



Rapport



Seyed Ahmad Farsad

20031258

Carlos Augusto Pluma Seda

20258763

Khalil Rerhrhaye

20179868

Jonas Gabirot

20185863

Professeur : Michel Boyer

Démonstrateur : Ahmed Imed Eddine Rabah

IFT2935

22 Avril 2023

1. La modélisation

Le sujet choisi pour ce projet est la bibliothèque (2). Le modèle E/A est présenté ci-dessus. Les objets ayant une existence propre et ayant un intérêt pour au moins un traitement de l'application sont **Adhèrent** et **Livre**. Cependant, on a ajouté la **Bibliothèque** dans le diagramme qu'on peut ignorer.

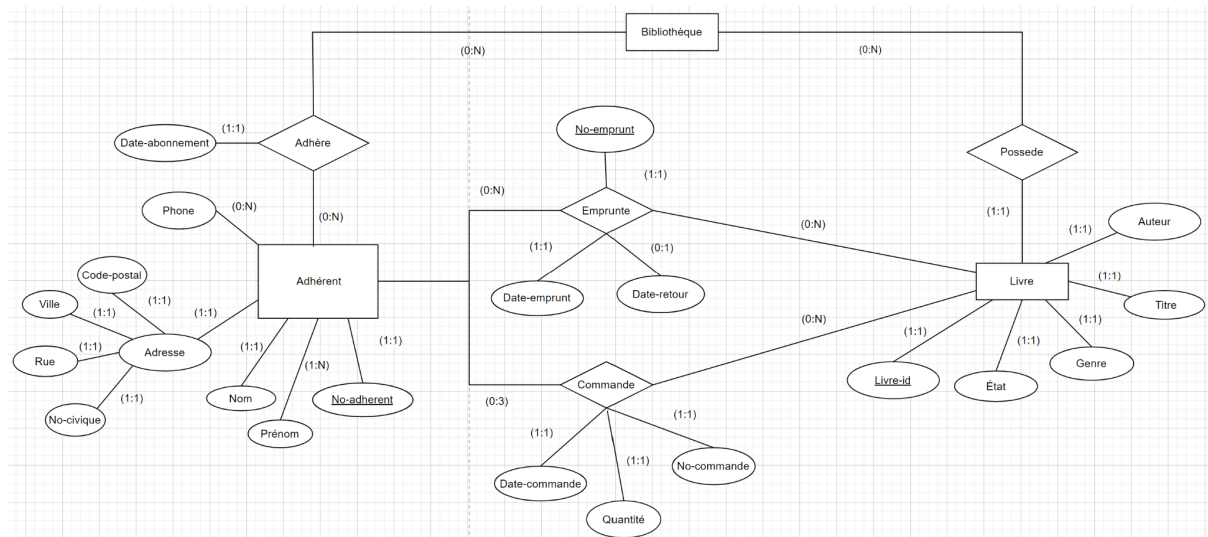


Figure 1

Puisque la **Bibliothèque** n'est pas un objet d'intérêt, on n'a pas tenu compte des attributs pour la Bibliothèque et les associations correspondantes (seulement un attribut Date-abonnement pour plus de clarté). En fait, on peut simplement considérer le diagramme ci-dessus:

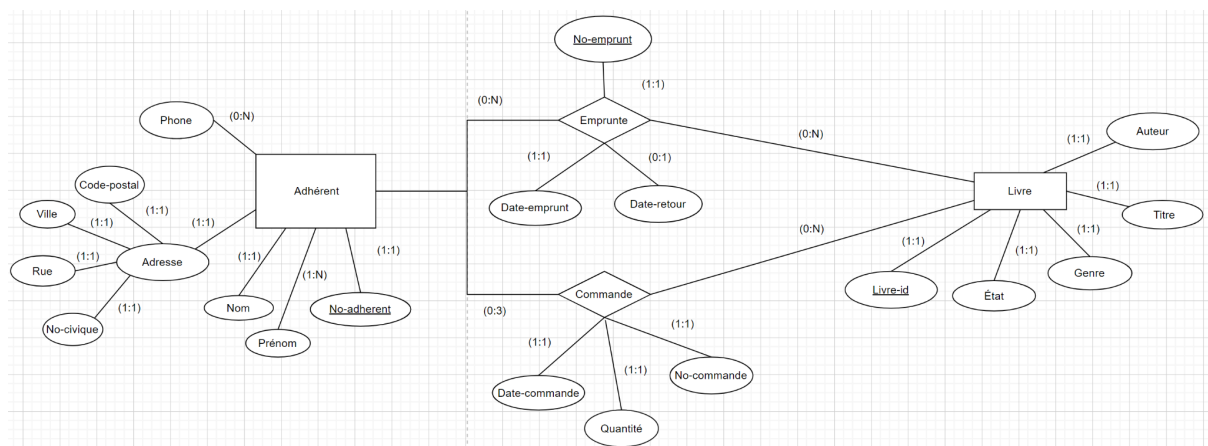


Figure 2

Pour les objets présents dans le diagramme, les attributs importants ont été associés. Pour le livre l'attribut "État" est un attribut binaire (qui détermine si le livre existe ou déjà emprunté. On n'utilisera pas cet attribut). On peut encore ajouter d'autre attributs comme Sexe et NAS pour le **Adherent** ou date-achat et editeur pour le **Livre** qu'on les a ignore, vue qu'ils ne sont pas determinants dans les transactions et la gestion de la abse de donnees de la bibliothèque.

Au lieu d'utiliser No-commande et No-emprunte pour identifier les associations **Commande** et **Emprunte**, on peut utiliser les clés primaires de l'**Adhèrent** et du **Livre**, comme illustré dans le Figure 3.

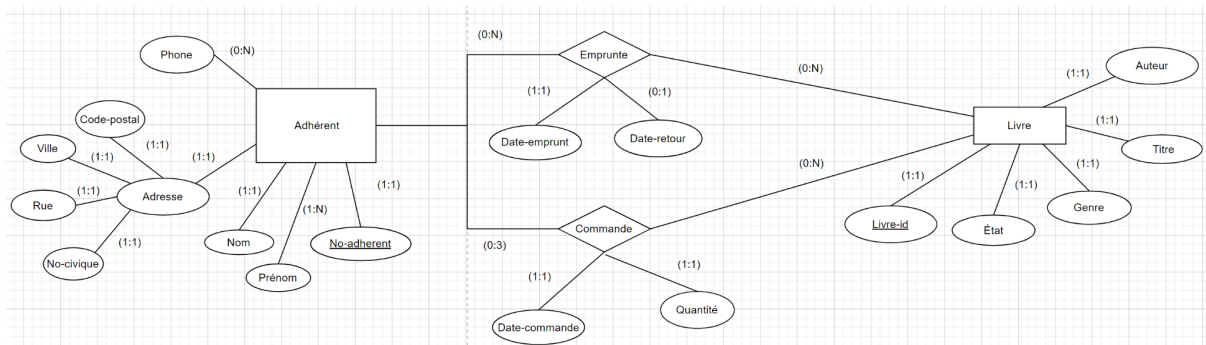


Figure 3

Pour les types d'association concernant Livre-Adhèrent, il y a deux approches possible:

- Interprétation où l'on ne garde que les emprunts courants si ce sont les seuls à lister pour les clients.

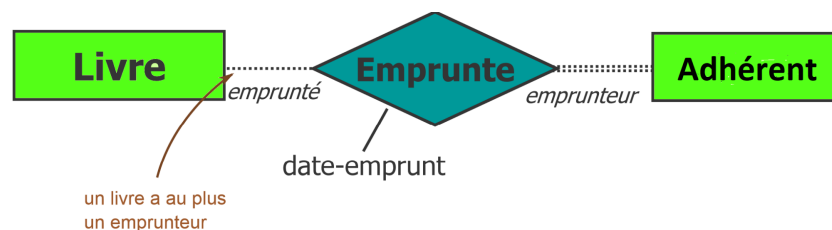


Figure 4

- Interprétation où l'on garde trace des emprunts pour pouvoir lister les emprunts même des livres retournés (un booléen suffirait au lieu de la date du retour).

