Rapport projet collections d'objets culturels

Hayk ZARIKIAN

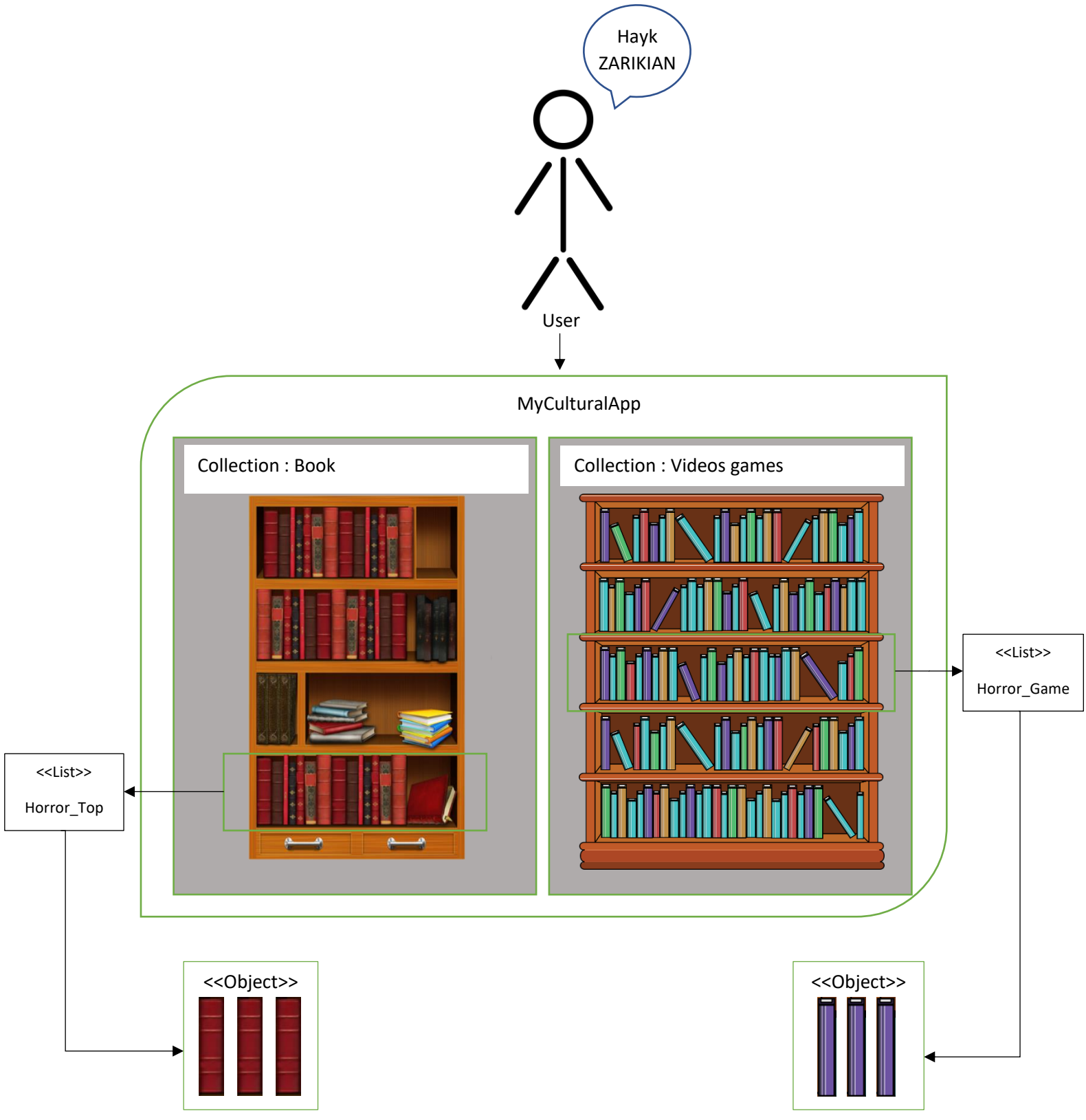

Table des matières

1 Conception de la base de données

1.1 Contexte et analyses des informations

Contexte :

Pour bien comprendre le sujet, il est important de concevoir un contexte réaliste afin de rendre la base de données fonctionnelle dans une conception d'application (Web, application, logiciel, …). Pour ce faire, j'ai choisi de me référer à une application mobile de gestion d'objets culturels.
L'objectif est d'avoir un profil utilisateur qui peut construire des collections puis des listes à l'intérieur de celle-ci afin d'injecter des objets culturels parmi celle disponible dans la bibliothèque générale (idéalement serveur web).

