[1]Travail pratique 1

Relix, pour des reliques eXtra

Automne 2023

- Ce travail sera réalisé individuellement.
- À remettre **le 17 octobre** 2023
- Le modèle relationnel doit être remis le 28 septembre.au format pdf et rien d'autre.
- Le travail compte pour 20 % de la note finale.

Attention !!

Pour ce travail, vous devez utiliser uniquement les concepts vus en classe.
Si vous avez besoin d'utiliser autre que les concepts présentés en classe alors vous devez :

1. Expliquer le concept avant son utilisation
2. Donner la source de l'information
3. Demander l'autorisation à l'enseignante.

Une évaluation orale pourrait se faire pour le travail remis pour n'importe quelle question et n'importe quel étudiant.

La débrouillardise et l'autonomie sont deux qualités recherchées chez un informaticien. **Le plagiat sous toutes ses formes c'est NON et entraine les sanctions prévues par la PIEA**

Objectifs

Ce travail vise l'atteinte des objectifs suivants :

1. Produire un modèle de données normalisé
2. Écrire des procédures stockées
3. Écrire des triggers.

[1] Source de l'image : https://hitek.fr/42/reliques-artefacts-civilisation-photos-photographies-armures-casques-epee-histoire-_11350

Mise en contexte :

Nous sommes dans un environnement du jeu vidéo de type médiéval, et nous souhaitons développer une application ***Relix*** où chaque joueur pourrait acheter des objets appelés également Reliques afin équiper son personnage.

Les reliques :

- Les reliques peuvent être des armures, des talismans ou des parchemins de sorts
- Toutes les Reliques sont identifiées par un **numéro unique (IDENTITY)**, ont un nom, un prix unitaire, un flag de disponibilité , une quantité en inventaire et une quantité minimale. Le flag est tout le temps à 1 aussi longtemps que la relique n'est pas supprimée. Sinon le flag est égal à 0.
- Les armures ont en plus une taille, le poids, et la matière qui la compose : Cuire, métal…
- Les talismans ont en plus les métaux dont ils sont faits (le métal qui compose le talisman: ivoire, or, argent, bronze etc), le type (bague, bracelet anneau, chaîne etc) et un pouvoir magique.
- Les parchemins ont en plus un sort, la durée du sort et l'antidote pour annuler l'effet du sort.

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

Les joueurs :

- Un joueur a un numéro **unique (IDENTITY)**, un alias **unique**, un nom, un prénom, un montant initial en écus et un niveau.
- Les niveaux sont : 1,2 , 3 , 4 ou 5.
- Les joueurs de niveau 1 ne peuvent pas acheter des parchemins .

Les achats :

Tous les joueurs peuvent acheter n'importe quelle relique, à condition qu'elle soit disponible en inventaire. La quantité d'une Relique achetée peut-être supérieure à 1.

Les achats sont conservés dans un panier. Le panier d'achats contient la quantité achetée de chaque Relique pour un joueur donné. Au fur et à mesure que les Reliques sont ajoutées dans le panier, l'inventaire de l' Relique est mis à jour. Les Reliques 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 contenu 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 Reliques achetées avec les quantités achetées . C'est au moment de payer le panier que l'historique des achats est mis à jour avec le contenu du panier.

Questions :

Partie , Conception de la base de données :

1. Donner le modèle relationnel de la base de données Relix. La remise est pour au plus tard **le jeudi 28 septembre minuit** au format pdf ou image et rien d'autre. Cette date est pour tous les groupes

Partie 2, Exploitation de la base de données : Procédures stockées et triggers.

Groupe 1, Initialisation de la base de données :

1. Créer la base de données : BDRelix:
2. Créer toutes les tables de la base de données en exécutant le script : **createTables.sql. Vous êtes obligés d'utiliser le script pour créer les tables afin de faciliter le débogage et la correction**
3. Exécuter le script **insertJoueur.sql** pour insérer les joueurs dans votre base de données.

Groupe 2, Gestion des Reliques. :

Écrire les procédures stockées suivantes pour la gestion des Reliques . Toutes les procédures et fonctions doivent-être exécutées et testées. Vos données doivent être cohérentes en tout temps→**Transactions**

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

2. Une procédure ***ajouterTalisman*** cette procédure permet d'ajouter un bijou dans la base de données. Insérer toutes les informations d'un bijou.

3. Une procédure ***ajouterParchemin***, cette procédure permet d'ajouter un parchemin dans la base de données. Insérer toutes les informations du parchemin

4. Donner le script d'exécution des procédures précédentes pour que le contenu des tables soit : (voir exemple plus loin)
   Pour tester vos procédures il faut vous référer aux 4 tableaux suivants.

Table Reliques

numRelique	nom	qtInventaire	prix	qtMin	typeRelique	flagDispo
1	Veste Elfique	50	300	5	A	1
2	La cape du roi	20	350	3	A	1
3	Casque de fer	20	80	5	A	1
4	Armure de Dragonnier	5	100	1	A	1
5	Anneau rouge feu	16	420	3	T	1
6	Le bracelet Holow	10	420	2	T	1
7	La pierre de Willow	20	500	4	T	1
8	La chaine de la sorcière	10	200	2	T	1
9	Les gants magiques	15	150	1	A	1
10	Le Boudin	10	200	2	P	1
11	L'aveugle	8	250	2	P	1
12	Fatiguant	12	250	1	P	1

Table Armures :

numRelique	matiere	poids	taille
1	Mithril	2	4
2	Mithril	1	7
3	Fer	50	8
4	Peau de dragon	30	12
9	Peau de dragon	12	6