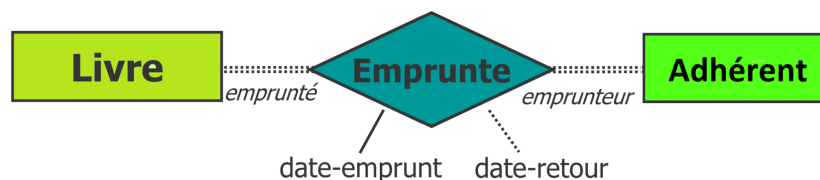


Figure 5

Dans la description du sujet bibliothèque, puisque la date de rendre est compte tenue, donc, on a choisi la deuxième approche. Pour l'action de commander, c'est pareil, car un livre peut être commandé par 0 à N abonné.

2. La transformation

Dans cette section, nous décrivons les étapes d'un algorithme de passage (règles du passage) du modèle E/A vers le modèle relationnel.

Étape 1 : Transformation des Types d'Entités Régulières. Pour chaque type d'entité régulière (forte) E dans le schéma E/A, créons une relation R qui inclut tous les attributs simples de E. N'incluons que les attributs de composants simples d'un attribut composite. Choisissons un des attributs clés de E comme clé primaire pour R. Si la clé choisie de E est

une composition, alors l'ensemble des attributs simples qui le composent forment ensemble le principal clé de R. Si plusieurs clés ont été identifiées pour E lors de la conception conceptuelle, les informations décrivant les attributs qui composent chaque clé supplémentaire est conservée afin de préciser clés supplémentaires (uniques) de la relation R. Les connaissances sur les clés sont également conservées à des fins d'indexation et d'autres types d'analyses.

Dans ce travail, nous choisissons No-adhérent, Livre-id, No-emprunt et No-commande comme clés primaires pour les relations Adhérent, Livre, Emprunte et Commande respectivement.

Le résultat après cette étape de mappage est comme suit:

+ **Adhérent**(No-adhérent, Nom, Prénom, Adresse, Phone)

+ **Livre**(Livre-id, Titre, Genre, Auteur)

Étape 2 : Transformation des types d'entités faibles. Pour chaque type d'entité faible W dans le schéma E/A avec le type d'entité propriétaire ET, créez une relation R et incluez tous les attributs simples (ou composants simples d'attributs composites) de W en tant qu'attributs de R. De plus, incluez en tant qu'attributs de clé étrangère de R, le ou les attributs de clé primaire de la ou des relations qui correspondent au(x) type(s) d'entité propriétaire ; cela prend soin de mapper le type de relation d'identification de W. La clé primaire de R est la combinaison de la ou des clés primaires du ou des propriétaires et de la clé partielle du type d'entité faible W, le cas échéant. S'il existe une entité faible de type E2 dont le propriétaire est également une entité faible de type E1, alors E1 doit être mappée avant E2 pour déterminer sa clé primaire en premier.

Dans notre travail n'a pas intégré des attributs faibles(weak), car on a supposé que chaque livre a un identifiant unique. Autrement dit, il n'y a pas d'entité **Exemplaire** pour les livres (comme dans le Figure 6), alors qu'il se peut qu'il en existe plusieurs dans la bibliothèque.

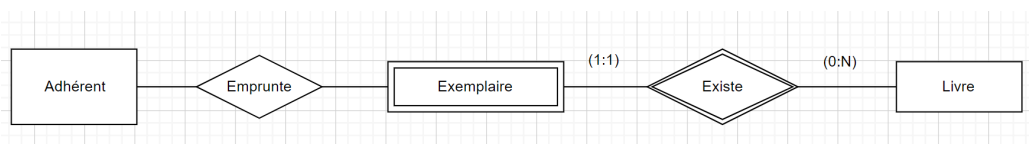


Figure 6

Étape 3 : Transformation des types de relations binaires 1:1. Pour chaque type de relation binaire 1:1 R dans le schéma E/A, identifiez les relations S et T qui correspondent aux types d'entités participant à R. Il existe trois approches possibles : (1) l'approche par clé étrangère, (2) la relation fusionnée et (3) l'approche des références croisées ou des relations relationnelles.

Dans notre projet, il n'y a pas de telle relation binaire 1:1, sauf si on tient compte l'entité du **Bibliothèque** (et donc nous aurions Livre-Possède), qui n'est pas de notre intérêt.

Étape 4 : Transformation des types de relations binaires 1:N. Il existe deux approches possibles : (1) l'approche par clé étrangère et (2) l'approche par référence croisée ou relation de relation. La première approche est généralement préférée car elle réduit le nombre de tables.

L'approche par clé étrangère : pour chaque type de relation R binaire régulier 1:N, identifiez la relation S qui représente le type d'entité participante du côté N du type de relation. Inclure comme clé étrangère dans S la clé primaire de la relation T qui représente l'autre type d'entité participant à R ; nous le faisons parce que chaque instance d'entité du côté N est liée à au plus une instance d'entité du côté 1 du type de relation. Inclure tout simple attributs (ou composants simples d'attributs composites) du type de relation 1:N en tant qu'attributs de S.

Encore une fois, une telle relation n'existe pas dans notre diagramme.

Étape 5 : Transformation des types de relations binaires M:N. Dans le modèle relationnel traditionnel sans attributs multivalués, la seule option pour les relations M:N est l'option de relation de relation (référence croisée). Pour chaque type de relation binaire M:N R, créons une nouvelle relation S pour représenter R. Incluons en tant qu'attributs de clé étrangère dans S les clés primaires des relations qui représentent les types d'entités participantes ; leur combinaison formera la clé primaire de S. Incluons également tous les attributs simples du type de relation M:N (ou composants simples d'attributs composites) en tant qu'attributs de S.

Dans notre travail, nous transformons le type de relation M:N **Emprunte** de la Figure 3 en créant la relation **Emprunte** dans la Figure 3. Nous incluons les clés primaires des relations **Adhérent** et **Livre** en tant que clés étrangères dans **Emprunte**.

Nous incluons également les attributs Date-emprunt et Date-retour dans **Emprunte** pour représenter les attributs Date-emprunt et Date-retour du type de relation. La clé primaire de la relation **Emprunte** est la combinaison des attributs de la clé étrangère {#No-adhérent, #Livre-id, Date-emprunt}.

+ **Emprunte** (#No-adhérent, #Livre-id, Date-emprunt, Date-retour)

On utilise la même approche pour l'association **Commande**.

Étape 6 : Transformation des attributs à plusieurs valeurs. Pour chaque attribut multivalué A, créons une nouvelle relation R. Cette relation R comporte un attribut correspondant à A, plus l'attribut de clé primaire K-comme une clé étrangère dans R - de la relation qui représente le type d'entité ou le type de relation qui a A comme attribut à valeurs multiples. La clé primaire de R est la combinaison de A et K. Si l'attribut multivalué est composé, nous incluons ses composants simples.

Dans notre sujet, les attributs multivalués possible sont **Phone**, **Auteur** et **Genre**. Comme il a été mentionné dans la description du sujet que “ **Les livres ont un titre, un genre et un auteur**”, on va les considérer comme les attributs monovalués. En ce qui concerne le **Phone**, on suppose que les abonnés enregistrent un seul numéro de téléphone dans leur profil de bibliothèque.

Étape 7 : Transformation des types de relations N-aires. dans notre diagramme il n'y a pas de telles relations. Donc pour résumer les règles de passage, on arrive finalement aux modèles relationnels ci-dessus:

+ **Adhérent**(No-adhérent, Nom, Prénom, Adresse, Phone)

+ **Livre**(Livre-id, Titre, Genre, Auteur)

- + **Emprunte**(#No-adhérent, #Livre-id, Date-emprunt, Date-retour)
- + **Commande**(#No-adhérent, #Livre-id, Date-commande, Quantité)

3. La normalisation

- Dépendances fonctionnelles pour chaque relation:

- **Adhérent:**
 $(A \rightarrow B, C, D, E)$
 No-adhérent \rightarrow Nom, Prénom, Adresse, Phone
 Un adhérent spécifique a un nom, prénom, une adresse et un numéro de téléphone uniques.
- **Livre:**
 $(F \rightarrow G, H, I)$
 Livre-id \rightarrow Titre, Genre, Auteur
 Le Livre-id est unique pour chaque livre et détermine ses autres attributs.
- **Emprunte:**
 $(A, F, J \rightarrow K)$
 No-adhérent, Livre-id, Date-emprunt \rightarrow Date-retour
 La combinaison de No-adhérent, Livre-id et Date-emprunt est unique et détermine la Date-retour.
- **Commande:**
 $(A, F, L \rightarrow M)$
 No-adhérent, Livre-id, Date-commande \rightarrow Quantité
 La combinaison de No-adhérent, Livre-id et Date-commande est unique et détermine la Quantité.

