Produire un modèle conceptuel de données
2. Produire un modèle relationnel de données.
3. Écrire des procédures stockées
4. Écrire des triggers.

Mise en contexte :

Nous sommes dans un environnement du jeu vidéo de type médiéval, et nous souhaitons développer une application qui permet la vente en ligne d'un certain nombre d'items concernant le jeu.

Les items offerts sont des Armes, des Armures et des Potions. Tous les items ont en commun un numéro **unique**, un nom, une quantité en inventaire, un prix unitaire, une quantité minimale en inventaire et un flag de disponibilité. Ce flag est tout le temps égal à 1 aussi longtemps que l'item n'est pas supprimé. Sinon le flag est égal à 0.

Les armes ont une efficacité (nombre de points de dommage), un genre (une main ou deux mains ou autre) et une description

Les armures ont une matière qui la compose, un poids et une taille.

Les potions ont un effet attendu et une durée pour l'effet.

Dans l'environnement du jeu, il y a plusieurs joueurs, chaque joueur peut acheter plusieurs items. Des joueurs différents peuvent acheter des items identiques.

Un joueur a un numéro **unique**, un alias **unique**, un nom, un prénom et un montant initial en écus.

Tous les joueurs peuvent acheter n'importe quel item, à condition qu'il soit disponible en inventaire. La quantité d'un Item achetée peut-être supérieure à 1.

Les achats sont conservés dans un panier. Le panier d'achats contient la quantité achetée de chaque item pour un joueur donné. Au fur et à mesure que les items sont ajoutés dans le panier, l'inventaire de l'item est mis à jour. Les items s'accumulent dans le panier jusqu'au moment où le joueur passe à la caisse pour payer le panier. Au moment de payer le panier le solde en écu du joueur est déduit du montant total des achats. Après que le joueur ait payé le conteneur du panier, le panier est supprimé.

Votre base de données doit-être conçue de sorte que l'on puisse conserver l'historique des achats des joueurs. Cet historique contient le contenu des paniers d'achats payés. On a pour chaque historique d'achats un numéro unique, la date à laquelle les achats ont été payés ainsi que la liste des items achetés .

C'est au moment de payer le panier que l'historique des achats est mis à jour avec le contenu du panier.

Questions :

Conception de la base de données :

1. Donner le modèle conceptuel et le modèle relationnel du jeu. Date de remise de cette partie **: le mardi 24 septembre à la fin de chaque séance du groupe concerné.**
2. Créer la base de données : BDItemsVosInitiales : Exemple BDItemsSY.
3. Créer toutes les tables de la base de données.
4. Peupler votre base de données avec des données initiales.

Exploitation de la base de données : Procédures stockées et triggers.

Groupe 1 :

Écrire les procédures stockées suivantes pour la gestion des Items.
(Toutes les procédures et fonctions doivent-être exécutées et testées)

1. Une procédure ***ajouterArme***, cette procédure permet d'ajouter une arme dans la base de données. Insérer toutes les informations d'une arme.

2. Une procédure ***ajouterArmure***, cette procédure permet d'ajouter une armure dans la base de données. Insérer toutes les informations d'une armure.

3. Une procédure ***ajouterPotion***, cette procédure permet d'ajouter une Potion dans la base de données. Insérer toutes les informations d'une armure.

4. Une procédure ***afficherItems*** cette procédure permet d'afficher les items selon le **type de l'item**. Afficher toutes les informations des items. Exemple pour une arme, nous allons afficher les informations (nom, quantité en stock, le prix unitaire, le nombre de points de dommage et le genre). Pour cette question, il est conseillé de créer des **vues** que vous allez utiliser dans votre procédure. Le code sera plus propre.

5. Une fonction ***prixMoyenItems*** qui retourne sous forme de table le prix moyen de chaque type d'item.

6. Une procédure ***supprimerItem,*** qui permet de supprimer un item étant donné un numéro d'item. La suppression d'un item implique la mise à jour du flag de disponibilité à 0. Un item supprimé qui est contenu dans des paniers doit aussi être supprimé.

Groupe 2

Écrire les procédures stockées suivantes pour la gestion des Achats.
(Toutes les procédures et fonctions doivent-être exécutées et testées)

1. Une fonction **montantPanier** qui retourne le montant total du panier d'achats d'un joueur étant donné son **alias**. S'il n'y a pas de panier pour le joueur, la fonction retourne 0;

2. Une procédure ***ajouterItemPanier*** qui permet d'ajouter un item au panier avec une quantité donnée pour un joueur étant donné son **alias**.