Table Talismans :

numRelique	matiere	typeTalisman	pouvoir
5	or du dragon	bague	control over mordor
6	Or argent et bronze	bague	rend invisible
7	pierre magique	pierre	Contrôle la foudre
8	argent	chaine	etouffe l'ennemi

Table Parchemins

numRelique	sort	duree	antidote
10	Ne pas parler	20	Menthe fraiche
11	Ne pas voir	10	Citron des neiges
12	Ne pas dormir	5	Camomile des rois

Exemple, exécution de la procédure : `ajouterParchemin`

```
execute ajouterParchemin
@nom = 'Fatiguant',
@qtInv = 12,
@prix = 250,
@qtMin = 1,
@sort='Ne pas dormir',
@duree = 5,
@antidote ='Camomile des rois';
```

5. Une procédure ***afficherRelique*** *(*@typeRelique) cette procédure permet d'afficher les Reliques selon le **type de la relique**. Afficher toutes les informations des reliques. Exemple pour une armure, nous allons afficher les informations (nom, quantité en stock, le prix unitaire, le poids, la taille, matériel). Pour cette question, vous devez créer des **vues** que vous allez utiliser dans votre procédure. Le code sera plus propre. **Les vues sont créées à l'extérieur des procédures**

6. Une fonction ***prixMoyenRelique*** qui retourne sous forme **de table** le prix moyen de chaque type de relique. (voir résultat)

typeRelique	moyennePrix
Armures	196
Parchemins	233
Talisman	385

7. Une procédure ***supprimerRelique*** *(*@numRelique*),* qui permet de supprimer une relique étant donné un numéro d'une relique. La suppression d'une relique implique la mise à jour du flag de disponibilité à 0. Une relique supprimée qui est contenue dans des paniers doit aussi être supprimée. **Exécutez cette procédure pour reliques dont les numRelique sont 4 et 8**

Groupe 3, Gestion du panier d'achat

Écrire les procédures stockées suivantes pour la gestion des Achats.

Toutes les procédures et fonctions doivent-être exécutées et testées. Pour cette partie, les données de test vous seront fournies au moment opportun.

1. Une fonction **montantPanier** (@alias) 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;

   Pour tester faire les insertions suivantes dans la table Paniers.

```
insert into Paniers(numJoueur,numRelique,qtRelique) values(3,1,2);
insert into Paniers (numJoueur,numRelique,qtRelique) values(3,3,1);
insert into Paniers (numJoueur,numRelique,qtRelique) values(6,6,1);
```

   Tester votre fonction, puis vider le panier en exécutant :

```
delete from Paniers;
```
(il faut vider le panier c'est une obligation)

2. Une procédure ***ajouterReliquePanier*** (@alias, @numRelique, @qtRelique) qui permet d'ajouter une Relique au panier avec une quantité donnée pour un joueur étant donné son **alias**.

   Les paramètres de la procédure sont donc : L'alias du joueur, le numéro de la relique et la quantité à acheter.

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

3. Une procédure ***modifierReliquePanier*** (@numRelique, @nouvelleQtRelique, @alias) qui permet de modifier la quantité achetée d'une relique dans le panier d'achats d'un joueur étant donné son **alias**.
    - Si la quantité en inventaire est insuffisante pour couvrir la nouvelle quantité, la modification est annulée;
    - La procédure met à jour la quantité et prend soin de mettre à jour l'inventaire de la relique
    - Si le solde du joueur est insuffisant pour couvrir le montant total du panier plus l'ajout de la relique, l'ajout est annulé
4. Une procédure ***supprimerReliquePanier*** (@numRelique, @alias) qui permet de supprimer une relique du panier d'achats d'un joueur étant donné son **alias**.
    - La procédure supprime la relique du panier et prend soin de mettre à jour l'inventaire de la relique
5. Une fonction **table *afficherPanier*** (@alias) qui retourne le contenu du panier d'un joueur. La fonction retourne sous forme d'une table le nom de la relique et son type (**'Armure', 'Talisman', 'Parchemin**), la quantité et le prix total.
6. Une procédure **p*ayerPanier*** (@alias) 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.
    - Mettre à jour l'historique des achats du joueur.
    - Supprimer le panier d'achats du joueur;
7. Une procédure ***SupprimerPanier*** (@alias) qui permet de supprimer le panier d'achats d'un joueur étant donné son **alias**;
    - Remettre en inventaire la quantité achetée de chaque relique contenue dans le panier du joueur;
    - Une fois l'inventaire mis à jour, on supprime le panier d'achats du joueur;
8. Une fonction **table *afficherAchats*** (@alias) 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 la relique, son type ('Armure', 'Parchemun', 'Talisaman') et la quantité.

9. Groupe 4, Les triggers pour l'intégrité des données

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

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

3. Écrire le trigger ***afterInsertPanier*** qui garantit que les joueurs de niveau 1 ne peuvent pas ajouter dans le panier des Items qui sont des parchemins.

4. Le trigger ***cascadeDeleteRelique*** qui fait la suppression en cascade d'une relique qui n'est pas dans l'historique des achats. Les reliques supprimées qui sont contenus dans des paniers doivent aussi être supprimés.

5. 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. ***(optionnellement)***

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

Ce qu'il faut remettre dans un dossier Zippé portant votre nom :

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

Évaluation

Élément évalués	**Pondération**
Modèle de données	15
Groupe 2 : Q1, Q2, Q3, Q5, Q6, Q7 sur 3 points chaque. Q4, 2 points	20
Groupe 3 : Q2, Q3, Q6, Q7 sur 6 les autres sur 4 : (4*6) + (4*4)	40
Groupe 4, chaque trigger sur 3 points	12
Script d'exécution	13
Total	100