- Normalisation des tables:

- **Adhérent** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Livre** : Ce tableau n'est pas normalisé puisque dans la bibliothèque vous pouvez avoir plusieurs livres du même auteur (ou des livres qui ont plusieurs auteurs, si on considère l'attribut **Auteur** comme multivalué), donc dans le tableau nous aurons des répétitions d'auteurs, par exemple :

Livre-id	Titre	Genre	Auteur
1	Harry Potter and the Half-Blood Prince	Fantasy	J. K. Rowling
2	Harry Potter and the Deathly Hallows	Fantasy	J. K. Rowling

Pour résoudre cela et pouvoir normaliser ce tableau nous pouvons diviser le tableau **Livre** en deux tables: une pour les auteurs et une pour la relation entre les livres et les auteurs. Cela permettra à plusieurs auteurs d'être associés à un livre et vice versa:

- **Livre**(Livre-id, Titre, Genre)
- **Auteurs**(Auteur-id, Auteur-Nom)
- **Livre-Auteur**(#Livre-id, #Auteur-id)

La table Livre-Auteur est une table de relations qui relie les livres et les auteurs. Bien qu'il puisse sembler qu'il y ait des données répétées, il est en fait conçu pour gérer efficacement la relation plusieurs à plusieurs entre les livres et les auteurs. La répétition de clés dans une table de relations est un comportement attendu et non un problème en soi.

La table livre-auteur possède deux clés étrangères, Livre-id et Auteur-id, qui sont respectivement les clés primaires des tables livre et auteur. La combinaison de ces deux clés étrangères forme la clé primaire dans la table livre_auteur, garantissant que chaque combinaison de livre et d'auteur est unique. Par conséquent, ce tableau est considéré comme normalisé et il n'y aura pas de problèmes de redondance.

De même, dans le tableau des livres, nous aurons des répétitions de genres de livres puisqu'un genre n'est pas exclusif à un livre, ce qui provoque des valeurs répétées dans le tableau, par exemple:

Livre-id	Titre	Genre
1	Les Miserables	Science-fiction
2	Game of Thrones	Science-fiction

Pour résoudre ce problème et pouvoir normaliser cette table, nous pouvons diviser la table des livres en une autre table. Cela vous permet d'avoir plusieurs livres avec le même genre sans répéter les données dans la table des livres:

- **Livre**(Livre-id, Titre, Genre)
- **Auteurs**(Auteur-id, Auteur-Nom)
- **Livre-Auteur**(#Livre-id, #Auteur-id)
- **Genres**(#Livre-id, Genre-nom)

La table Genres a une clé étrangère Livre-id et une clé primaire Genre-nom. La combinaison de ces deux clés forme la clé primaire dans la table Genres, garantissant que chaque

combinaison de livre et de genre est unique. Par conséquent, ce tableau est considéré comme normalisé et il n'y aura pas de problèmes de redondance.

- **Genres** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.

Dans ce cas, la table Genres possède un attribut de clé primaire composé de deux attributs : Livre-id et Genre-nom. Puisqu'il s'agit d'une clé primaire, chaque combinaison de valeurs Livre-id et Genre-nom est garantie d'être unique dans la table, il n'y aurait donc pas de problèmes de redondance.

Il est vrai qu'il peut y avoir plusieurs livres d'un même genre, mais chacun d'eux aurait un identifiant unique dans la table Livre, ce qui garantit l'unicité de la valeur Livre-id dans la table Genres. Par conséquent, en incluant les deux attributs dans la clé primaire de la table Genres, nous nous assurons qu'il n'y aura pas de doublons dans la table et qu'elle est conforme à la forme normale Boyce-Codd (BCNF).

- **Emprunte** : cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Commande** : cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.

Les tables en BCNF remplissent les conditions suivantes :

1. Ils sont en 3NF, ce qui signifie que tous les attributs non clés dépendent fonctionnellement de la clé primaire complète et qu'il n'y a pas de dépendances transitives.
2. Pour chaque dépendance fonctionnelle $X \rightarrow Y$, X doit être une super-clé (c'est-à-dire un ensemble minimal d'attributs qui détermine de manière unique tous les autres attributs de la table).

Dans ce cas, après la modification, toutes les tables remplissent ces conditions.

- Le schéma final de la base de données obtenu:

- + Adhérent(No-adhérent, Nom, Prénom, Adresse, Phone)
- + Livre(Livre-id, Titre)
- + Auteurs(Auteur-id, Auteur-Nom)
- + Livre-Auteur(#Livre-id, #Auteur-id)
- + Genres(#Livre-id, Genre-nom)
- + Emprunte(#No-adhérent, #Livre-id, Date-emprunt, Date-retour)
- + Commande(#No-adhérent, #Livre-id, Date-commande, Quantité)

4. L'implémentation

- Création des tableaux

```
BEGIN TRANSACTION;

CREATE TABLE Adherent (
  No_adherent INTEGER NOT NULL PRIMARY KEY,
  Nom VARCHAR(255) NOT NULL,
  Prenom VARCHAR(255) NOT NULL,
  Adresse VARCHAR(255) NOT NULL,
  Phone VARCHAR(15) NOT NULL
);

CREATE TABLE Livre (
  Livre_id INTEGER NOT NULL PRIMARY KEY,
  Titre VARCHAR(255) NOT NULL
);

CREATE TABLE Genres (
  Livre_id INTEGER NOT NULL,
  Genre_nom VARCHAR(255) NOT NULL,
  PRIMARY KEY (Livre_id, Genre_nom)
);

CREATE TABLE Auteurs (
  Auteur_id INTEGER NOT NULL PRIMARY KEY,
  Auteur_nom VARCHAR(255) NOT NULL
);

CREATE TABLE Livre_Auteur (
  Livre_id INTEGER NOT NULL,
  Auteur_id INTEGER NOT NULL,
  PRIMARY KEY (Livre_id, Auteur_id)
);

CREATE TABLE Emprunte (
  No_adherent INTEGER NOT NULL,
  Livre_id INTEGER NOT NULL,
```

```

Date_emprunt DATE NOT NULL,
Date_retour DATE,
PRIMARY KEY (No_adherent, Livre_id, Date_emprunt)
);
CREATE TABLE Commande (
No_adherent INTEGER NOT NULL,
Livre_id INTEGER NOT NULL,
Date_commande DATE NOT NULL,
Quantite INTEGER NOT NULL, CHECK (Quantite <= 3),
PRIMARY KEY (No_adherent, Livre_id, Date_commande)
);
COMMIT;

```

- Création de références

```
BEGIN TRANSACTION;
```

```

CREATE SEQUENCE adherent_no_adherent_seq
  AS integer
  START WITH 1
  INCREMENT BY 1
  NO MINVALUE
  NO MAXVALUE
  CACHE 1;

```

```
ALTER SEQUENCE adherent_no_adherent_seq OWNED BY Adherent.No_adherent;
```

```

CREATE SEQUENCE livre_livre_id_seq
  AS integer
  START WITH 1
  INCREMENT BY 1
  NO MINVALUE
  NO MAXVALUE
  CACHE 1;

```

```
ALTER SEQUENCE livre_livre_id_seq OWNED BY Livre.Livre_id;
```

```

CREATE SEQUENCE auteurs_auteur_id_seq
  AS integer
  START WITH 1

