Projet 1

Cours: IFT2935 - H21
Professeur : Michel Boyer

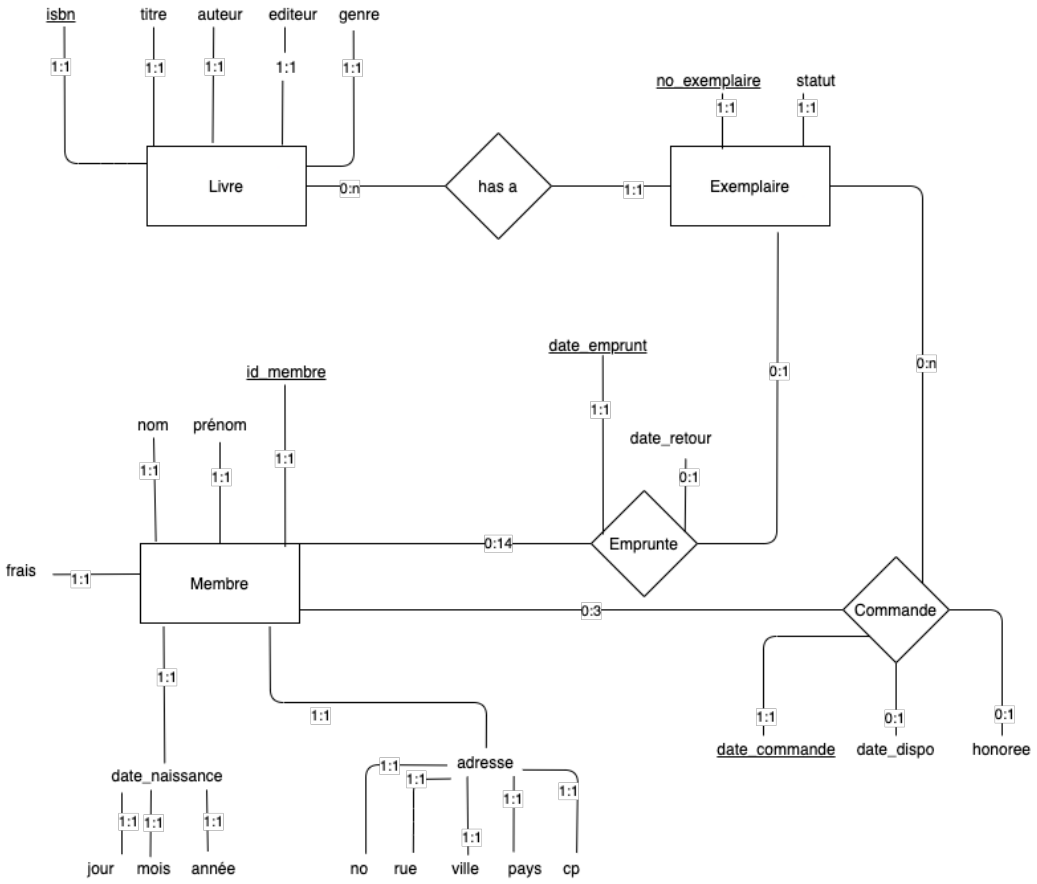
Alice Dorion, 20091504
Alex Mongrain 20034572
Abderrahim Tabta 20133680
Skander Ben Ahmed, p1034870

Pour le 16 avril 2021 à 23:59

Contents

1 Modélisation

1.1 Diagramme entité-association



source: https://app.diagrams.net/#G1Cr33Aw2Q9hgp0qhGKXZzacYhI373vbto

1.2 Explication

Nous avons commencé par créer les types d'entité *Livre* et *Exemplaire*. L'exemplaire d'un livre est le livre physique qu'on peut retrouver ou emprunter à la bibliothèque. L'entité Livre, à travers ses attributs, donne toutes les informations nécessaires sur un livre qui peut être identifié avec un isbn unique. Un livre peut avoir plusieurs genres et plusieurs auteurs. Chaque livre peut avoir plusieurs exemplaires qui sont identifiés par leurs numéros. Chaque exemplaire a un statut qui nous permet de savoir s'il est disponible à l'emprunt, emprunté ou réservé.