Analyse du sujet :

Les tables à créer

- **USERS** : Table stockant les utilisateurs.
- **COLLECTIONS** : Table stockant les collections associées à un utilisateur.
- **LISTS** : Table stockant les listes associées à une collection.
- **OBJECTS_REFERENCES** : Table stockant les références objets disponibles à injecter.
- **OBJECTS_USERS** : Table stockant les objets utilisateurs associées à une référence contenant dans une liste.

1.2 Modèle entité-association et contraintes d'intégrités / règles

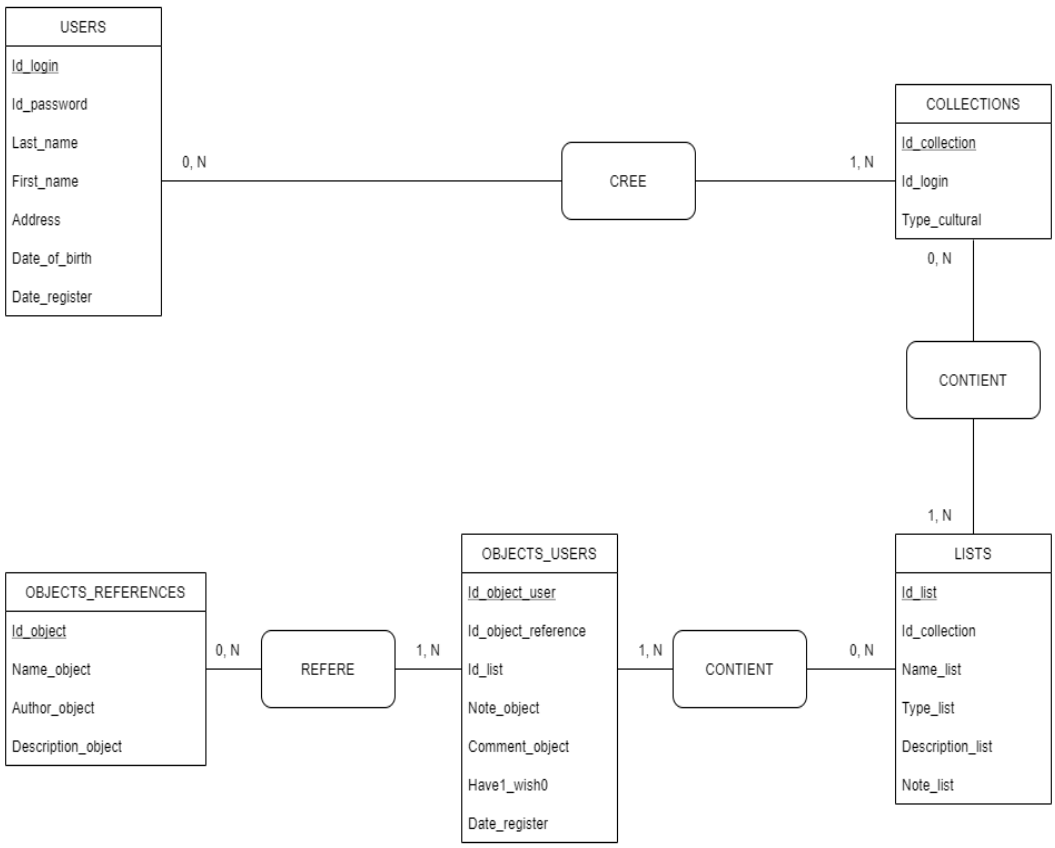

Figure 1 : Modèle Entité-Association

Explication du modèle :

Aucun ou plusieurs utilisateurs peuvent être inscrit dans la base de données. L'utilisateur peut créer aucune ou plusieurs collections puis créer aucune ou plusieurs listes associées à cette collection. Cependant l'utilisateur ne peut pas créer une liste sans créer de collection associée à celle-ci. Une liste contient aucun ou plusieurs objets utilisateurs mais un objet utilisateurs est obligatoirement dans une liste. Un objet utilisateur se réfère obligatoirement à un objet de référence. Il y a des objets de référence qui ne sont pas référés dans un objet d'utilisateur.

