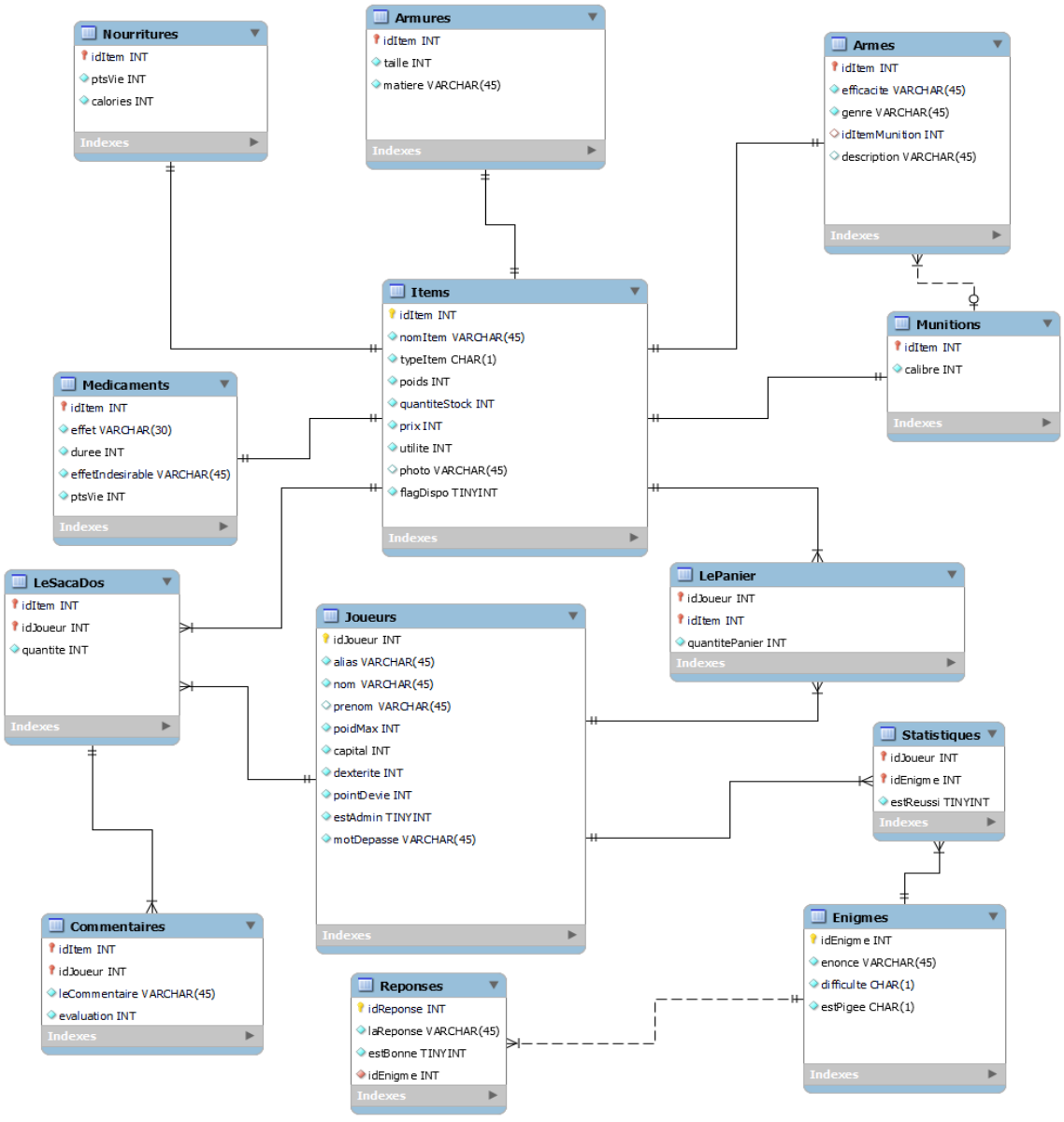
# Modèle suggéré pour la base de données : Knapsak