De plus, nous avons créé le type d'entité *Membre* qui représente les adhérents de la bibliothèque qui sont identifiés par leur id's uniques. En plus de leur nom et adresse, nous avons supposé que la bibliothèque aurait intérêt à garder des informations sur les abonnés tels que les frais associés aux retards et aux commandes non-honorées, la date d'expiration de leur abonnement et leur date de naissance pour des fins de statistiques. Le membre a aussi une adresse courriel pour qu'il puisse être contacté, par exemple, lorsqu'un livre qu'il a commandé devient disponible. Nous avons assumé que la bibliothèque serait située au Canada et que les membres doivent avoir une adresse de résidence

dans le pays. L'attribut adresse n'a donc pas de pays comme sous-attribut.

Membre a deux types d'associations avec *Exemplaire*. En effet, chaque membre peut soit emprunter ou commander un ou plusieurs exemplaires. Un exemplaire peut être commandé ou emprunté par un seul membre à la fois.

Un emprunt est identifié par les entités qui sont impliqués ainsi que la date d'emprunt. Il y a une date de retour prévue *date_ du*, qui peut changer lors de renouvellements d'emprunt, ainsi qu'une *date_ retour* qui est la date à laquelle le livre a été retourné (s'il est retourné).

Une commande quant à elle a une date de commande, une *date_ dispo* à partir de laquelle l'exemplaire est disponible pour l'emprunt (s'il le devient) et une *date_ emprunt* renvoyant à l'emprunt de l'exemplaire si la commande est honorée. Nous n'avons pas ajouté d'attribut pour une date d'échéance puisqu'elle peut être retrouvée à partir de la *date_ dispo*. Une commande pouvant être annulée à tout moment avant la date d'échéance, elle a aussi un attribut *annul*.

2 Transformation

2.1 Modèle relationnel

Livre (isbn, titre, éditeur)
Auteur (auteur, #isbn)
Genre (genre, #isbn)
Exemplaire (#isbn, no_exemplaire, statut)
Membre (id_membre, prenom, nom, courriel, frais, date_exp, date_naiss, no_civ, apt, rue, ville, cp)
Emprunte (#id_membre, #isbn, #no_exemplaire, date_emprunt, date_du, date_retour)
Commande (#id_membre, #isbn, #no_exemplaire, date_commande, date_dispo, #date_emprunt, annul)

2.2 Explication

D'abord, les types d'entités Livre, Exemplaire et Membre deviennent des relations.

Livre : La relation Livre a comme attributs les propriétés simples isbn, titre et éditeur. Nous avons créé une relation pour ses attributs multivalués **Auteur** et **Genre**.

Auteur : Cette relation a l'attribut auteur lui-même ainsi que la clé primaire de Livre dont il en était attribut.

Genre : Similairement à Auteur, elle contient l'attribut genre lui-même et la clé primaire de Livre.

Exemplaire : Exemplaire a une association avec lien 1:1 avec Livre. On y ajoute donc la clé externe #isbn en plus de ses propres attributs.

Membre : Enfin, pour ce qui de l'entité Membre, on ajoute dans la relation toutes ses propriétés simples. Pour ce qui est de l'attribut composite adresse, on ajoute toutes ses composantes simples (no_civ, rue ville ,pays, cp) dans la relation.

Ensuite, les associations significatives deviennent des relations.

Emprunte : Emprunte est une association sans lien 1:1 avec d'autres entités.
Elle devient donc une relation qui a comme attributs ses propres attributs, date_emprunt, date_du et date_retour, ainsi que les clés primaires des entités impliquées, isbn, no_exemplaire et id_membre.

Commande : Commande est une association sans lien 1:1 avec d'autres entités.
Elle devient donc une relation qui a comme attributs ses propres attributs date_commande, date_dispo, date_emrpunt, annul ainsi que les clés primaires des entités impliquées isbn, no_exemplaire et id_membre.

