



Travail pratique 1, Automne 2026 *Knapsack, ... le sac à dos de survie.* ¹

- Ce travail sera réalisé individuellement et compte pour 20 % de la note finale.
- À remettre :
 - Le modèle de données : jeudi 01 octobre avant minuit
 - Les questions des groupe 2 et 3 vendredi 09 octobre avant minuit
 - Les questions du groupe 4 le vendredi 23 octobre 2026

Attention !!

Pour ce travail, vous devez utiliser uniquement les concepts vus en classe ou dans le cours de KB6 (Introduction aux bases de données)

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 ou à l'enseignant

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 entraîne les sanctions prévues par la PIEA**

Objectifs

1. Produire un modèle de données normalisé
2. Écrire des procédures stockées (Procédures, fonctions et triggers)
3. Veiller à la cohérence des données par les transactions
4. Utiliser les curseurs.
5. Développer son esprit d'analyse et son autonomie.

Préambule :

En algorithmique, le problème du sac à dos, parfois noté (KP) (de l'anglais **Knapsack**

Problem) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids.

(https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos)

¹ Source de l'image : <https://www.flickr.com/photos/playstationblog/3715830608/>

Mise en situation

Nous sommes dans un environnement de jeu vidéo de type **Fallout**, et nous souhaitons développer une application qui permet l'achat d'un certain nombre d'items concernant le jeu Fallout. L'achat d'item utilise un panier d'achat.

Items du jeu :

1. Un Item du jeu a un numéro unique, un nom, une quantité en stock, un type d'item (armes, armures, médicaments, nourritures), un prix unitaire , **un poids** (en livres) et un flag de disponibilité.
2. Les armes ont une portée, les dégâts infligés et une force requise pour la manipuler.
3. Les armures ont une matière qui compose l'armure et la taille : exemple des gants, casques, etc.)
4. Les armures ont un degré (un pourcentage) de résistance faces aux dégâts. Chaque armure peut avoir un seuil de résistance face au feu, au plasma, au laser, aux explosions etc.
5. La nourriture(Fourmis rôties, faux-filet, cuisse de poulet, bourbon, eau, etc.). qui redonne des points de vie et de la dextérité, ont un apport caloriques et d'autres informations.
6. Les médicaments (stimpack, Rad-x, sérum mystérieux etc...) qui redonne des points de vie et de la dextérité et qui ont un effet, la durée de l'effet et l'effet indésirable.

Les joueurs :

1. Un joueur a un alias unique, un nom, un prénom et **un montant initial en caps** (capsules), une dextérité (en points), le nombre de points de vie et un poids maximal à transporter . Les joueurs sont identifiés par un numéro séquentiel. **(IDENTITY)**
2. Tous les joueurs peuvent acheter n'importe quel item.
3. La quantité d'un item acheté peut être supérieur ou égal 1
4. Un joueur ne peut pas transporter plus qu'un certain poids sinon il perd de la dextérité.
5. La dextérité ne doit jamais être en dessous d'une certaine valeur.
6. Régulièrement le joueur doit se nourrir et prendre des médicaments pour que le nombre de points de vie ne baisse pas.
7. L'inventaire du joueur est transporté dans un sac à dos.
8. Initialement tous les joueurs ont un nombre fixe de capsules (caps) qu'ils peuvent utiliser pour acheter des Items.

Les achats :

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, le stock 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 caps du joueur est déduit du montant total des achats et la dextérité du joueur est mise à jour. Après que le joueur ai payé le contenu du panier, le panier est supprimé.

Votre base de données doit-être conçue de sorte que l'on puisse **conserver l'inventaire des joueurs dans un sac à doc**. L' inventaire contient le contenu des paniers d'achats payés.

C'est au moment de payer le panier que l'inventaire des achats (le sac à dos) est mis à jour avec le contenu du panier.

Le Sac à dos :

Le sac à dos du joueur représente l'inventaire du joueur. Le poids du sac à dos est important pour la dextérité du joueur.

Puisque le joueur transporte son sac à dos, voici la règle qui s'applique.

- Si le poids du sac à dos est inférieur ou égal au poids maximum à transporter, le joueur garde ses points de dextérité.
- Pour chaque unité de dépassement de poids, le joueur perd un point de dextérité.
- Le poids dépassé ne peut être plus grand que la moitié du poids max sinon le joueur meurt.

Exemple :

PoidsMax du joueur	Poids du Sac à do	Dextérité
50	45	50 reste 50
50	55	50 devient 45 (50-5)
50	75 =(50 +50/2)	50 devient 25 (50-25)
50	76 →Impossible	

Questions :

Partie1, Conception de la base de données :

1. Donner le modèle relationnel de la base de données **DbKnapsak**. La remise est pour au plus tard **jeudi 01 octobre 23h59** 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 : **DbKnapsak**. ;
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.

