Rapport du project IFT2935

Groupe 25

Yingxu Huo
Sydney Luu
Ru Qian
Qiwu Wen

Introduction

Dans le cadre de ce projet de bases de données, nous avons développé une application de gestion de bibliothèque.
L'objectif principal est de modéliser, implémenter et exploiter une base de données permettant de gérer les livres, les adhérents, les emprunts et les commandes, tout en assurant la normalisation et l'optimisation des requêtes.

Pour répondre aux exigences du projet, nous avons suivi toutes les étapes demandées, depuis la modélisation E/A jusqu'au développement d'une interface graphique permettant de répondre aux différentes questions sur les données.

Choix techniques :

Initialement, l'implémentation de la base de données devait se faire sur PostgreSQL. Cependant, après évaluation des objectifs du projet, nous avons estimé que l'application était exclusivement destinée aux gestionnaires de la bibliothèque et non à un large public de lecteurs. Ainsi :

1. La concurrence des accès aux données est faible, car il y aura peu de gestionnaires utilisant simultanément l'application, contrairement à une plateforme publique.
2. La confidentialité des données doit être assurée, car les informations des adhérents et des prêts sont sensibles.
3. Le déploiement peut être effectué localement sur un nombre fixe de machines, sans avoir besoin de communications HTTP entre client et serveur, ce qui réduit les risques liés aux connexions réseau.
4. Le recours à un serveur n'étant pas indispensable, nous avons choisi d'éviter les ressources supplémentaires liées à l'hébergement et à la maintenance d'un serveur.

Sur la base de ces 4 points, nous avons jugé pertinent d'utiliser SQLite à la place de PostgreSQL. Cela a transformé l'architecture du projet : au lieu d'une communication serveur-GUI, nous avons opté pour un modèle local fichier-GUI.
Cette solution a simplifié le processus de déploiement et s'est mieux alignée avec les besoins réels de l'application décrits ci-dessus.

Pour la partie développement, nous avons utilisé Python avec la bibliothèque Tkinter afin de concevoir l'interface graphique.

Le ficher FicherPostgre.sql est la version Postgre pour la correction du projet. Et le

fichier Projet.sql est utilisé pour la transformation vers le ficher .db afin d'exécuter le programme d'interface GUI.

VideoProjet2935.mov est une petite démonstration vidéo pour montrer tous les fonctionnalités de l'interface GUI.
Les dépendances nécessaires sont seulement les packages tkinter et sqlite3, donc il y a aucun et un fichier README fournit les instructions d'exécution ainsi qu'une présentation succincte du projet.

Par ailleurs, l'application a été packagée en fichier dist/interface_bibliotheque.exe afin de permettre une exécution directe et simplifiée sous le système d'exploitation Windows.

Le choix de SQLite a permis une meilleure adéquation avec les objectifs du projet et a facilité l'expérience utilisateur finale.

La modélisation

Figure 1 représente le modèle E/A qui correspond au projet bibliothèque.