3 Normalisation des relations

3.1 Définition des dépendances fonctionnelles

Livre

Clés: {isbn}
DFEs: isbn → titre, éditeur

Auteur

Clés: {isbn}
DFEs: isbn → auteur

Genre

Clés: {isbn}
DFEs: isbn → genre

Exemplaire

Clés: isbn , no_exemplaire
DFEs: Isbn, no_exemplaire → statut

Membre

Clés: {id_membre}
{courriel}
DFEs:
id_membre → prenom, nom, courriel, frais, date_exp, date_naiss, no_civ, apt, rue, ville, cp
courriel → id_membre
prenom, nom, date_naiss, no_civ, apt, cp → id_membre
cp → rue, ville

Emprunte

Clés: {isbn, no_exemplaire, date_emprunt}
{isbn, no_exemplaire, date_retour}
DFEs:
isbn, no_exemplaire, date_emprunt → id_membre, date_du, date_retour
isbn, no_exemplaire, date_retour → id_membre, date_du, date_emprunt

Note: La date d'emprunt à elle seule ne détermine pas nécessairement la date de retour car cette dernière peut changer avec des renouvellements d'emprunt. Nous supposons que la bibliothèque ne va pas empêcher qu'un membre commande plusieurs exemplaires du même livre, car c'est un scénario qui ne risque pas souvent d'arriver. {id_membre, isbn, date_emprunt} par exemple ne peuvent donc pas former une clé sans no_exemplaire.

Commande

Clés:	{id_membre, isbn, no_exemplaire, date_commande}	
DFEs:	id_membre, isbn, no_exemplaire, date_commande →	date_dispo, date_emprunt, annul

Note: date_dispo ne peut pas former une clé avec isbn et no_exemplaire, car si un membre annule sa commande lorsque l'exemplaire est disponible, l'attribut date_dispo d'une commande de ce même exemplaire par un autre membre prendrait la même valeur.

3.2 Normalisation

Livre

1NF:	Il n'y a pas d'attribut non-atomique dans cette table.
2NF:	Titre et éditeur ne sont pas en dépendance fonctionnelle d'une partie de la clé.
3NF:	Il n'y a pas de dépendance fonctionnelle transitive non plus.
BCNF:	Il n'y a qu'une seule df, la relation est donc en BCNF.

Auteur

BCNF:	On a montré dans le tp3 qu'une table avec seulement deux atributs est toujours en BCNF.

Genre

BCNF:	On a montré dans le tp3 qu'une table avec seulement deux atributs est toujours en BCNF.

Exemplaire

1NF:	Tous les attributs sont atomiques.
2NF:	Statut, le seul attribut non clé, n'est pas en dépendance fonctionnelle d'une partie de la clé (isbn,no_exemplaire).
3NF:	La seule dépendance fonctionnelle est isbn, no_exemplaire → statut, il n'y a donc pas de DF transitive.
BCNF:	Il ny a qu'une seule DF, la relation est donc en BCNF.

Membre

1NF:	Tous les attributs sont atomiques (date_exp et date_naiss sont des timestamps).
2NF:	La table n'est pas en 2NF car il y a transitivité dans prenom, nom, date_naiss, no_civ, apt, cp → id_membre et cp → rue.

Il faut donc diviser Membre en deux relations:

- Membre(id_membre, prenom, nom, frais, date_exp, date_naiss, no_civ, cp), avec les DF:

  id_membre → prenom, nom, frais, date_exp, date_naiss, no_civ, apt, cp
  courriel → id_membre lp
  prenom, nom, date_naiss, no_civ, apt, cp → id_membre

- Bloc_poste(cp, rue, ville), avec la DF:

  cp → rue, ville

Membre (suite)
3NF: Membre et Bloc_poste sont en 3NF puisqu'elles n'ont pas de DF transitive.
BCNF: Membre et Bloc_poste sont en BCNF puisque les déterminants de toutes les DF sont des superclés.

Emprunte
1NF: Tous les attributs sont atomiques.
2NF: Il n'y a pas de dépendance partielle.
3NF: Il n'y a pas de dépendance transitive.
BCNF: Tous les déterminants des DF élémentaires sont des clés candidates.

