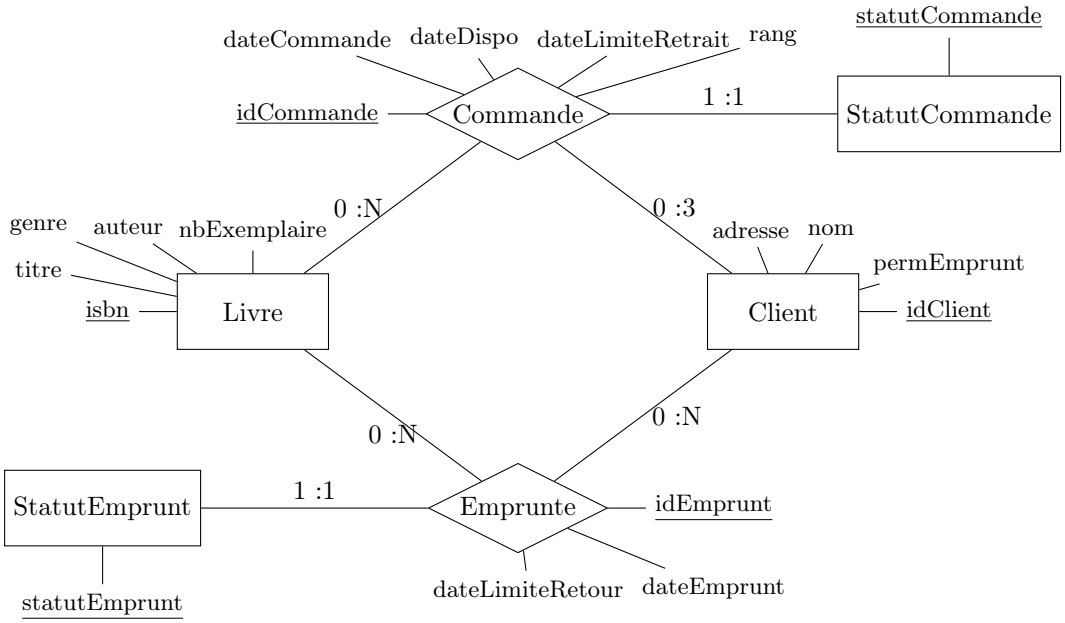
# Projet

Alexandre Toutant 20028191 - Nom2 - nom3 -nom4

17 avril 2025

## 1 Modélisation en modèle E/A




Explications :

- **isbn et ids** : Ajoutés afin d'identifier facilement les différents éléments.
- **Livre.nbExemplaire** : Représente le nombre d'exemplaires disponibles pour être emprunté. Par exemple, disons que la bibliothèque possède deux exemplaires du *Multidictionnaire de la langue française*, et qu'un client à une commande ou emprunte directement ce livre, le nomnbre d'exemplaires disponibles sera diminué. Ceci explique aussi la cardinalité de la relation entre **Livre** et **Emprunte**, puisqu'un livre avec plusieurs exemplaires peut donc être emprunté plusieurs fois en même temps.
- **Commande.rang** : Représente le rang de cette commande parmi toutes les commandes sur le même livre. Disons que deux clients créent une commande pour le *Multidictionnaire de la langue française* alors qu'il n'y a qu'un seul exemplaire dans la bibliothèque, les commandes seront traitées selon la politique "première arrivée, première servie" (*FIFO*). Le rang permet de savoir combien de commandes sont devant la notre.
- **Commande.dateCommande** : Représente la date où la commande fut émise, dans le but d'ordonner la file d'attente.
- **Commande.dateDispo et Commande.dateLimiteRetrait** : Utilent pour avoir un interval où le livre est réservé au client ayant fait cette commande. Ainsi, lorsque le livre commandé est disponible, ces deux attributs sont mis à jour. Si la date de limite de retrait est dépassée, alors la commande devient expirée et la priorité est donnée à la prochaine commande.