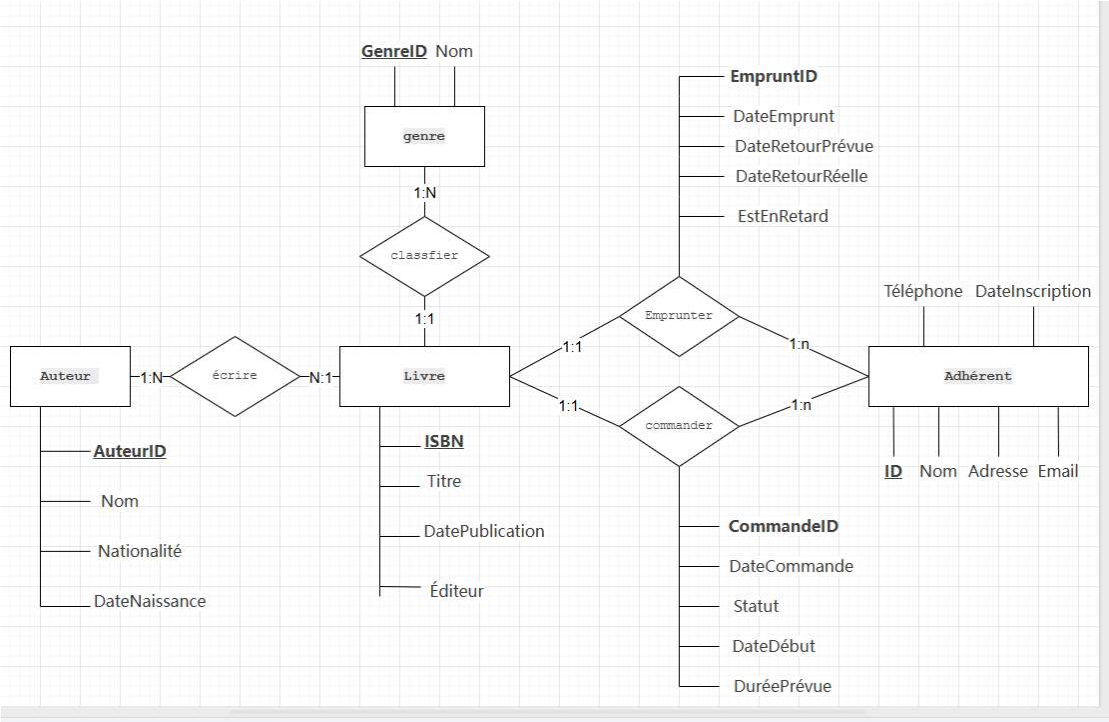


Figure 1 : modèleeses E/A du projet

Pour chaque entité du modèle, il est nécessaire de disposer d'un identifiant unique afin d'éviter tout conflit de données, comme des cas d'homonymie entre deux adhérents ou deux livres portant le même titre.

Ainsi, nous avons attribué un identifiant propre (ID) à chaque entité pour assurer l'unicité des enregistrements et garantir l'intégrité des relations.

Pour l'entité **Livre**, nous avons choisi d'utiliser l'**ISBN** comme clé primaire.
L'ISBN est un numéro normalisé attribué à chaque ouvrage publié, ce qui en fait un identifiant naturel et fiable pour la gestion de livres.

Nous avons supposé, pour simplifier le modèle, qu'il n'existait pas de duplications d'exemplaires pour un même ISBN au sein de la bibliothèque.

Cette approche permet de simplifier la gestion des données tout en respectant la cohérence du modèle.

Dans la gestion courante d'une bibliothèque, les adhérents sont généralement contactés par SMS, courriel ou courrier postal afin de les informer des dates de retour, des événements spéciaux, ou des changements concernant leur statut d'abonnement.
Ainsi, il est nécessaire de conserver dans la base de données des informations de contact précises telles que **l'adresse**, **l'email** et **le numéro de téléphone**.
Cela permet d'assurer une communication efficace et un meilleur suivi des interactions avec les adhérents.

Transformation

Dans la transformation du modèle Entité/Association vers le modèle relationnel, chaque **entité** devient une **relation** contenant tous ses attributs, et son identifiant devient une **clé primaire**.
Les **associations de type 1:N** sont gérées en ajoutant une **clé étrangère** du côté N. (Emprunter et Commande)
Les **associations de type N:N** sont transformées en **relations autonomes** avec une **clé primaire composée**.(Ecrire)
Cette approche garantit l'intégrité référentielle du modèle.

Modèle relationnel obtenu

Genre(GenreID, Nom)

Livre(ISBN, Titre, DatePublication, Editeur, #GenreID)

Auteur(AuteurID, Nom, Nationalite, DateNaissance, Adresse)

Ecrire(#AuteurID, #ISBN)

Adherent(ID ,Nom, Adresse, Email, Telephone, DateInscription)

Emprunter(EmpruntID, #ISBN, #AdherentID, DateEmprunt, DateRetourPrevue, DateRetourReelle, EstEnRetard)

Commander(CommandeID, #ISBN, #AdherentID, DateCommande, Statut, DateDebut, DureePrevue)

Normalisation

Dépendances fonctionnelles

Pour chaque relation, nous définissons les dépendances fonctionnelles principales :

·Genre(GenreID, Nom)