Commande
1NF: Tous les attributs sont atomiques.
2NF: Il n'y a pas de dépendance partielle.
3NF: Il n'y a pas de dépendance transitive.
BCNF: Il n'y a qu'une clé unique.

Les tables normalisées sont dans le fichier projet.sql. Il contient une table additionnelle *statuts* pour la contrainte sur les valeurs possibles de l'attribut Exemplaire.statut .

4 Q&A - Requêtes

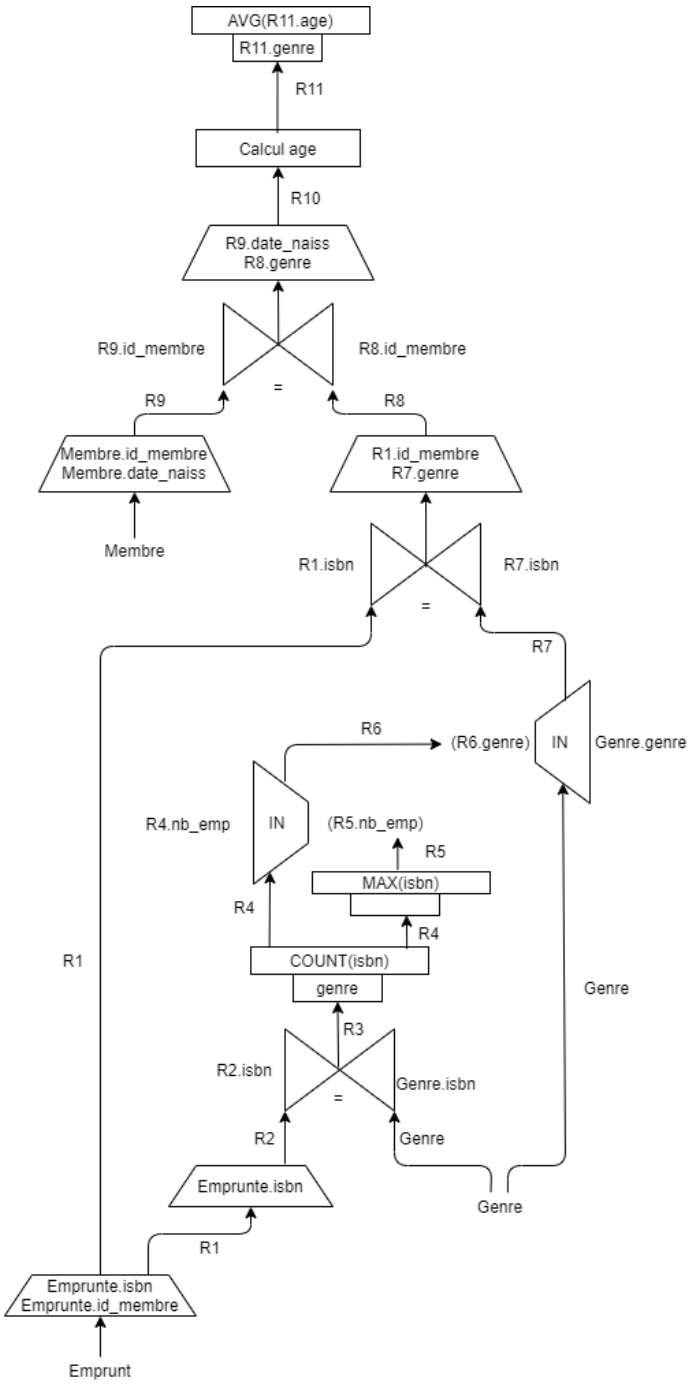
1- Le nombre d'abonnés qui ont accumulé des frais de retard de plus de 5$.
$result \leftarrow \mathscr{A}_{count(id\_membre)}(\sigma_{frais>5}(\pi_{id\_membre,frais}(Membre)))$

Aspect d'optimisation: Nous avons fait la projection des attributs id_membre et frais sur Membre avant la sélection.