```

```
INCREMENT BY 1  
NO MINVALUE  
NO MAXVALUE  
CACHE 1;
```

```
ALTER SEQUENCE auteurs_auteur_id_seq OWNED BY Auteurs.Auteur_id;
```

```
ALTER TABLE ONLY Adherent ALTER COLUMN no_adherent SET DEFAULT  
nextval('adherent_no_adherent_seq'::regclass);
```

```
ALTER TABLE ONLY Auteurs ALTER COLUMN Auteur_id SET DEFAULT  
nextval('auteurs_auteur_id_seq'::regclass);
```

```
ALTER TABLE ONLY Livre ALTER COLUMN Livre_id SET DEFAULT  
nextval('livre_livre_id_seq'::regclass);
```

```
ALTER TABLE Livre_Auteur  
ADD CONSTRAINT livre_auteur_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
ALTER TABLE Livre_Auteur  
ADD CONSTRAINT livre_auteur_auteurid  
FOREIGN KEY (Auteur_id)  
REFERENCES Auteurs(Auteur_id);
```

```
ALTER TABLE Genres  
ADD CONSTRAINT genres_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
ALTER TABLE Emprunte  
ADD CONSTRAINT emprunte_adherent  
FOREIGN KEY (No_adherent)  
REFERENCES Adherent(No_adherent);
```

```
ALTER TABLE Emprunte  
ADD CONSTRAINT emprunt_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
ALTER TABLE Commande  
ADD CONSTRAINT commande_adherent  
FOREIGN KEY (No_adherent)
```

```
REFERENCES Adherent(No_adherent);
```

```
ALTER TABLE Commande  
ADD CONSTRAINT commande_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
COMMIT;
```

- Insertion de données

```
BEGIN TRANSACTION;
```

```
INSERT INTO Adherent (Nom, Prenom, Adresse, Phone) VALUES  
(('Seda', 'Carlos', '3066 Boulevard Decarie', '514-555-1234'),  
(('Solis', 'Diego', '456 Chihuahua St', '514-555-2345'),  
(('Galicia', 'Eduardo', '789 Chicago St', '514-555-3456'),  
(('Torres', 'Sofia', '321 Vendome St', '514-555-4567'),  
(('Camargo', 'Eva', '654 Pina St', '514-555-5678'),  
(('Ocean', 'Frank', '987 Justin St', '514-555-6789'),  
(('Johnson', 'Drake', '147 Hillary St', '514-555-7890'),  
(('Camargo', 'Alfonso', '258 Microsoft St', '514-555-8901'),  
(('Solis', 'Ivan', '369 Veracruz St', '514-555-9012'),  
(('Miles', 'Judy', '963 Sinaloa St', '514-555-0123'),  
(('Pena', 'Mali', '741 Juan St', '514-555-1230');
```

```
INSERT INTO Auteurs (Auteur_nom) VALUES  
(('Gabriel Garcia Marquez'),  
(('Antoine de Saint-Exupéry'),  
(('Homero'),  
(('Lewis Carroll'),  
(('Herman Melville'),  
(('Rachael Lippincott'),  
(('Miiki Daughtry'),  
(('Stephen King'),  
(('Owen King'),  
(('John Green'),  
(('Maureen Johnson'),  
(('Lauren Myracle'),  
(('Victor Hugo'),  
(('Guillaume Apollinaire'),  
(('George R.R.Martin');
```

INSERT INTO Livre (Titre) VALUES

('Cien años de soledad'),
('Le petit Prince'),
('Odyssee'),
('Alice au pays des merveilles'),
('Moby Dick'),
('A deux metres de toi'),
('Belles endormies'),
('Nuit blanche'),
('Les Miserables'),
('Calligrammes'),
('Game of Thrones'),
('Amour aux temps du choléra'),
('Chronique dune mort annoncee'),
('Dolores Claiborne');

INSERT INTO Genres (Livre_Id, Genre_nom) VALUES

(1, 'Science-fiction'),
(2, 'Fantasy'),
(3, 'Romance'),
(4, 'Romance'),
(5, 'Policier'),
(6, 'Fantasy'),
(7, 'Biographie'),
(8, 'Histoire'),
(9, 'Science-fiction'),
(10, 'Science-fiction'),
(11, 'Science-fiction'),
(12, 'Romance'),
(13, 'Fantasy'),
(14, 'Science-fiction');

INSERT INTO Livre_Auteur (Livre_id, Auteur_id) VALUES

(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(12, 1),
(7, 8),
(8, 10),

(13, 1),
(14, 12),
(9, 13),
(10, 14),
(11, 15);

INSERT INTO Emprunte (No_adherent, Livre_id, Date_emprunt, Date_retour) VALUES
(1, 1, '2022-01-15', '2022-01-29'),
(2, 2, '2022-02-20', '2022-03-06'),
(3, 3, '2022-03-15', '2022-03-29'),
(4, 4, '2022-04-30', '2022-05-14'),
(5, 1, '2022-06-01', '2022-06-15'),
(6, 6, '2022-06-20', '2022-07-04'),
(7, 7, '2022-07-25', '2022-08-08'),
(8, 8, '2022-08-15', '2022-08-29'),
(9, 1, '2022-09-10', '2022-09-24'),
(10, 2, '2022-10-25', '2022-11-08'),
(11, 11, '2022-11-20', '2022-12-04'),
(1, 12, '2023-04-01', '2023-04-29'),
(5, 13, '2023-03-02', '2023-04-30');

INSERT INTO Commande (No_adherent, Livre_id, Date_commande, Quantite) VALUES
(1, 8, '2022-01-10', 1),
(2, 3, '2022-02-12', 3),
(3, 9, '2022-03-05', 2),
(4, 5, '2022-04-20', 1),
(5, 6, '2022-05-28', 2),
(6, 7, '2022-06-10', 3),
(7, 8, '2022-07-18', 1),
(8, 9, '2022-08-10', 1),
(9, 2, '2022-09-01', 1),
(10, 11, '2023-01-15', 2),
(11, 8, '2023-02-10', 1);

COMMIT;

5. Question/Réponse

Q1. La bibliothèque souhaite pouvoir connaître à tout moment la situation de chaque adhérent (Liste de livres empruntés avec retards éventuels)?

Algèbre relationnelle:

$r1 \leftarrow \sigma_{(\text{EXTRACT}(\text{DAY FROM COALESCE}(\text{Date_retour}, \text{NOW}())) - \text{Date_emprunt}) > 14}(\text{Emprunte})$

$r2 \leftarrow \pi_{\text{Prénom}, \text{Nom}, \text{Date-emprunt}, \text{Titre}, \text{Now}(), \text{DateDifference}}(r1 \bowtie \text{Adhérent} \bowtie \text{Livre})$

SQL1:

```
SELECT
    CONCAT (aa.Prenom, ' ', aa.Nom) Adherent,
    ee.Date_emprunt,
    ll.Titre,
    NOW(),
    EXTRACT(DAY FROM COALESCE(ee.Date_retour, NOW()) - ee.Date_emprunt) AS
DateDifference
FROM Emprunte ee, Livre ll, Adherent aa WHERE ee.Livre_id = ll.Livre_id AND
aa.No_adherent = ee.No_adherent
AND EXTRACT(DAY FROM COALESCE(ee.Date_retour, NOW()) - ee.Date_emprunt) > 14
```

SQL2:

```
SELECT aa.Nom, aa.Prenom, ll.Titre
FROM Adherent aa,
Livre ll,
(SELECT ee.No_adherent, ee.Livre_id
FROM Emprunte ee
WHERE EXTRACT(DAY FROM
COALESCE(ee.Date_retour, NOW()) - ee.Date_emprunt) > 14) Returned_delay
WHERE aa.No_adherent = returned_delay.No_adherent and ll.Livre_id =
Returned_delay.Livre_id
ORDER BY aa.Nom, aa.Prenom
```

Q2. La bibliothèque souhaite également pouvoir collecter des statistiques sur la pratique des abonnés (nombre de livre empruntés) ?