Dépendances : GenreID -> Nom

·Livre(ISBN, Titre, DatePublication, Editeur, #GenreID)

Dépendances : ISBN-> Titre, DatePublication, Editeur, GenreID

·Auteur(AuteurID, Nom, Nationalite, DateNaissance, Adresse)

Dépendances : AuteurID -> Nom, Nationalite, DateNaissance, Adresse

·Ecrire(#AuteurID, #ISBN)

Dépendances : aucune, puisque la clé première contient déjà tous les attributs

·Adherent(ID ,Nom, Adresse, Email, Telephone, DateInscription)

Dépendances :

ID -> Nom, Adresse, Email, Telephone, DateInscription

·Emprunter(EmpruntID, #ISBN, #AdherentID, DateEmprunt, DateRetourPrevue, DateRetourReelle, EstEnRetard)

Dépendances :

EmpruntID -> ISBN, AdherentID, DateEmprunt, DateRetourReelle

DateEmprunt -> DateRetourPrevue

(DateRetourPrevue , DateRetourReelle) ->EstEnRetard

·Commander(CommandeID, #ISBN, #AdherentID, DateCommande, Statut,

DateDebut, DureePrevue)

CommandeID->ISBN, AdherentID, DateCommande, Statut, DateDebut, DureePrevue

NOTE : **DateDebut** et **DureePrevue** dépendent fonctionnellement de CommandeID. Cependant, leur présence effective est soumise à une contrainte métier liée à l'attribut Statut : ils sont uniquement renseignés lorsque la commande est honorée.

Normaliser les tables

À l'exception de la table **Emprunter**, toutes les autres tables sont telles que les attributs non-clés dépendent uniquement de la clé primaire unique, ou qu'il n'existe aucune dépendance fonctionnelle supplémentaire.
Ainsi, la **1NF** est satisfaite (absence de valeurs répétées), la **2NF** est satisfaite (les attributs dépendent entièrement d'une clé primaire simple et non d'une partie de clé composite), la **3NF** est satisfaite (aucun attribut non-clé ne dépend d'un autre attribut non-clé), et la **BCNF** est satisfaite (tous les attributs dépendent uniquement d'une clé candidate).

Dans la table **Emprunter**, l'attribut **EstEnRetard** peut être déduit à partir de **DateRetourPrevue** et **DateRetourReelle**.
De même, **DateRetourPrevue** peut être calculé automatiquement à partir de **DateEmprunt**, en appliquant la règle métier d'une durée d'emprunt fixe de 14 jours.
Afin d'éviter toute redondance et de respecter les règles de normalisation, ces deux attributs ont été supprimés.

Après cette modification, la table **Emprunter** ne contient plus que des attributs directement dépendants de la clé primaire **EmpruntID**,
et respecte ainsi pleinement les formes normales 1NF, 2NF, 3NF ainsi que la **BCNF**.

Le schéma final

·Emprunter(EmpruntID, #ISBN, #AdherentID, DateEmprunt, DateRetourReelle)

Dépendances :

EmpruntID -> ISBN, AdherentID, DateEmprunt, DateRetourReelle

Et toutes les autres tables restent les mêmes. Et le schéma final respecte le **BCNF**.

L’implémentation

Conformément aux exigences du projet, l'implémentation de la base de données devait être réalisée sur **PostgreSQL**.
Cependant, pour des raisons pratiques expliquées précédemment (développement local simplifié avec Tkinter), l'application principale utilise une base **SQLite**.
Afin de respecter le cahier des charges, nous avons donc fourni deux versions du schéma SQL :

- Une version compatible **SQLite**, utilisée par l'application.
- Une version spécifique pour **PostgreSQL**, respectant le modèle normalisé obtenu après l'étape de normalisation.

Toutes les relations ont été remplies avec plus de 10 tuples chacune, comme demandé.

La version PostgreSQL est aussi disponible pour la correction du projet : ficherPostgre.sql

La version SQLite pour rouler sqliteTransform.py est le ficher Projet.sql

Question/réponse

Dans cette section, nous avons sélectionné quatre questions complexes portant sur l’exploitation de la base de données.
Pour chaque question, nous présentons :

- une expression en algèbre relationnelle,
- une requête SQL correspondante,

Les requêtes ont été conçues de manière à retourner des résultats pertinents et à respecter les critères de performance attendus.