Questions du groupe 2, Les triggers pour l'intégrité de la base de données, **date de remise vendredi 09 octobre**

1. Écrire le trigger **CTRLPrixArme** qui garantit que le prix des armes est plus élevé que le prix des autres type d'items
2. Écrire le trigger **CTRLPtdeVie** qui garantit le nombre de point de vie d'un médicament est plus élevé que le nombre de pts de vie d'une nourriture
3. Le trigger **CTRLInsertArme** qui permet de garantir qu'un numéro inséré dans la table Armes ne sera pas utilisé (inséré) dans la table Armure ni la table médicaments, ni la table nourritures
4. Un trigger **CTRLInsertArmure** qui permet de garantir qu'un numéro inséré dans la Armures ni pas dans Armes, ni Nourritures ni Médicament
5. Tester TOUS VOS triggers par les requêtes DML appropriées.

Questions du groupe 3, Gestion des Items ,date de remise vendredi 09 octobre
Écrire les procédures stockées suivantes pour la gestion des Items.

Toutes les procédures et fonctions doivent-être exécutées et testées. Pour tester vos procédures il faut vous référer aux 5 tableaux suivants.

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 données d'une armure.
3. Une procédure **ajouterNourriture**, cette procédure permet d'ajouter une nourriture dans la base de données. Insérer toutes les informations d'une nourriture.
4. Une procédure **ajouterMedicament**, cette procédure permet d'ajouter un médicament dans la base de données. Insérer toutes les informations de d'un médicament
5. Donner le script d'exécution des procédures d'insertion pour que le contenu de vos tables soit au minimum le suivant : (voir exemple d'exécution plus bas). Commencez par donner les cas qui marchent pour toutes les procédures afin de ne pas briser la séquence de idItem.

Table Item :

idItem	nom	qtStock	prix	typeltem	flagDispo	poids
1	Agile du desert	80	210	A	1	5
2	Carabine	10	210	A	1	5
3	Fusil d'assaut	10	300	A	1	15
4	Arme Red Ryder	5	1200	A	1	5
5	Lance Flamme	5	800	A	1	15
6	Arme Alien	2	800	A	0	5
7	Armure de combat	20	60	R	1	10
8	Veste en cuire	5	10	R	1	3
9	Super Armure	2	150	R	1	15
10	Super Armure renforcée	4	200	R	1	15
11	Viande de cheval	10	10	N	1	1
12	Tas de fourmies	100	20	N	1	2
13	viande de mouches	100	20	N	1	1
14	Macaroni	50	20	N	1	3
15	Radix	30	40	M	1	1
16	Stimpak	200	70	M	1	4
17	Serum	12	60	M	1	1
18	Vim	5	50	M	1	2
19	Pâtée pour chien	30	6	N	1	2
20	Viande de Lophius	20	30	N	1	3

Table Armes

idItem	portee	degat	forceRequise
1	25	15	4
2	40	20	5
3	45	16	5
4	32	25	5
5	5	100	6
6	10	100	2

Table Armures :

idItem	matiere	taille
7	Fer	8
8	Cuire	12
9	Cuivre	10
10	Cuivre et cuire	10

Table nourritures :

idItem	calorie	nbDex	nbPtvie
11	200	5	2
12	500	15	8
13	50	2	2
14	70	3	2
19	30	3	10
20	200	8	5

Table Médicaments :

idItem	effet	dureEffet	indesirable	nbDex	nbPtvie
15	enlève les radiations	24	Vomissement	10	40
16	soigne les seignements	100	Vision flou	5	45
17	se régénérer	100	aucun	2	40
18	reposant	24	aucun	5	20

Exemple d'exécution pour nourriture :

```
execute ajouterNourriture
@nom = 'Pâtée pour chien',
@qtInv = 30,
@prix = 6,
@poids=2,
@nbDex =3,
@nbPtvie =10,
@calorie = 30;
```

Exécutez le script DetailArmure.sql pour compléter les insertions dans les tables :ResistanceArmures et TypeDegats;

- 6- Une procédure **afficherItems** (@typeltem) 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 la portée, le nombre de point de dégâts et la force requise). Pour une armure afficher nom, quantité en stock, le prix unitaire, la matière et la résistance aux dégâts (afficher le nom du dégât au lieu du code).
Pour cette question, il est conseillé de créer des **vues** que vous allez utiliser dans votre procédure. Le code sera plus propre. **Les vues sont créées à l'extérieur des procédures**
- 7- Une fonction Table **prixMoyen** qui retourne pour chaque type d'item le prix moyen de l'item. vous devez avoir ce résultat :

typeltem	moyennePrix
Armes	586
Armures	105
Medicaments	55
Nourritures	17

- 8- Écrire une fonction table **meilleureResistance**(@nomDegat) qui permet de retourner sous forme de table la liste des armures ayant une meilleur résistance à un dégât. Afficher le nom de l'armure, le nom du dégât et le degrés de résistance. Exemple lorsque la valeur du paramètre est **Laser** cette fonction contient :

nom	nomDegat	degre
Veste en cuire	Laser	30
Super Armure	Laser	30

Questions du groupe 4 Gestion du panier , **date de remise vendredi 23 octobre**
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.