Algèbre relationnelle:

$r1 \leftarrow \rho_{(\text{No-adherent}, \text{Books_borrowed})}(\text{No-adherent} \mathcal{A}_{\text{COUNT}(\text{No-adherent})}(\text{Emprunte}))$

$r2 \leftarrow \pi_{\text{No_adherent}, \text{Books_borrowed}, (\sigma_{(\text{Adherent.Nom} < \text{ALT.Nom})}(\text{Adherent} \times (\rho_{(\text{ALT})}(\text{Adherent})))), (\sigma_{(\text{Adherent.Prénom} < \text{ALT.Prénom})}$

$(\text{Adherent} \times (\rho_{(\text{ALT})}(\text{Adherent}))))(r1 \bowtie \text{Adhérent})$

$/*\text{MAX}(\text{aa.Nom}) = (\sigma_{(\text{Adherent.Nom} < \text{ALT.Nom})}(\text{Adherent} \times (\rho_{(\text{ALT})}(\text{Adherent}))))*/$

SQL1:

```
SELECT ee.No_adherent, COUNT(ee.No_adherent) Books_borrowed, MAX(aa.Nom),
MAX(aa.Prenom) FROM Emprunte ee, Adherent aa
WHERE ee.No_adherent = aa.No_adherent
GROUP BY ee.No_adherent
```

SQL2:

```
SELECT aa.*, gg.Books_borrowed
FROM Adherent aa INNER JOIN
    (SELECT ee.No_adherent, COUNT(ee.No_adherent) Books_borrowed
    from Emprunte ee GROUP BY ee.No_adherent) gg
ON aa.No_adherent = gg.No_adherent
```

Q3. La bibliothèque souhaite aussi pouvoir faire des statistiques sur la pratique des abonnés (nombre de livres empruntés par an)?

Algèbre relationnelle:

$r1 \leftarrow \pi_{\text{Nom, Prénom, Date-emprunt.Year, (Adhérent.No-adhérent, Date-emprunt.Year } \mathcal{A}_{\text{COUNT(*)}} \text{ (Adhérent } \bowtie_L$
Emprunte)

$r2 \leftarrow \rho_{\text{Nom, Prénom, yyyy, Number_of_books}} (r1)$

SQL:

```
SELECT aa.Nom, aa.Prenom,
EXTRACT(YEAR FROM ee.Date_emprunt) yyyy, COUNT(*) Number_of_books
FROM Adherent aa LEFT JOIN Emprunte ee ON aa.No_adherent = ee.No_adherent
GROUP BY aa.No_adherent, EXTRACT(YEAR FROM ee.Date_emprunt)
ORDER BY EXTRACT(YEAR FROM ee.Date_emprunt)
```

Q4. La bibliothèque souhaite aussi pouvoir faire des statistiques sur la pratique des abonnés (répartition des emprunts par genre)?

Algèbre relationnelle:

$r1 \leftarrow \pi_{\text{Nom, Prénom, (Adhérent.No-adhérent, Genre-nom } \mathcal{A}_{\text{COUNT(*)}} \text{ (Adhérent } \bowtie \text{ Emprunte } \bowtie \text{ Livre } \bowtie$
Genres)

$r2 \leftarrow \rho_{\text{Nom, Prénom, Genre-nom, Borrowings_per_genre}} (r1)$

SQL:

```
SELECT aa.Nom, aa.Prenom, gg.Genre_nom, COUNT(*) Borrowings_per_genre FROM
Emprunte ee, Livre ll, Adherent aa, Genres gg
WHERE ee.Livre_id = ll.Livre_id AND aa.No_adherent = ee.No_adherent AND gg.Livre_id
= ll.Livre_id
GROUP BY aa.No_adherent, gg.Genre_nom
ORDER BY count(*) DESC
```

Q5. La bibliothèque souhaite aussi pouvoir faire des statistiques sur la pratique des abonnés (nombre d'emprunts par livre)?**Algèbre relationnelle:**

$$r1 \leftarrow \pi_{\text{Nom, Prénom, Titre, (Adhérent.No-adhérent, Livre-id)}} \mathcal{A}_{\text{COUNT(*)}} (\text{Emprunte} \bowtie \text{Livre} \bowtie \text{Adhérent})$$

$$r2 \leftarrow \rho_{(\text{Nom, Prénom, Titre, Borrowings_per_book})} (r1)$$
SQL:

```
SELECT aa.Nom, aa.Prenom, ll.Titre, COUNT(*) Borrowings_per_book FROM Emprunte
ee, Livre ll, Adherent aa
WHERE ee.Livre_id = ll.Livre_id AND aa.No_adherent = ee.No_adherent
GROUP BY aa.No_adherent, ll.Livre_id
ORDER BY COUNT(*) DESC
```

- Optimisation:

Pour optimiser les requêtes, on a effectué autant que possible les opérations dans l'ordre suivante pour les questions 1 à 5 :

1. Sélections
2. Projections
3. Jointures et Produits

- Déclencheurs

Dans l'énoncé du projet il y a des exigences qu'on peut implémenter en utilisant de déclencheurs:

- 1- "les abonnés peuvent commander des livres. Ils peuvent en commander trois au maximum".

```
/* Max 3 commandes */
```

```
CREATE OR REPLACE FUNCTION enforce_max_3_command() RETURNS trigger AS $$
DECLARE
    max_command_count INTEGER := 3;
```

```

Quantite INTEGER := 0;
must_check BOOLEAN := false;
BEGIN
  IF TG_OP = 'INSERT' THEN
    must_check := true;
  END IF;

  IF TG_OP = 'UPDATE' THEN
    IF (NEW.owner != OLD.owner) THEN
      must_check := true;
    END IF;
  END IF;

  IF must_check THEN
    -- prevent concurrent inserts from multiple transactions
    LOCK TABLE Commande IN EXCLUSIVE MODE;

    SELECT INTO Quantite COUNT(*)
    FROM Commande
    WHERE owner = NEW.owner;

    IF Quantite >= max_command_count THEN
      RAISE EXCEPTION 'Cannot insert more than % commande for each user.',
max_command_count;
    END IF;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER enforce_max_3_command
  BEFORE INSERT OR UPDATE ON Commande
  FOR EACH ROW EXECUTE PROCEDURE enforce_max_3_command();
2- “Une commande peut être annulée ou honorée si le livre commandé a finalement été
emprunté”.

CREATE OR REPLACE FUNCTION cancel_commande()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
  if (EXISTS(SELECT * FROM Commande WHERE Livre_id = NEW.Livre_id AND
No_adherent = NEW.No_adherent) )THEN

```

```
        DELETE FROM Commande WHERE Livre_id = NEW.Livre_id AND No_adherent
= NEW.No_adherent;
    END IF;
END;
$$
```

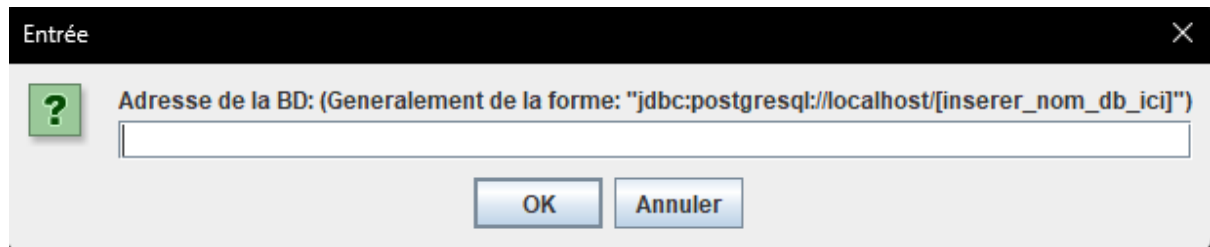
```
CREATE TRIGGER check_cancel_commande
    AFTER INSERT ON Emprunte
    FOR EACH ROW
    /* Action if condition true */
    EXECUTE FUNCTION cancel_commande();
```

6. Interaction avec un langage 3G (JAVA)

Etape 1: Se connecter a la database:

Si c'est une database local de pgadmin, l'adresse sera surement de la forme:

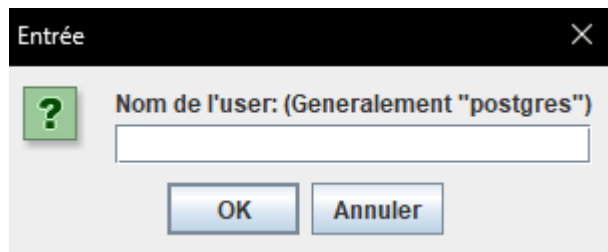
`jdbc:postgresql://localhost/[insérer_nom_db_ici]`



A screenshot of a Java dialog box titled 'Entrée' with a close button (X) in the top right corner. On the left is a green square icon with a white question mark. The main text reads 'Adresse de la BD: (Généralement de la forme: "jdbc:postgresql://localhost/[insérer_nom_db_ici]")'. Below the text is a single-line text input field. At the bottom are two buttons: 'OK' and 'Annuler'.