Nous utilisons le format PostgreSQL dans ce rapport.

1. Trouver les 3 membres ayant emprunté le plus de livres actuellement.

```
SELECT a.ID, a.NOM, COUNT(e.EmpruntID) AS NbEmprunts
FROM Adherent a
        JOIN Emprunter e ON a.ID = e.AdherentID
```

```
GROUP BY a.Nom

ORDER BY NbEmprunts DESC

LIMIT 3;
```

π_Nom, COUNT(EmpruntID) (γ_Nom, COUNT(EmpruntID)→NbEmprunts (Adherent ⋈ Emprunter))

2. Lister les livres actuellement en état "en_attente" ainsi que les coordonnées des adhérents correspondants.

```
SELECT c.CommandeID, l.Titre, a.Nom, a.Email, a.Téléphone, c.DateCommande

FROM Commander c

        JOIN Livre l ON c.ISBN = l.ISBN

        JOIN Adherent a ON c.AdherentID = a.ID

WHERE c.Statut = 'en_attente';
```

π_Titre, Nom, Email, Téléphone, DateCommande

(σ_Statut='en_attente' (Commander ⋈ Livre ⋈ Adherent))

3. Lister les adhérents ayant emprunté le livre <*1984*> ainsi qu'au moins un autre livre du genre "Philosophie"

```
SELECT DISTINCT a.Nom

FROM Adherent a

WHERE a.ID IN (

    SELECT e1.AdherentID

    FROM Emprunter e1

            JOIN Livre l1 ON e1.ISBN = l1.ISBN
```

```
        WHERE l1.Titre = '1984'
    )
    AND a.ID IN (
        SELECT e2.AdherentID
        FROM Emprunter e2
                JOIN Livre l2 ON e2.ISBN = l2.ISBN
        WHERE l2.GenreID = 3    -- Philosophie
    );
    ------------------------------------------------------------
    π_Nom (
        (σ_Titre='1984'(Livre) ⋈ Emprunter) [AdherentID]
        ∩
        (σ_GenreID=3(Livre) ⋈ Emprunter) [AdherentID]
    ) ⋈ Adherent
```

NOTE: Nous avons légèrement modifié cette question dans notre code pour rechercher les emprunteurs d'un livre et/ou d'un genre donné.

4. Trouver les emprunts déjà retournés avec un retard de plus de 3 jours, ainsi que les informations sur les adhérents et les livres concernés.

```
SELECT a.ID AS Adherent, l.Titre, e.DateEmprunt , e.DateRetourRéelle
FROM Emprunter e
        JOIN Adherent a ON e.AdherentID = a.ID
        JOIN Livre l ON e.ISBN = l.ISBN
WHERE e.DateRetourRéelle IS NOT NULL
   AND e.DateRetourRéelle > e. DateEmprunt + INTERVAL '17 days';
//14 jours de duré légal+ 3 jours de retard
```

π_Nom, Titre, DateEmprunt, DateRetourRéelle

(σ_(DateRetourRéelle > (DateEmprunt +17))

(Emprunter ⋈ Adherent ⋈ Livre))

NOTE: Nous avons légèrement modifié cette question dans notre code pour rechercher les emprunteurs avec N jours de retard au lieu de 3 jours.

Développement [Python]

L'application graphique a été développée en **Python** avec la bibliothèque **Tkinter**. Elle permet de répondre aux quatre questions choisies dans la section précédente grâce à l'affichage de résultats interactifs.