Une collection est associée à un utilisateur via « Id_login » de la table **COLLECTIONS**, une liste est associée à une collection via « Id_collection » de la table **LISTS**, un objet utilisateur possède un objet de référence via « Id_object_reference » de la table **OBJECTS_USERS** et possède l'identifiant de la liste via « Id_list » de la table **OBJECTS_USERS**. Un objet de référence est un objet disponible pour un objet utilisateur (médiathèque générale, idéalement serveur web). En résumé c'est grâce à l'attribut « Id_list » de la table **OBJECTS_USERS** qu'on détermine à qui appartient l'objet de référence.

Contraintes d'intégrités / règles :

Table **USERS** :

- L'identifiant de l'utilisateur (NOT NULL) en « **PRIMARY KEY** » est le login **UNIQUE**.
- Le login est composé de la première lettre du prénom et des sept premières lettres du nom en minuscule suivies de deux chiffres.
- Le mot de passe (NOT NULL) ne peut contenir que des lettres minuscules et majuscules, des chiffres et le caractère '_'.
- Le nom et le prénom doivent être « NOT NULL ».
- La date d'enregistrement (NOT NULL) est la date du jour de l'insertion de l'utilisateur dans la base.
- Idéalement hachage des informations confidentielles.

Table **COLLECTIONS** :

- L'identifiant de la collection (NOT NULL) en « **PRIMARY KEY** » est **UNIQUE** et s'auto-incrémente.
- L'identifiant de l'utilisateur en « **FOREIGN KEY** » est une clé étrangère qui fait référence à la table **USERS**. La collection est associée à l'identifiant d'un l'utilisateur.
- Le type de l'objet culturel doit être « NOT NULL ».
- Il n'est pas possible qu'un utilisateur ait deux collections de mêmes types.
- Un utilisateur peut avoir plusieurs collections.

Exemple :

Deux utilisateurs avec l'identifiant : hzarikia01 et hzarikia02 peuvent avoir la même collection cependant l'identifiant associé ne sera pas le même. Par ailleurs, hzarikia01 ne peut pas avoir deux collections de mêmes types.

Table **LISTS** :

- L'identifiant de la liste (NOT NULL) en « **PRIMARY KEY** » est **UNIQUE** et s'auto-incrémente.
- L'identifiant de la collection en « **FOREIGN KEY** » est une clé étrangère qui fait référence à la table **COLLECTIONS.** Une liste est associée à une collection.
- Le nom et le type de la liste doit être « NOT NULL ».
- Les listes d'une même collection doivent être unique. L'unicité de la liste se mesure avec le nom de la liste (pas de doublon).
- La note est comprise entre 1 et 20.

Exemple : Dans une collection « Livres », il n'est pas possible d'avoir deux fois la liste « Horror ».

Table **OBJECTS_USERS** :

- L'identifiant de l'objet utilisateur (NOT NULL) en « **PRIMARY KEY** » est **UNIQUE** et s'auto incrémente.
- L'identifiant de l'objet référence et l'identifiant de la liste en « **FOREIGN KEY** » sont des clés étrangères des tables **OBJETS_REFERENCES** et **LISTS**. Un objet utilisateur est référé à un objet de référence et à une liste.
- Un objet ne peut être qu'une fois dans une même liste.
- La note est comprise entre 1 et 20.
- Si l'objet est possédé par l'utilisateur l'attribut « Have1_wish0 » (NOT NULL) sera à 1 sinon il sera à 0 pour le souhait de l'objet. On considère, si L'objet n'est pas possédé ni souhaité alors l'objet n'est pas dans la liste.
- La date d'enregistrement (NOT NULL) de l'objet est la date d'ajout de celle-ci.

Exemple : Il n'est pas possible d'avoir un livre qui est à la fois dans la liste « Horror » et dans la liste « Action ». Cependant, le livre peut être dans une autre liste d'une autre collection.

Table **OBJECTS_REFERENCES** :

- L'identifiant de la référence d'objet (NOT NULL) en « **PRIMARY KEY** » est **UNIQUE** et s'auto-incrémente.
- Le nom de l'objet doit être « NOT NULL ».
- Un objet est unique (nom différent).