Ensuite, insérer le nom de l'utilisateur:

Sur pgadmin, l'utilisateur par défaut est: postgres



A screenshot of a Java dialog box titled 'Entrée' with a close button (X) in the top right corner. On the left is a green square icon with a white question mark. The main text reads 'Nom de l'utilisateur: (Généralement "postgres")'. Below the text is a single-line text input field. At the bottom are two buttons: 'OK' and 'Annuler'.

Puis le mot de passe de l'utilisateur:

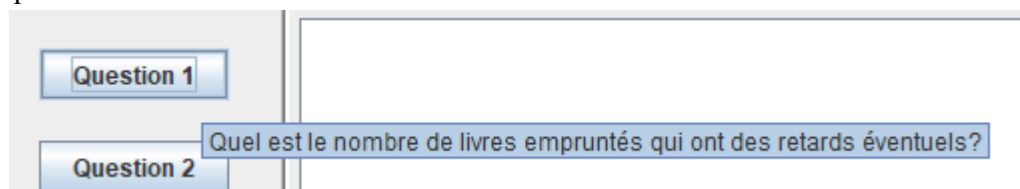


A screenshot of a Java dialog box titled 'Entrée' with a close button (X) in the top right corner. On the left is a green square icon with a white question mark. The main text reads 'Password de l'utilisateur:'. Below the text is a single-line text input field. At the bottom are two buttons: 'OK' and 'Annuler'.

Si la connexion à la DB échoue, c'est que les informations que vous avez entrées sont fausses, le programme vous redemandera à nouveau de vous connecter.

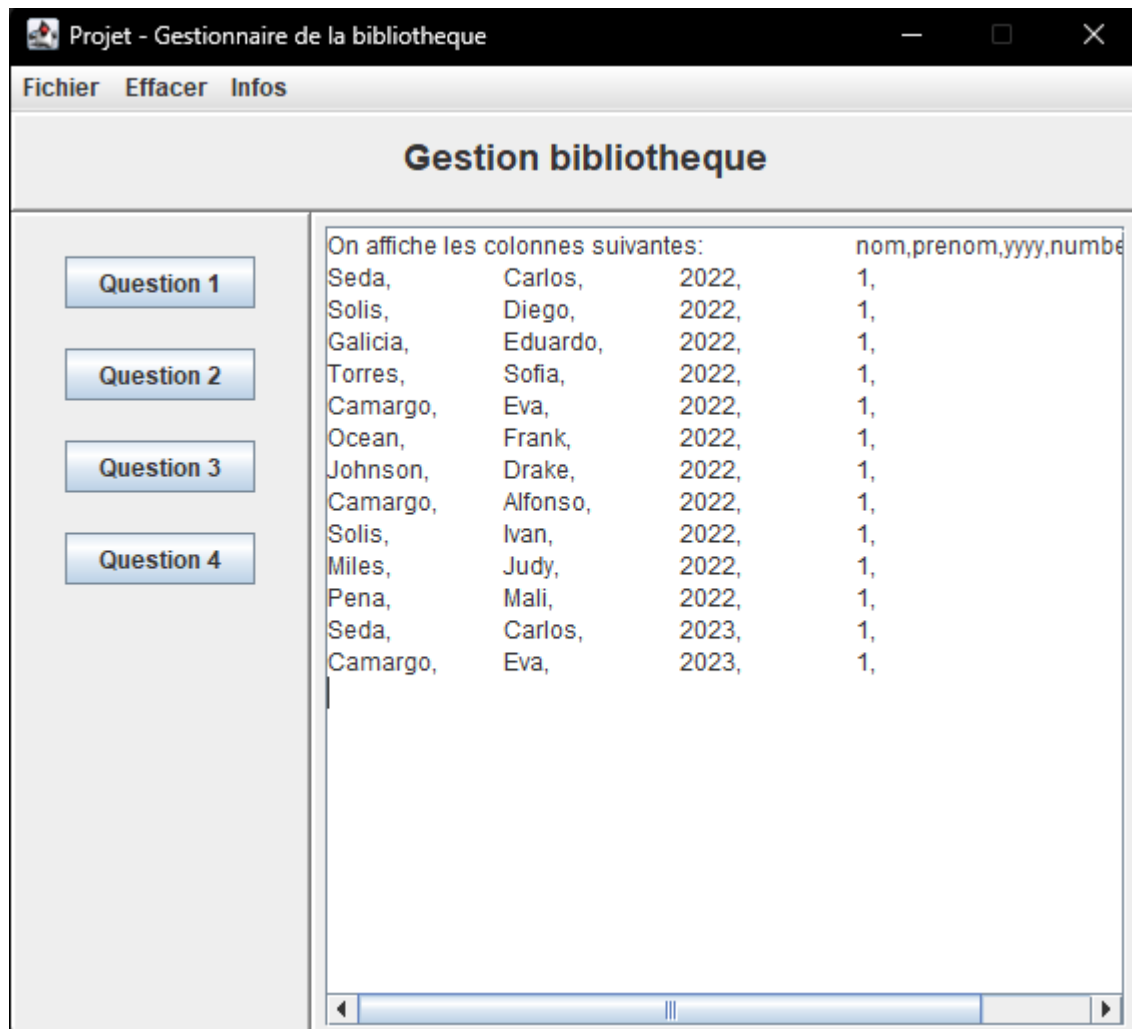
Etape 2: Utilisation

Une fois connecté à la base, vous pouvez passer la souris au-dessus de n'importe quelle question afin de savoir son contenu:



A screenshot of a user interface showing a list of questions. On the left, there are two buttons labeled 'Question 1' and 'Question 2'. To the right of these buttons is a large text area. A tooltip is visible over the text area, containing the text 'Quel est le nombre de livres empruntés qui ont des retards éventuels?'.

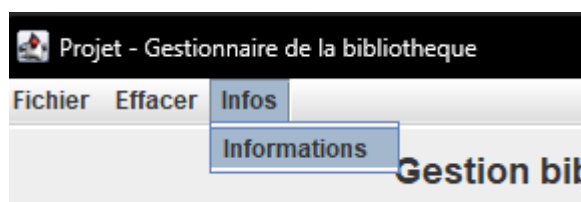
En cliquant sur une des questions, la zone de texte à droite contiendra la réponse de celle-ci, la première ligne correspondant aux colonnes qui seront affichées, puis les lignes suivantes correspondent au résultat du query.



Pour effacer le contenu de la zone texte, cliquez dans le menu en haut sur Effacer, puis Effacer le contenu

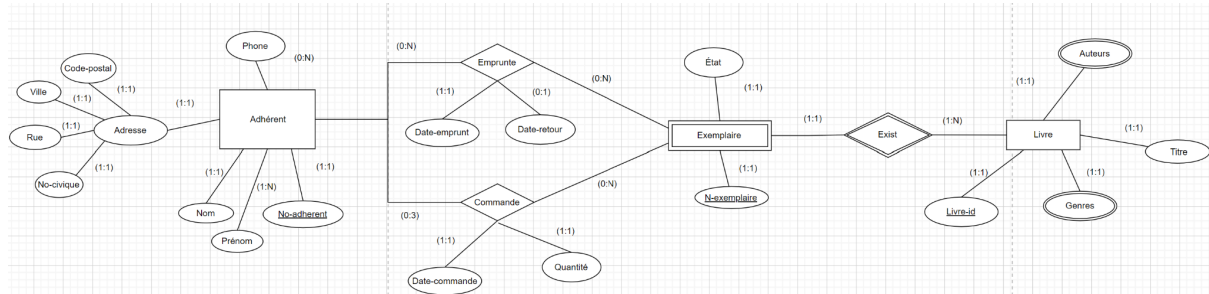


Puis finalement pour avoir des infos sur le programme, cliquez sur Infos dans le menu:



Solution Alternative(Pas Etapes 5 et 6)

1. La modélisation



2. La transformation

- + **Adhèrent**(No-adhèrent, Nom, Prénom, Adresse, Phone)
- + **Livre**(Livre-id, Titre, Genres, Auteurs)
- + **Auteurs** (#Livre-id, Auteur-nom)
- + **Genres** (#Livre-id, Genre-nom)
- + **Exemple** (No-exemple, #Livre-id, État)
- + **Existe**(#Livre-id, #No-exemple) // On n'a pas besoin de cette relation
- + **Emprunte**(#No-adhèrent, #No-exemple, Date-emprunt, Date-retour)
- + **Commande**(#No-adhèrent, #No-exemple, Date-commande, Quantité)

Pour **Auteurs** la clé étrangère est #Livre-id et la clé primaire est {#Livre-id, Auteur-nom}.