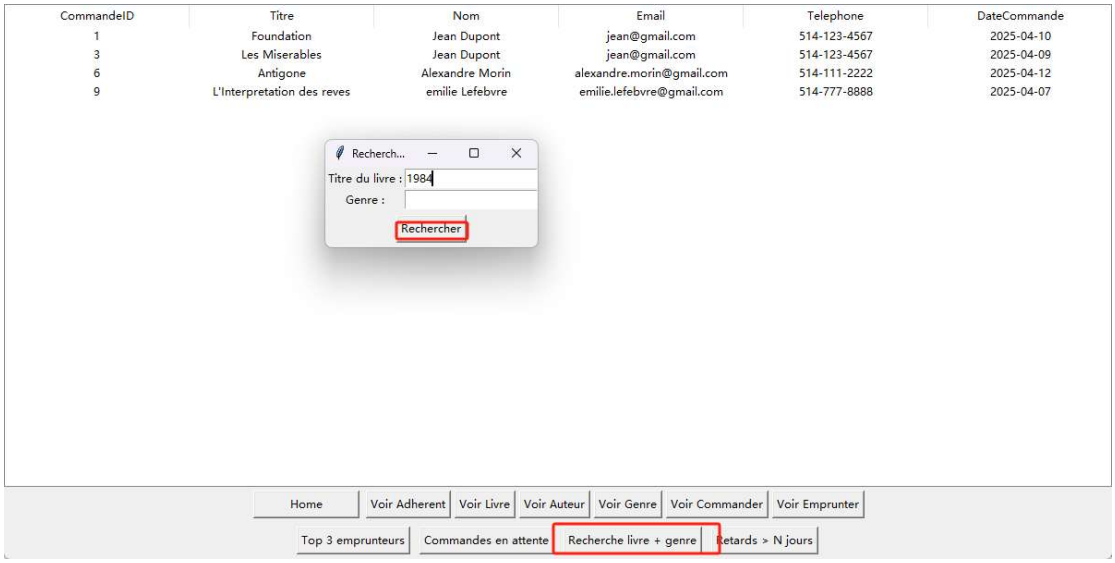
Organisation du projet Python :

Le fichier **interface_bibliotheque.py** constitue le programme principal de l'application graphique de gestion de la bibliothèque.
Le fichier **sqliteTransform.py** est un script utilitaire permettant de transformer le fichier SQL de création de base de données en un fichier **SQLite (.db)** utilisable par l'application.