— **StatutCommande.statutCommande** : Permet de faire l'énumération des statuts possibles pour une commande. Les statuts possibles pour une commande sont les suivants :
  - En attente : La commande est passée, mais le client n'a pas encore la priorité sur ce document.
  - Disponible pour le client : Le client à la priorité pour réserver ce document, pour une fenêtre de deux jours.
  - Expirée : Le client avait la priorité pour réserver ce document, mais celui-ci a trop attendu. Il pert donc sa priorité et la commande n'est plus active.
  - Annulée : Le client à décidé d'annuler la commander de son côté ; la commande n'est plus active.
  - Honorée : Le client à réservé ce document. La commande n'est plus active.
— **Client.permEmprunt** : Permet de vérifier si un client peut emprunter ou non. Par exemple, nous pourrions décider qu'un client ayant deux retards actifs serait interdit d'emprunter un troisième document jusqu'au règlement du retard.
— **Client.adresse** : Afin de sauver de l'espace, nous n'avons pas représenté l'adresse comme étant un attribut complexe. Par contre, nous le décomposeront dans le modèle relationnel.
— **StatutEmprunt.statutEmprunt** : Permet de faire l'énumération des statuts possibles pour un emprunt. Ceux-ci permettent de faire des statistiques sur les pratiques des clients. Les statuts possibles pour un emprunt sont les suivants :
  - Retourné : Un emprunt est terminé, sans retard ; celui-ci n'est plus actif et le livre redevient disponible.
  - Réglé : Un emprunt était en retard, mais le client à réglé ce problème. L'emprunt n'est plus actif et le livre redevient disponible.
  - En retard : Un emprunt est en retard ; une action du client est nécessaire. Le livre est encore en possession du client.
  - En cours : Un emprunt est en cours, sans retard. Le livre est encore en possession du client.

# 2 Transformation en modèle relationnel

Explications :

— Pour commencer, toutes les entités faisant parties du modèle E/A seront conservées comme des tables différentes dans le modèle relationnel.
— Puisque nous n'avons pas de relation 2D 1 :1, les associations présentes dans le modèle E/A seront aussi présentes dans le modèle relationnel, tout en ayant des clés étrangères pour lier les entités concernées.
— Afin de permettre d'avoir un client avec plusieurs adresses, et qu'une adresse puisse être partagée par plusieurs clients, nous avons créé les table **AdresseClient** et **Adresse**.
— Afin de nous faciliter la vie plus tard avec la gestion des rangs des commandes, nous avons créé la table **RangCommande**.

Voici le modèle relationnel correspondant :

- Client(idClient, permEmprunt, nom)
- AdresseClient(#idClient, #idAdresse)
- Adresse(idAdresse, rue, ville, code, pays)
- Livre(isbn, titre, genre, auteur, nbExemplaire)
- StatutEmprunt(statutEmprunt)
- StatutCommande(statutCommande)
- Commande(idCommande, #isbn, #idClient, #statutCommande, dateCommande, dateDispo, dateLimiteRetrait)
- RangCommande(#idCommande, #isbn, rang)
- Emprunte(idEmprunt, #isbn, #idClient, #statutEmprunt, dateEmprunt, dateLimiteRetour)

# 3 Normalisation

Pour la normalisation, nous normaliserons en 3NF. Pour les dépendances fonctionnelles, nous laissons de côté les dépendances triviales.

## 3.1 Client(idClient, permEmprunt, nom)

### 3.1.1 Dépendances fonctionnelles

Chaque client à un nom et potentiellement le droit d'emprunter des livres. Donc, l'identifiant client permet de déterminer les deux autres.

$$idClient \rightarrow permEmprunt,\ nom$$

### 3.1.2 3NF

Comme nous avons déjà séparé **adresse** en d'autres relation, cette relation est en 1NF ; tous les attributs contiennent une valeur atomique (1NF). Tout attribut non clé ne dépend pas d'une partie de la clé, donc cette relation est 2NF. Tout attribut non clé dépend de la clé, donc cette relation est 3NF.