Pour **Genres** la clé étrangère est #Livre-id et la clé primaire est {#Livre-id, Genre-nom}.

Pour **Exemple** la clé étrangère est #Livre-id et la clé primaire est {#Livre-id, No-exemple}.

Pour **Emprunte** la clé étrangère est {#No-adhèrent, #No-exemple} et la clé primaire est {#No-adhèrent, #No-exemple, Date-emprunt}.

Pour **Commande** la clé étrangère est {#No-adhèrent, #No-exemple} et la clé primaire est {#No-adhèrent, #No-exemple, Date-commande}.

3. La normalisation

- Dépendances fonctionnelles pour chaque relation:

- **Adhérent:**

$(A \rightarrow B, C, D, E)$

No-adhérent \rightarrow Nom, Prénom, Adresse, Phone

Un adhérent spécifique a un nom, prénom, une adresse et un numéro de téléphone uniques.

- **Livre:**

$(F \rightarrow G, H, I)$

Livre-id \rightarrow Titre, Genre, Auteur

Le Livre-id est unique pour chaque livre et détermine ses autres attributs.

- **Auteurs:**

$(F, J \rightarrow \text{Aucun})$

Livre-id, Auteur-nom \rightarrow Aucun

La combinaison de Livre-id et auteur-nom est unique.

- **Genres:**

$(F, K \rightarrow \text{Aucun})$

Livre-id, Genre-nom \rightarrow Aucun

La combinaison de Livre-id et genre-nom est unique.

- **Exemplaire:**

$(F, L \rightarrow M)$

Livre-id, No-exemplaire \rightarrow État

La combinaison de Livre-id et No-exemplaire est unique pour chaque exemplaire et détermine son état.

- **Emprunte:**

$(A, L, N \rightarrow O)$

No-adhérent, Livre-id, Date-emprunt \rightarrow Date-retour

La combinaison de No-adhérent, Livre-id et Date-emprunt est unique et détermine la Date-retour.

- **Commande:**

$(A, L, P \rightarrow Q)$

No-adhérent, No-exemplaire, Date-commande \rightarrow Quantité

La combinaison de No-adhérent, No-exemplaire et Date-commande est unique et détermine la Quantité.

- Normalisation des tables:

- **Adhérent** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Livre** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Genres** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.

Dans ce cas, la table Genres possède un attribut de clé primaire composé de deux attributs : Livre-id et Genre-nom. Puisqu'il s'agit d'une clé primaire, chaque combinaison de valeurs Livre-id et Genre-nom est garantie d'être unique dans la table, il n'y aurait donc pas de problèmes de redondance.

Il est vrai qu'il peut y avoir plusieurs livres d'un même genre, mais chacun d'eux aurait un identifiant unique dans la table Livre, ce qui garantit l'unicité de la valeur Livre-id dans la table Genres. Par conséquent, en incluant les deux attributs dans la clé primaire de la table Genres, nous nous assurons qu'il n'y aura pas de doublons dans la table et qu'elle est conforme à la forme normale Boyce-Codd (BCNF).

- **Auteurs** : Ce tableau n'est pas normalisé puisque dans la bibliothèque vous pouvez avoir plusieurs livres du même auteur (ou des livres qui ont plusieurs auteurs, si on considère l'attribut **Auteur** comme multivalué), donc dans le tableau nous aurons des répétitions d'auteurs, par exemple :

Livre-id	Auteur-nom
1	J. K. Rowling
2	J. K. Rowling

Pour résoudre cela et pouvoir normaliser ce tableau nous pouvons diviser le tableau **Auteurs** en deux tables: une pour les auteurs et une pour la relation entre les livres et les auteurs. Cela permettra à plusieurs auteurs d'être associés à un livre et vice versa:

- AuteurNom(Auteur-id, Auteur-nom)
- Auteurs(#Livre-id, #Auteur-id)
- **Exemplaire** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Commande** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.
- **Emprunte** : Cette table est déjà en BCNF car la clé primaire (super clé) détermine tous les autres attributs et il n'y a pas de dépendances transitives.

- Le schéma final de la base de données obtenu:

- + Adhérent(No-adhérent, Nom, Prénom, Adresse, Phone)
- + Livre(Livre-id, Titre)
- + AuteurNom(Auteur-id, Auteur-nom)
- + Auteurs(#Livre-id, #Auteur-id)
- + Genres(#Livre-id, Genre-nom)
- + Exemplaire(#Livre-id, No-exemplaire, État)
- + Emprunte(#No-adhérent, #No-exemplaire, Date-emprunt, Date-retour)
- + Commande(#No-adhérent, #No-exemplaire, Date-commande, Quantité)

4. L'implémentation

Création des tableaux

```
BEGIN TRANSACTION;  
  
CREATE TABLE Adherent (  
    No_adherent INTEGER PRIMARY KEY,  
    Nom VARCHAR(255) NOT NULL,  
    Prenom VARCHAR(255) NOT NULL,  
    Adresse VARCHAR(255) NOT NULL,  
    Phone VARCHAR(15) NOT NULL  
);
```

```
CREATE TABLE Livre (  
    Livre_id INTEGER PRIMARY KEY,  
    Titre VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE Auteursnom (  
    Auteur_id INTEGER PRIMARY KEY,  
    Auteur_nom VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE Auteurs (  
Livre_id INTEGER NOT NULL,  
Auteur_id INTEGER NOT NULL,  
PRIMARY KEY (Livre_id, Auteur_id)  
);
```

```
CREATE TABLE Genres (  
Livre_id INTEGER NOT NULL,  
Genre_nom VARCHAR(255) NOT NULL,  
PRIMARY KEY (Livre_id, Genre_nom)  
);
```

```
CREATE TABLE Exempleire (  
No_exempleire INTEGER NOT NULL,  
Livre_id INTEGER NOT NULL,  
Etat VARCHAR(255) NOT NULL,  
PRIMARY KEY (Livre_id, No_exempleire)  
);
```

```
CREATE TABLE Emprunte (  
No_adherent INTEGER NOT NULL,  
No_exempleire INTEGER NOT NULL,  
Date_emprunt DATE NOT NULL,  
Date_retour DATE,  
PRIMARY KEY (No_adherent, No_exempleire, Date_emprunt)  
);
```

```
CREATE TABLE Commande (  
No_adherent INTEGER NOT NULL,
```

```
No_exemplaire INTEGER NOT NULL,  
Date_commande DATE NOT NULL,  
Quantite INTEGER NOT NULL,  
PRIMARY KEY (No_adherent, No_exemplaire, Date_commande)  
);  
  
COMMIT;
```