## Le modèle :

Le modèle suivant est une suggestion pour votre BD Knapsak. Vous n'êtes pas obligé de prendre le modèle suivant si votre base de données **est normalisée et validée.**



## Quelques détails :

1. Aucune contrainte CKECK n'est représentée dans le modèle. Il faut les ajouter manuellement : ALTER TABLE
    a. estAdmin prend la valeur 0 ou 1
    b. le capital initial (DEFAULT) il est égal à 1000.
    c. le typeItem : est CHAR (1).
        r→ pour armures,
        a→ pour armes,
        m→ pour medicaments
        n→ pour nourritures
        u→ pour munitions
    d. l'utilité d'un item prend les valeur 1, 2, 3, 4, et 5. Les médicaments et la nourriture ont l'utilité 1.
    e. Le nombre de points de vie initial d'un joueur est 10. Un maximum pourrait être fixé.
    f. Le nombre de points de vie d'une nourriture est entre 1 et 5.
    g. Le nombre de points de vie pour les médicament est entre 5 et 10.
    h. La dextérité initiale et maximale est 100.
    i. À l'insertion, le mot de passe doit être haché.

    j. Pour la table Items, vous pouvez ajouter un attribut : flagDispo. flagDispo est DEFAULT 1, a comme valeur 1 si l'item est disponible 0 s'il n'est pas disponible (on ne le vend plus). Ce qui n'a rien à voir avec la quantité. Un item peut avoir sa quantié 0 mais il reste disponible. Vous pouvez utiliser le flagDispo si vous voulez supprimer un item alors que celui-ci est dans LeSacaDos du joueur.
    k. Pour la table Enigmes, estPigee prend la valeur o (pour oui) ou n (pour non)
    l. difficulte prend les valeurs : d pour difficile, m pour moyen et f pour facile
    m. Dans Statistiques, estReussi prend la valeur 0 ou 1.
2. Pour autres les contraintes :
    a. La clé primaire est clairement visible en jaune
    b. La clé primaire qui est en même temps clé étrangère est une clé rouge
    c. Le losange bleu plein : NOT NULL (obligatoire)
    d. Le losange vide : optionnel.
3. Pour la table Paniers, il faudra la vider après chaque achat. Après la confirmation de l'achat d'un joueur, vous devez vider la table. Nous n'avons pas besoin de garder les données du panier. La table panier, est comme une table temporaire. D'ailleurs il est possible de gérer les achats sans avoir la table : **paniers**.
4. La table ressources n'est pas dans le modèle. Elle ne sera pas utilsée.

## Les tables, ce que le modèle ne montre pas :

| Tables | Les attributs PK | Les attribut FK |
|---|---|---|
| Items | idItem | Aucune |
| Armes | idItem | idItem (vient de la table Items)<br>idItemMunition vient de la table Munitions. |
| Armures | idItem | idItem (vient de la table Items) |
| Munitions | idItem | idItem (vient de la table Items) |
| Medicaments | idItem | idItem (vient de la table Items) |
| Nourritures | idItem | idItem (vient de la table Items) |
| Joueurs | idJoueur | aucune |
| LeSacaDos | idJoueur et idItem | idJoueur et idItem |
| Commentaires | idJoueur et idItem | idJoueur vient de LeSacaDos<br>idItem vient de la table LeSacaDos |
| Paniers | idJoueur et idItem | idJoueur et idItem |

Pour la table Commentaires le lien est avec LeSacaDos de cette façon on garantit qu'un joueur ne peut pas commenter un Items qu'il ne possède pas dans son inventaire.

On voit bien que la table Items est la table qui contient les informations communes à un Item. Les détails d'un Item sont stockés dans les tables qui correspondent au typeItem. Exemple les détails d'une Arme sont dans la table Armes.