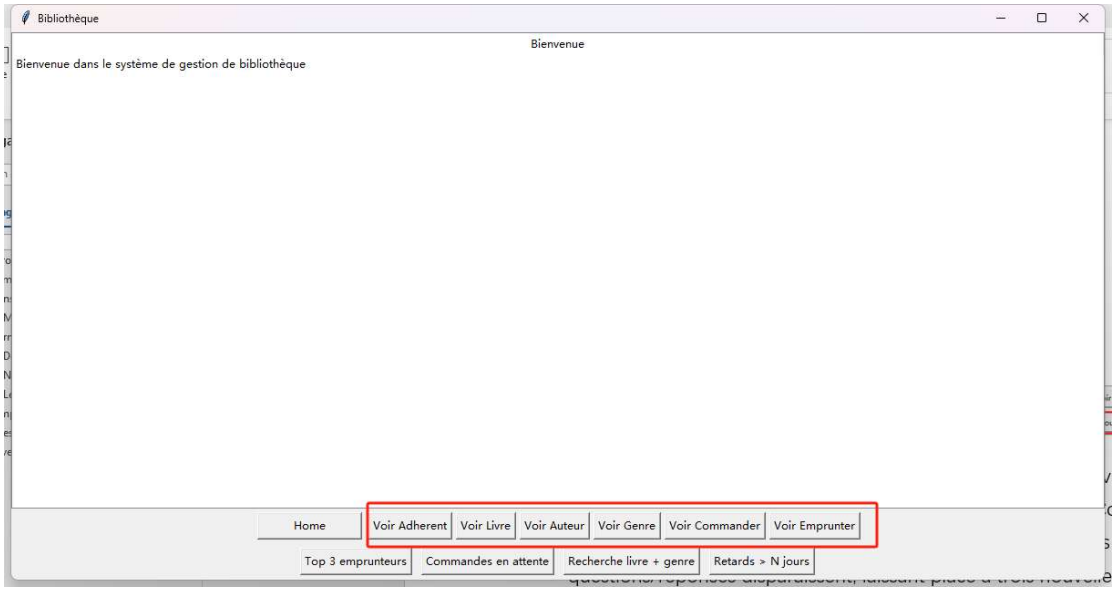
Explication de l'interface

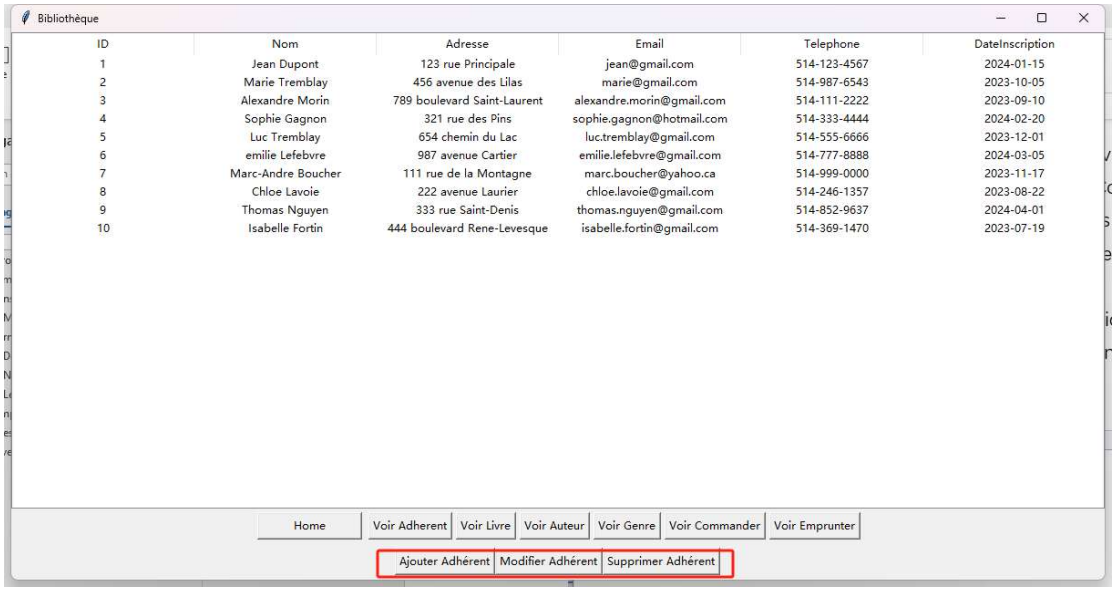
Toutes les opérations de modification, d'ajout ou de suppression réalisées via l'interface graphique affectent directement le fichier **bibliotheque.db**, et non le fichier **.sql** initial.
Par conséquent, pour éviter toute perte de données, il est recommandé de **sauvegarder le fichier .db** avant d'exécuter à nouveau **sqliteTransform.py**, car ce dernier régénère une base vide à partir du script SQL.

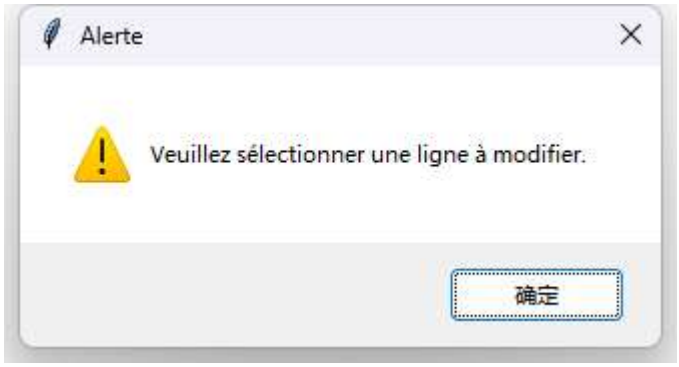
Sur la page d'accueil (**Home**), quatre boutons dédiés permettent d'exécuter directement chacune des requêtes liées aux questions/réponses sélectionnées. Chaque bouton affiche instantanément les résultats dans un tableau.