Création de références

```
BEGIN TRANSACTION;  
  
CREATE SEQUENCE adherent_no_adherent_seq  
  AS integer  
  START WITH 1  
  INCREMENT BY 1  
  NO MINVALUE  
  NO MAXVALUE  
  CACHE 1;  
  
ALTER SEQUENCE adherent_no_adherent_seq OWNED BY  
Adherent.No_adherent;  
  
CREATE SEQUENCE livre_livre_id_seq  
  AS integer  
  START WITH 1  
  INCREMENT BY 1  
  NO MINVALUE  
  NO MAXVALUE  
  CACHE 1;  
  
ALTER SEQUENCE livre_livre_id_seq OWNED BY Livre.Livre_id;  
  
CREATE SEQUENCE auteurnom_auteur_id_seq  
  AS integer  
  START WITH 1  
  INCREMENT BY 1  
  NO MINVALUE
```

```
NO MAXVALUE  
CACHE 1;
```

```
ALTER SEQUENCE auteurnom_auteur_id_seq OWNED BY Auteurnom.Auteur_id;
```

```
CREATE SEQUENCE exemplaire_no_exemplaire_seq  
  AS integer  
  START WITH 1  
  INCREMENT BY 1  
  NO MINVALUE  
  NO MAXVALUE  
  CACHE 1;
```

```
ALTER SEQUENCE exemplaire_no_exemplaire_seq OWNED BY  
Exemplaire.No_exemplaire;
```

```
ALTER TABLE ONLY Adherent ALTER COLUMN no_adherent SET DEFAULT  
nextval('adherent_no_adherent_seq'::regclass);
```

```
ALTER TABLE ONLY Auteurnom ALTER COLUMN Auteur_id SET DEFAULT  
nextval('auteurnom_auteur_id_seq'::regclass);
```

```
ALTER TABLE ONLY Livre ALTER COLUMN Livre_id SET DEFAULT  
nextval('livre_livre_id_seq'::regclass);
```

```
ALTER TABLE ONLY Exemplaire ALTER COLUMN No_exemplaire SET  
DEFAULT nextval('exemplaire_no_exemplaire_seq'::regclass);
```

```
ALTER TABLE Exemplaire ADD CONSTRAINT unique_No_exemplaire UNIQUE  
(No_exemplaire);
```

```
ALTER TABLE Auteurs  
ADD CONSTRAINT Auteurs_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
ALTER TABLE Auteurs  
ADD CONSTRAINT Auteurs_auteurid  
FOREIGN KEY (Auteur_id)  
REFERENCES Auteurnom(auteur_id);
```

```
ALTER TABLE Genres  
ADD CONSTRAINT genres_livreid  
FOREIGN KEY (livre_id)
```

```
REFERENCES livre(livre_id);
```

```
ALTER TABLE Exemple  
ADD CONSTRAINT Exemple_livreid  
FOREIGN KEY (Livre_id)  
REFERENCES Livre(Livre_id);
```

```
ALTER TABLE Emprunte  
ADD CONSTRAINT Emprunte_adherent  
FOREIGN KEY (No_adherent)  
REFERENCES Adherent(No_adherent);
```

```
ALTER TABLE Emprunte  
ADD CONSTRAINT Emprunte_noexemple  
FOREIGN KEY (No_exemple)  
REFERENCES Exemple(No_exemple);
```

```
ALTER TABLE Commande  
ADD CONSTRAINT Commande_adherent  
FOREIGN KEY (No_adherent)  
REFERENCES Adherent(No_adherent);
```

```
ALTER TABLE Commande  
ADD CONSTRAINT Commande_noexemple  
FOREIGN KEY (No_exemple)  
REFERENCES Exemple(No_exemple);
```

```
COMMIT;
```

Insertion de données

```
BEGIN TRANSACTION;
```

```
INSERT INTO Adherent (Nom, Prenom, Adresse, Phone) VALUES  
(('Seda', 'Carlos', '3066 Boulevard Decarie', '514-555-1234'),  
(('Solis', 'Diego', '456 Chihuahua St', '514-555-2345'),  
(('Galicia', 'Eduardo', '789 Chicago St', '514-555-3456'),  
(('Torres', 'Sofia', '321 Vendome St', '514-555-4567'),  
(('Camargo', 'Eva', '654 Pina St', '514-555-5678'),  
(('Ocean', 'Frank', '987 Justin St', '514-555-6789'),  
(('Johnson', 'Drake', '147 Hillary St', '514-555-7890'),  
(('Camargo', 'Alfonso', '258 Microsoft St', '514-555-8901'),
```

('Solis', 'Ivan', '369 Veracruz St', '514-555-9012'),
('Miles', 'Judy', '963 Sinaloa St', '514-555-0123'),
('Pena', 'Mali', '741 Juan St', '514-555-1230');

INSERT INTO Auteurs (Auteur_nom) VALUES
('Gabriel Garcia Marquez'),
('Antoine de Saint-Exupéry'),
('Homero'),
('Lewis Carroll'),
('Herman Melville'),
('Rachael Lippincott'),
('Miiki Daughtry'),
('Stephen King'),
('Owen King'),
('John Green'),
('Maureen Johnson'),
('Lauren Myracle'),
('Victor Hugo'),
('Guillaume Apollinaire'),
('George R.R.Martin');

INSERT INTO Livre (Titre) VALUES
('Cien años de soledad'),
('Le petit Prince'),
('Odyssee'),
('Alice au pays des merveilles'),
('Moby Dick'),
('A deux metres de toi'),
('Belles endormies'),
('Nuit blanche'),
('Les Miserables'),
('Calligrammes'),
('Game of Thrones'),
('Amour aux temps du choléra'),
('Chronique d'une mort annoncée'),
('Dolores Claiborne');

INSERT INTO Auteurs (Livre_id, Auteur_id) VALUES
(1, 1),
(2, 2),
(3, 3),

(4, 4),
(5, 5),
(6, 6),
(12, 1),
(7, 8),
(8, 10),
(13, 1),
(14, 12),
(9, 13),
(10, 14),
(11, 15);

INSERT INTO Genres (Livre_id, Genre_nom) VALUES

(1, 'Romance'),
(2, 'Fantaisie'),
(3, 'Aventure'),
(4, 'Fantaisie'),
(5, 'Aventure'),
(6, 'Romance'),
(12, 'Romance'),
(7, 'Romance'),
(8, 'Romance'),
(13, 'Romance'),
(14, 'Romance'),
(9, 'Romance'),
(10, 'Romance'),
(11, 'Romance');

INSERT INTO Exemplaire (Livre_id, Etat) VALUES

(1, 'Disponible'),
(1, 'Disponible'),
(1, 'Disponible'),
(1, 'Disponible'),
(2, 'Disponible'),
(2, 'Disponible'),
(2, 'Emprunte'),
(3, 'Disponible'),
(3, 'Disponible'),
(3, 'Emprunte'),
(3, 'Disponible'),

(4, 'Disponible'),
(4, 'Disponible'),
(4, 'Disponible'),
(4, 'Emprunte'),
(5, 'Disponible'),
(5, 'Disponible'),
(5, 'Emprunte');

INSERT INTO Emprunte (No_adherent, No_exemplaire, Date_emprunt,
Date_retour) VALUES

(1, 1, '2022-01-15', '2022-01-29'),
(2, 2, '2022-02-20', '2022-03-06'),
(3, 3, '2022-03-15', '2022-03-29'),
(4, 4, '2022-04-30', '2022-05-14'),
(5, 1, '2022-06-01', '2022-06-15'),
(6, 6, '2022-06-20', '2022-07-04'),
(7, 7, '2022-07-25', '2022-08-08'),
(8, 8, '2022-08-15', '2022-08-29'),
(9, 1, '2022-09-10', '2022-09-24'),
(10, 2, '2022-10-25', '2022-11-08'),
(11, 11, '2022-11-20', '2022-12-04'),
(1, 12, '2023-04-01', '2023-04-29'),
(5, 13, '2023-03-02', '2023-04-30');

INSERT INTO Commande (No_adherent, No_exemplaire, Date_commande,
Quantite) VALUES

(1, 8, '2022-01-10', 1),
(2, 3, '2022-02-12', 3),
(3, 9, '2022-03-05', 2),
(4, 5, '2022-04-20', 1),
(5, 6, '2022-05-28', 2),
(6, 7, '2022-06-10', 3),
(7, 8, '2022-07-18', 1),
(8, 9, '2022-08-10', 1),
(9, 2, '2022-09-01', 1),
(10, 11, '2023-01-15', 2),
(11, 8, '2023-02-10', 1);

COMMIT;