Pour ajouter un Item, il faut insérer toutes les informations d'un Items, **dans ce cas une procédure stockées est recommandée :** (Exemples)

Procédure pour ajouter des munitions.

```
use dbsimba;

delimiter |
create procedure ajouterMunition(
    in pNom varchar(50),
    in pQuantite int,
    in pPrix int,
    in pPhoto varchar(100),
    in putilite int,
    in pcalibre int,
    in ppoids int)

begin
    declare ptypeItem char(1) default 'u';
    declare pidItem int;
        start transaction;
           insert into Items (nomItem, quantiteStock, prix, photo, typeItem, utilite, poids)
           values ( pNom, pQuantite, pPrix, pPhoto, ptypeItem, putilite, ppoids);
           select LAST_INSERT_ID() into pidItem;
           insert into Munitions (idItem, calibre)
           values (pidItem, pcalibre);
        commit;

end |
```

**Appel de la procédure**

```
call ajouterMunition('Une balle',21,10,'balle.jpg',5,32,12);
```

```
call ajouterMunition('Une autre',10,10,'balle.jpg',5,12,12);
```

Pour la procédure suivante, les munitions doivent exister...(principe de la FK)

```
use dbsimba;

drop procedure if exists ajouterArme;

delimiter |
create procedure ajouterArme(
    in pNom varchar(50),
    in pQuantite int,
    in pPrix int,
    in pPhoto varchar(100),
    in ppoids int,
    in putilite int,
    in pDescription varchar(500),
    in pEfficacite varchar(30),
    in pGenreArme varchar(45),
    in pidMunition int)

begin
    declare pTypeItem char(1) default 'a';
    declare pidItem int;
          start transaction;
               insert into Items (nomItem, quantiteStock, prix, photo,typeItem, utilite, poids)
               values ( pNom, pQuantite, pPrix, pPhoto, ptypeItem,putilite,ppoids);

               select LAST_INSERT_ID() into pidItem;

               insert into Armes (idItem, description,efficacite, genre,idItemMunition)
               values (pidItem, pdescription,pEfficacite, pGenreArme, pidMunition);
               commit;

end |
```

**Appel de la procédure :**

```
call ajouterArme('Hache',10,60,'hache.jpg',2,10,'description','efficace','deuxmain',1);
```

## Détails concernant les données de la base de données

- Le prix d'un item est de l'ordre de 1 à 100 caps . Vous n'avez pas besoin d'une contrainte CHECK. C'est à titre indicatif seulement.
- Le montant initial d'un joueur est : 1000 caps
- Le poids maximal à transporter est 50 livres
- Le poids des Items est entre 1 et 10 livres
- La dextérité initiale et maximale est de 100.
- Pour chaque livre supplémentaire qui dépasse le poids max à transporter, on perd un point dextérité.
- Intialement, le nombre de points de vie d'un joueur est 10. Maximun pourrait être fixé.
- Pour faire un DELETE ou un UPDATE sans que le WHERE porte sur la PRIMARY KEY utilisez : SET SQL_SAFE_UPDATES = 0;

| Stories techniques pour la BD | Détails |
|---|---|
| Créer et peupler la base de données | Vous devez vérifier que votre modèle de données répond, puis faire les opérations suivantes dans l'ordre.<br>1. Créer les tables de la base de données à partir de votre modèle<br>2. Ajouter toutes les contraintes check<br>3. Créer les triggers s'il y'a lieu<br>4. Créer les procédures stockées pour insérer des items. Ici les procédures stockées sont obligatoires pour garantir la cohérence des données.<br>5. Appel des procédures pour faire les insertions. Au moins 5 items de chaque type.<br>6. Vérifier que vos insertions sont OK<br><br>Pour le point 1, vous devez garder une copie de votre modèle au cas où vous aurez à recréer la BD.<br>Pour les points 2 à 6 vous devez garder un fichier SQL. |

Un trigger vous sera fourni pour contrôler la quantité des médicaments et de la nourriture dans le sac à  dos du joueur.