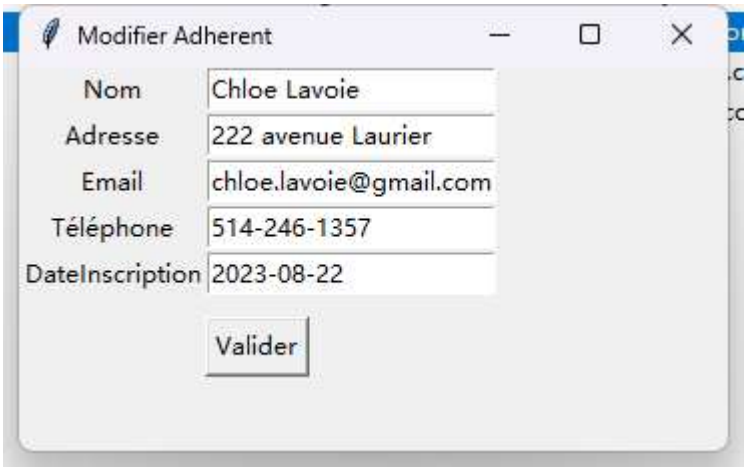


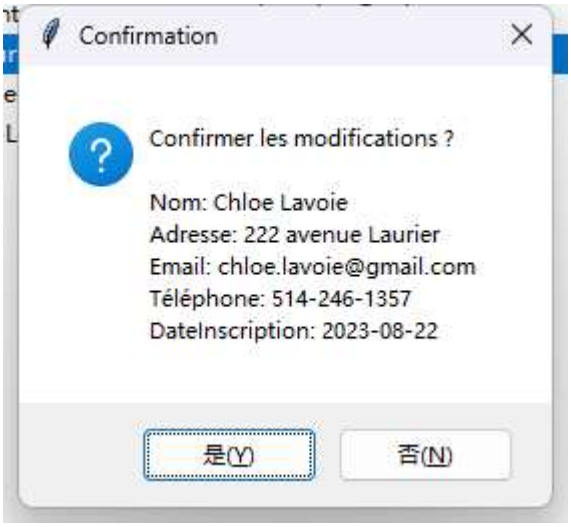
De plus, une série de boutons est proposée pour accéder à la visualisation des différentes tables principales (Adherent, Livre, Auteur, Genre, Commander, Emprunter).
Lorsque l'utilisateur sélectionne l'une de ces tables, les boutons liés aux questions/réponses disparaissent, laissant place à trois nouvelles options spécifiques : **Ajouter**, **Modifier** et **Supprimer** des enregistrements.
Ces opérations sont disponibles uniquement après avoir sélectionné un élément, et des messages d'alerte apparaissent automatiquement en cas de tentative sans sélection préalable.


Bibliothèque
Bienvenue
Bienvenue dans le système de gestion de bibliothèque
Home
Voir Adherent
Voir Livre
Voir Auteur
Voir Genre
Voir Commander
Voir Emprunter
Top 3 emprunteurs
Commandes en attente
Recherche livre + genre
Retards > N jours


Bibliothèque
ID Nom Adresse Email Telephone DateInscription
1 Jean Dupont 123 rue Principale jean@gmail.com 514-123-4567 2024-01-15
2 Marie Tremblay 456 avenue des Lilas marie@gmail.com 514-987-6543 2023-10-05
3 Alexandre Morin 789 boulevard Saint-Laurent alexandre.morin@gmail.com 514-111-2222 2023-09-10
4 Sophie Gagnon 321 rue des Pins sophie.gagnon@hotmail.com 514-333-4444 2024-02-20
5 Luc Tremblay 654 chemin du Lac luc.tremblay@gmail.com 514-555-6666 2023-12-01
6 emilie Lefebvre 987 avenue Cartier emilie.lefebvre@gmail.com 514-777-8888 2024-03-05
7 Marc-Andre Boucher 111 rue de la Montagne marc.boucher@yahoo.ca 514-999-0000 2023-11-17
8 Chloe Lavoie 222 avenue Laurier chloe.lavoie@gmail.com 514-246-1357 2023-08-22
9 Thomas Nguyen 333 rue Saint-Denis thomas.nguyen@gmail.com 514-852-9637 2024-04-01
10 Isabelle Fortin 444 boulevard Rene-Levesque isabelle.fortin@gmail.com 514-369-1470 2023-07-19
Home
Voir Adherent
Voir Livre
Voir Auteur
Voir Genre
Voir Commander
Voir Emprunter
Ajouter Adhérent
Modifier Adhérent
Supprimer Adhérent