    - Si la quantité en inventaire de l'item est insuffisante, l'ajout est annulé.
    - Si le solde du joueur est insuffisant pour couvrir le montant total du panier plus l'ajout de l'item, l'ajout est annulé.
    - La procédure ajoute l'item au panier et prend soin de mettre à jour l'inventaire de l'item.
    - Si l'item existe déjà dans le panier du joueur, la quantité est mise à jour uniquement.

3. Une procédure ***modifierItemPanier*** qui permet de modifier la quantité achetée d'un item dans le panier d'achats d'un joueur étant donné son **alias**.

    - Si la quantité en inventaire est insuffisante pour couvrir la nouvelle quantité, en tenant compte du retour, la modification est annulée;
    - La procédure met à jour la quantité et prend soin de mettre à jour l'inventaire de l'item.

4. Une procédure ***supprimerItemPanier*** qui permet de supprimer un item du panier d'achats d'un joueur étant donné son **alias**.

    - La procédure supprime l'item du panier et prend soin de mettre à jour l'inventaire de l'item.

5. Une fonction **table *afficherPanier*** qui retourne le contenu du panier d'un joueur. La fonction retourne sous forme d'une table le nom de l'item et son type (**'Arme', 'Armure', 'Potion'**), la quantité et le prix total.

6. Une procédure **p*ayerPanier*** qui permet de payer le panier d'achats d'un joueur étant donné son **alias**.

    - Mettre à jour le solde du joueur avec le montant du panier.
    - **<u>Optionnellement</u> (bonus)**
        i. Mettre à jour l'historique des achats du joueur.
    - Supprimer le panier d'achats du joueur;

7. Une procédure ***SupprimerPanier*** qui permet de supprimer le panier d'achats d'un joueur étant donné son **alias**;

    - Remettre en inventaire la quantité achetée de chaque item contenue dans le panier du joueur;
    - Une fois l'inventaire mis à jour, on supprime le panier d'achats du joueur;

<u>Optionnellement (bonus)</u>

8. Une fonction **table *afficherAchats*** qui retourne la liste de tous les achats payés étant donné **l'alias** d'un joueur. La table retournée contient l'alias du joueur, le nom de l'item, son type ('Arme', 'Armure', 'Potion') et la quantité.

Groupe 3

1. Le trigger ***beforeInsertArmure*** qui permet de garantir qu'un numéro inséré dans la table Armures ne sera pas utilisé (inséré) dans la table Armes ni la table potions.

2. Un trigger ***beforeInsertArme*** qui permet de garantir qu'un numéro inséré dans la table Armes ne sera pas utilisé (inséré) dans la table Armures ni la table potions.

3. Un trigger ***updateStockItem*** qui permet d'augmenter la quantité en stock d'un Item lorsque la quantité limite est atteinte. Ce qui veut dire si la quantité en stock ne doit pas être plus petite que la quantité limite. On doit augmenter la quantité du triple de la quantité limite.

4. Le trigger ***cascadeDeleteItem*** qui fait la suppression en cascade d'un item qui n'est pas dans l'historique des achats. Les items supprimés qui sont contenus dans des paniers doivent aussi être supprimés.

5. Tester TOUS VOS triggers par les requêtes DML appropriées.

Ce qu'il faut remettre <u>dans un dossier Zippé portant votre nom</u> :

1. Les modèles de la base de données. La BD doit être normalisée. Ces modèles seront remis au format pdf;
2. Un script SQL contenant la création des tables;
3. Un script SQL contenant les procédures et les fonctions stockées;
4. Un script SQL contenant TOUS les triggers;
5. Un script SQL contenant les requêtes d'exécution des procédures, des fonctions et de la vérification du déclenchement des triggers;

Évaluation

Élément évalués	Pondération
Groupe 1 (3*6)	18
Groupe 2, toutes les questions sur 4 sauf Q2 qui vaut 6	30
Groupe 3, chaque trigger sur 4 points	16
Modèles de données	12
Script d'Exécution	10
Création des tables avec les contraintes	10
La base de données est bien peuplée	4
Total	100

Bonus : 5 points :