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 (idJoueur, idItem, qtItemPanier) values(100, 1, 2);  
insert into Paniers (idJoueur, idItem, qtItemPanier) values(100, 8, 5);  
insert into aniers (idJoueur, idItem, qtItemPanier) values(100, 11, 2);
```

Pour les insertions précédente, la fonction devrait retourner : **490** pour l'alias '**Maximus**'

2. Écrire une fonction **PoidsPanier**(@alias) qui retourne le poids 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 les données précédentes cette fonction retourne **27** pour l'alias '**Maximus**'
3. Écrire une fonction **PoidsSacAdos**(@alias) qui retourne le poids total du sac à dos d'un joueur étant donné son **alias**. S'il n'y a pas de sac à dos, la fonction retourne 0;

Pour tester votre fonction précédente, **PoidsSacAdos**(@alias), insérer les données suivantes :

```
insert into SacAdos (idJoueur, idItem, qtItem) values(100, 1, 2);  
insert into SacAdos (idJoueur, idItem, qtItem) values(100, 8, 5);  
insert into SacAdos (idJoueur, idItem, qtItem) values(100, 11, 2);  
insert into SacAdos (idJoueur, idItem, qtItem) values(100, 18, 2);  
insert into SacAdos (idJoueur, idItem, qtItem) values(100, 20, 2);
```

Attention !

Vider la table Paniers → **delete from** Paniers;
Vider la table SacAdos → **delete from** SacAdos;

4. Une procédure **ajouterItemPanier** (@alias, @idItem, @qtItem) qui permet d'ajouter un item 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 l'item et la quantité à acheter.

- Si la quantité en stock 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.
 - Si le poids du panier fera en sorte que le poids du sac à dos dépasse de la moitié du poids max alors l'ajout au panier est annulé. Ce que nous voulons dire est que le joueur pourrait excéder son poids mais pas plus que de moitié. Exemple un joueur ayant comme poids max 30 pourrait dépasser de 15 .
 - La procédure ajoute l'item au panier et prend soin de mettre à jour le stock de l'item.
 - Si l'item existe déjà dans le panier du joueur, la quantité est mise à jour uniquement.
5. Une procédure **supprimerItemPanier** (@idItem, @alias) 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 le stock de l'item.
6. 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 l'item et son type ('Arme', 'Armure', 'Nourriture', 'Medicament'), la quantité et le prix total par Item.
- Pour les questions 7 et 8 , l'utilisation de curseur est obligatoire**
7. Une procédure **payerPanier** (@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 la dextérité du joueur en tenant compte du dépassement du poids Max à transporter.
 - Mettre à jour le sac à dos des achats du joueur. (Précision : Si l'item est déjà dans le sac à dos du joueur, il faut mettre à jour la quantité, sinon il faut insérer l'item dans le sac à dos du joueur)
 - Supprimer le panier d'achats du joueur;
8. Une procédure **supprimerPanier** (@alias) qui permet de supprimer le panier d'achats (abandonner un achat) d'un joueur étant donné son **alias**;
- Remettre en stock la quantité achetée de chaque item contenue dans le panier du joueur;
 - Une fois le stock mis à jour, on supprime le panier d'achats du joueur;

9. Une procédure **mangerNourriture**(@alias, @idItem, @qtItem) qui permet à un joueur de consommer de la nourriture. La procédure doit mettre à jour la dextérité, le nombre de points de vie du joueur, la quantité de la nourriture dans le sac à dos
10. Une procédure **seSoigner** (@alias, @idItem, @qtItem) qui permet à un joueur de d'utiliser un médicament. La procédure doit mettre à jour la dextérité, le nombre de points de vie du joueur , la quantité des médicament dans le sac à dos
11. Une fonction **table afficherSacAdos**(@alias) qui retourne la liste de tous les achats payés (Sac à dos) étant donné l'**alias** d'un joueur. La table retournée contient l'alias du joueur, le nom de l'item, son type ('Arme', 'Armure', 'Nourriture', 'Medicament') et la quantité.
12. Le trigger **cascadeDeleteItem** qui fait la suppression en cascade d'un item qui n'est pas dans le sac à dos du joueur. Les items supprimés qui sont contenus dans des paniers doivent aussi être supprimés. Si l'item est dans le sac à dos du joueur alors le flag doit être à zéro dans la table Items.
13. Écrire le trigger **CTRLPrixItem** qui garantit que le prix d'un Item ne peut pas être révisé à la baisse.

Grille de correction

Élément évalués	Points	Total
Modèle de données	8	8
Questions du Groupe 2		14
1, 2, 3 incluant les test	4 points chaque question	12
4	2	2
Questions du Groupe 3		24
1 et 2 incluant les test	4 points chacune	8
3 et 4 incluant les test	2 points chacune	4
6, 7 et 8	4 points chacune	12
Questions du Groupe 4		54
1, 2 et 3 incluant les test	4 points chacune	12
4	6 points	6
5,6	4 points chacune	8
7 (payer panier)	6	6
8(supprimer panier	5	5
9,10,11,13	3 chacune	12
12	5	5