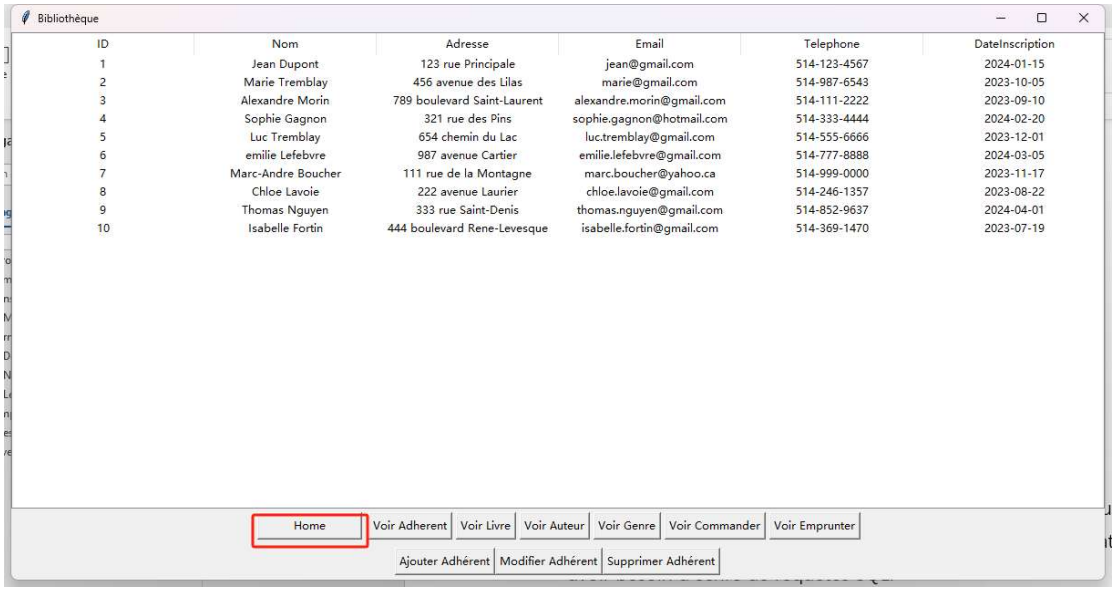

Alerte
Veuillez sélectionner une ligne à modifier.
确定





Cette conception vise à **simplifier la gestion des données** pour les administrateurs de la bibliothèque, en leur permettant d'effectuer toutes les opérations courantes sans avoir besoin d'écrire de requêtes SQL.

Enfin, un bouton **Home** reste accessible à tout moment, permettant de revenir facilement à la page principale pour exécuter de nouvelles requêtes analytiques.


Bibliothèque
ID Nom Adresse Email Telephone DateInscription
1 Jean Dupont 123 rue Principale jean@gmail.com 514-123-4567 2024-01-15
2 Marie Tremblay 456 avenue des Lilas marie@gmail.com 514-987-6543 2023-10-05
3 Alexandre Morin 789 boulevard Saint-Laurent alexandre.morin@gmail.com 514-111-2222 2023-09-10
4 Sophie Gagnon 321 rue des Pins sophie.gagnon@hotmail.com 514-333-4444 2024-02-20
5 Luc Tremblay 654 chemin du Lac luc.tremblay@gmail.com 514-555-6666 2023-12-01
6 emilie Lefebvre 987 avenue Cartier emilie.lefebvre@gmail.com 514-777-8888 2024-03-05
7 Marc-Andre Boucher 111 rue de la Montagne marc.boucher@yahoo.ca 514-999-0000 2023-11-17
8 Chloe Lavoie 222 avenue Laurier chloe.lavoie@gmail.com 514-246-1357 2023-08-22
9 Thomas Nguyen 333 rue Saint-Denis thomas.nguyen@gmail.com 514-852-9637 2024-04-01
10 Isabelle Fortin 444 boulevard Rene-Levesque isabelle.fortin@gmail.com 514-369-1470 2023-07-19
Home
Voir Adherent
Voir Livre
Voir Auteur
Voir Genre
Voir Commander
Voir Emprunter
Ajouter Adhérent
Modifier Adhérent
Supprimer Adhérent