## 3.2 AdresseClient(#idClient, #idAdresse)

### 3.2.1 Dépendances fonctionnelles

Aucune dépendance fonctionnelle non triviale ici, puisque qu'il n'y a pas d'attribut non-clé.

### 3.2.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.3 Adresse(idAdresse, rue, ville, code, pays)

### 3.3.1 Dépendances fonctionnelles

Chaque adresse à un identifiant unique permettant de retrouver ses détails. Donc, cet identifiant permet de retrouver de manière unique les autres attributs.

$$idAdresse \rightarrow rue,\ ville,\ code,\ pays$$

### 3.3.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.4 Livre(isbn, titre, genre, auteur, nbExemplaire)

### 3.4.1 Dépendances fonctionnelles

Chaque livre a un identifiant unique. Plusieurs titres semblables peuvent exister, et un auteur peut avoir écrit plusieurs livres. Donc, seul l'identifiant unique d'un livre permet de déterminer les autres attributs.

$$isbn \rightarrow titre,\ genre,\ auteur,\ nbExemplaire$$

### 3.4.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.5 StatutEmprunt(statutEmprunt) et StatutCommande(statutCommande)

### 3.5.1 Dépendances fonctionnelles

Ces deux relations n'ont qu'un seul attribut, donc aucune dépendance fonctionnelle non triviale.

### 3.5.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.6 Commande(idCommande, #isbn, #idClient, #statutCommande, dateCommande, dateDispo, dateLimiteRetrait)

### 3.6.1 Dépendances fonctionnelles

L'identifiant unique permet de déterminer de manière unique les autres attributs. Rien de dépend de l'isbn, car il peut y avoir plusieurs commandes sur un même livre ; un client peut faire jusqu'à trois commandes, donc il peut y avoir plusieurs commandes pour un même client ; il peut y avoir plusieurs commandes pour une même date.

$$idCommande \rightarrow isbn,\ idClient,\ statutCommande,\ dateCommande,\ dateDispo,\ dateLimiteRetrait$$

### 3.6.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.7 RangCommande(#idCommande, #isbn, rang)

### 3.7.1 Dépendances fonctionnelles

Puisqu'il peut y avoir plusieurs commandes avec un même rang (mais différents livre), et qu'un livre peut avoir plusieurs commandes en attente, seul l'identifiant de commande permet de trouver les deux autres attributs.

$$idCommande \rightarrow isbn,\ rang$$

### 3.7.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

## 3.8 Emprunte(idEmprunt, #isbn, #idClient, #statutEmprunt, dateEmprunt, dateLimiteRetour)

### 3.8.1 Dépendances fonctionnelles

Puisqu'il peut y avoir plusieurs exemplaires d'un même livre, l'isbn ne permet par de déterminer un emprunt unique. Un client peut avoir plusieurs emprunts (complétés ou actifs), et plusieurs emprunts peuvent afin le même statut ; même chose pour les dates. Donc, seul l'identifiant d'emprunt permet de déterminer les autres attributs.

$$idEmprunt \rightarrow\ isbn,\ idClient,\ statutEmprunt,\ dateEmprunt,\ dateLimiteRetour$$

### 3.8.2 3NF

Les attributs sont atomiques (1NF), ne dépendent pas d'une partie de la clé (2NF), et dépendent directement de la clé (3NF).

### 3.9 Schéma final

Le schéma n'a pas changé, puisque toutes les relations étaient déjà 3NF.

- Client(idClient, permEmprunt, nom)
- AdresseClient(#idClient, #idAdresse)
- Adresse(idAdresse, rue, ville, code, pays)
- Livre(isbn, titre, genre, auteur, nbExemplaire)
- StatutEmprunt(statutEmprunt)
- StatutCommande(statutCommande)
- Commande(idCommande, #isbn, #idClient, #statutCommande, dateCommande, dateDispo, dateLimiteRetrait)
- RangCommande(#idCommande, #isbn, rang)
- Emprunte(idEmprunt, #isbn, #idClient, #statutEmprunt, dateEmprunt, dateLimiteRetour)