2- Âge moyen des membres ayant loué un livre du genre le plus populaire (le plus emprunté).

$$R1 \leftarrow \pi_{(isbn,id\_membre)}(Emprunte)$$
$$R2 \leftarrow \pi_{(isbn)}(R1)$$
$$R3 \leftarrow R2 \bowtie Genre$$
$$R4 \leftarrow \rho_{(genre,nb\_emp)}(genre\ \mathscr{A}_{COUNT(isbn)}(R3))$$
$$R5 \leftarrow \rho_{(nb\_emp)}(\mathscr{A}_{MAX(nb\_emp)}(R4))$$
$$R6 \leftarrow \pi_{(genre)}(\sigma_{(nb\_emp\ IN\ R5)}(R4))$$
$$R7 \leftarrow \sigma_{(genre\ IN\ R6)}(Genre)$$
$$R8 \leftarrow \pi_{(id\_membre,genre)}(R1 \bowtie R7)$$
$$R9 \leftarrow \pi_{(id\_membre,date\_naiss)}(Membre)$$
$$R10 \leftarrow \pi_{(date\_naiss,genre)}(R9 \bowtie R8)$$
$$R11 \leftarrow \rho_{(genre,age)}(\pi_{(genre,date\_part('year',\ CURRENT\_DATE)\ -\ date\_part('year',date\_naiss))}(R10))$$
$$\rho_{(genre,age\_moy)}(genre\ \mathscr{A}_{AVG(age)}(R11))$$

Aspect d'optimisation:



On peut voir sur le diagramme qu'avant chaque jointure, on effectue une réduction en largeur ou en hauteur de chaque table. Ainsi chaque table résultante a une hauteur et une largeur minimale. La table produite en sortie est donc optimisée au maximum.

3- Liste de noms et prénoms des membres et leur retard accumulé en nombre de jours total sur tous leurs emprunts, en ordre décroissant de nombre de jours de retard.

$$
\begin{aligned}
mem &\leftarrow \pi_{(id\_membre,prenom,nom)}(Membre) \\
emp1 &\leftarrow \sigma_{(date\_retour>date\_du)}(Emprunte) \\
emp2 &\leftarrow \sigma_{(date\_retour=NULL \wedge (CURRENT\_DATE>date\_du))}(Emprunte) \\
emp\_retard1 &\leftarrow \rho_{(id\_membre,isbn,no\_ex,date\_emprunt,retard)}(\\
&\qquad \pi_{(id\_membre,isbn,no\_ex,date\_emprunt,(date\_retour-date\_du))}(emp1)) \\
emp\_retard2 &\leftarrow \rho_{(id\_membre,isbn,no\_ex,date\_emprunt,retard)}(\\
&\qquad \pi_{(id\_membre,isbn,no\_ex,date\_emprunt,CURRENT\_DATE-date\_du)}(emp2)) \\
retard\_total &\leftarrow \rho_{(id\_membre,total)}({}_{id\_membre}\mathscr{A}_{sum(retard)}(emp\_retard1 \cup emp\_retard2)) \\
result &\leftarrow \pi_{prenom,nom,total}(mem \bowtie_{id\_membre} retard\_total)
\end{aligned}
$$

Aspect d'optimisation: Nous avons restraint le nombre de colonnes des tables en effectuant des projections avant les manipulations de sélection et de jointure pour ne conserver que les attributs requis. Par exemple, la relation *mem* qui est utilisée dans la jointure finale est le résultat d'une projection sur Membre qui ne conserve que les colonnes *id_ membre, prenom* et *nom*. Nous ne pouvons pas faire moins de projections pour obtenir *emp1, emp2, emp_ retard1* et *emp_ retard2*, car étant donné que les projections en algèbre relationnelle sont l'équivalent de SELECT DISTINCT en *sql*, les colonnes des attributs faisant partie de la clé sont requises pour garder toutes les lignes distinctes. Notre implémentation de cette requête en code *sql* est plus efficace car elle ne garde pas autant de colonnes dans les résultats intermédiaires.