1.3 Modèle Logique relationnel

USERS(Id_login : STRING, Id_password : STRING, Last_name : STRING, First_name : STRING, Address : STRING, Date_birth : DATE, Date_register : DATE)

COLLECTIONS(Id_collection : INT, Id_login : STRING, Type_cultural : STRING)

LISTS(Id_list : INT, Id_collection : INT, Name_list : STRING, Type_list : STRING, Description_list : STRING, Note_list : INT)

OBJECTS_REFERENCES(Id_object : INT, Name_object : STRING, Author_object : STRING, Description_object : STRING)

OBJECTS_USERS(Id_object_user : INT, Id_object : INT, Id_list : INT, Note_object : INT, Comment_object : STRING, Have1_wish0 : BOOLEAN, Date_register : DATE)

1.4 Exemple de construction de la base

Table **USERS** :

Id_login	Id_password	Last_name	First_name	Address	Date_birth	Date_register
'hzarikia01'	'HZarikian01'	'ZARIKIAN'	'Hayk'	'NULL'	2000/01/05	2022/10/30
'jgosling02'	'JGosling'	'GOSLING'	'James'	'NULL'	1955/05/19	2022/10/30

Table **COLLECTIONS** :

Id_collection	Id_login	Type_cultural
1	'hzarikia01'	'Livres'
2	'Jgosling02'	'Livres'
3	'hzarikia01'	'Jeux vidéos'

Table **LISTS** :

Id_list	Id_collection	Name_list	Type_list	Description_list	Note_list
1	1	'Horreur top'	'Horreur'	'Top horreur livres'	18
2	3	'Games top'	'Action'	'Top games'	20
3	2	'Aventure top'	'Aventure'	'Top aventure'	15

Table **OBJETS_REFERENCES** :

Id_object	Name_object	Author_object	Description_object
1	'Resident Evil'	'Patrick Hellio'	'Des zombies et des…'
2	'Horror games'	'NULL'	'Ne te retourne pas…'
3	'Call of Duty MW2'	'Activision'	'NULL'
4	'God of War'	'Monica Studio'	'NULL'

Table **OBJECTS_USERS** :

Id_object_user	Id_object	Id_list	Note_object	Comment_object	Have1_wish0
1	1	1	18	'Bien aimé'	1
2	1	2	20	'Bien aimé'	1
3	2	1	18	'Bien aimé'	1
4	3	3	15	'Bien aimé'	1
5	3	3	NULL	'NULL'	0

Date_register
2022/10/30
2022/10/30
2022/10/30
2022/10/30
2022/10/30

2 Implémentation de la base de données

2.1 Création des tables

La création des tables se fait avec le fichier « createTableCultural.sql » qui construit la table **USERS**, **COLLECTIONS**, **OBJECTS_REFERENCES**, **LISTS** et **OBJECTS_USERS** avec les séquences **Id_collection**, **Id_object_reference**, **Id_list** et **Id_object_user** pour l'auto incrémentation de ces attributs.

2.2 Suppression des tables

La suppression des tables se fait avec le fichier « deleteTableCultural.sql » qui supprime par l'ordre inverse de la création des tables **OBJECTS_USERS**, **LISTS**, **OBJECTS_REFERENCES**, **COLLECTIONS** avec les séquences **Id_object_user**, **Id_list**, **Id_object_reference**, **Id_collection.**

2.3 Insertion de données

L'insertion des données se fait avec le fichier « insertDataCultural.sql » qui remplit les tables **USERS**, **COLLECTIONS**, **OBJECTS_REFERENCES**, **LISTS** et **OBJECTS_USERS** de données qui font références à des personnages, livres, musiques, films et des jeux vidéos qui sont réels pour apporter plus de cohérence dans la construction de la base de données.

2.4 Script d'exécution des fichiers

Le script « scriptLaunchBDCultural.sql » lance la création des tables, l'insertion des données puis l'affichage des tables, enfin la suppression de celle-ci. Attention, il faut être connecté à une base avec la commande « sqlplus ».

Dans le cas où le script ne fonctionne pas, veuillez consulter le répertoire « captureResult ».