4- Le genre avec le plus grand nombre de commandes honorées et leur nombre de commandes honorées

$$
\begin{aligned}
com &\leftarrow \pi_{(id\_membre,isbn,no\_exemplaire,date\_commande)}(\sigma_{(date\_emprunt\neq NULL)}(Commande)) \\
genre\_com &\leftarrow \rho_{(genre,nb\_com)}({}_{genre}\mathscr{A}_{count(*)}(Genre \bowtie_{isbn} com)) \\
max\_genre &\leftarrow \rho_{(genre,nb\_com)}({}_{genre}\mathscr{A}_{max(nb\_com)}(genre\_com)) \\
result &\leftarrow \pi_{genre,nb\_com}(genre\_com \bowtie_{nb_com} max\_genre)
\end{aligned}
$$

Aspect d'optimisation: Dans notre implémentation sql, cette requête est plus efficace car nous avons utilisé SELECT plutôt que la projection propre. En effet, la table intermédiaire *com* dans l'implémentation ne garde que la colonne *isbn* plutot que toute la clé de Commande.

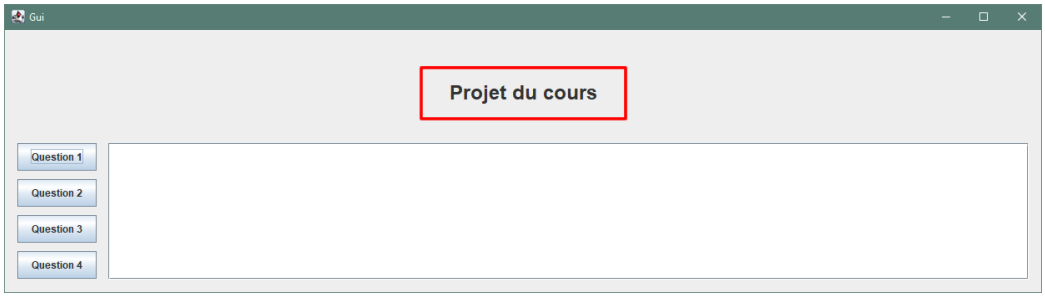
5 GUI - Manuel d'utilisation

Librairie utilisée :	postgresql-42.2.19
Compilé avec :	Java SE Development Kit 13.0.2 (JDK 13.0.2)
Exécution :	Démarrer postgres et insérer rouler le script SQL Script.sql Dans la console : java -jar Gui.jar

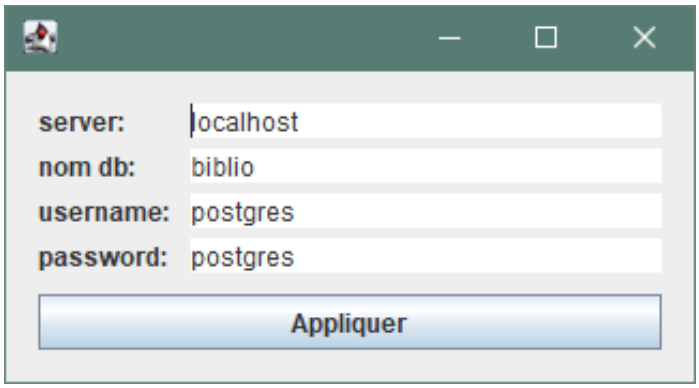
Par défaut les valeurs pour la connexion à la base de données sont:

server :	localhost
database :	biblio
user :	postgres
password :	Dans la console : postgres

Ces valeurs peuvent être changées en cliquant sur le titre de l'interface (projet du cours)



Une fenêtre s'ouvrira, modifiez les valeurs et appuyez sur "Appliquer" pour appliquer les changements



Appuyez sur un des boutons Question 1, Question 2, Question 3 ou Question 4. Si les valeurs de server, database, username et password sont valides, la boite de texte affichera les lignes du résultat de la requête SQL avec dans chaque ligne la valeur des colonnes séparées par une virgule.

Si la valeur entrée pour serveur est invalide, la boite de texte affichera le message suivant lorsque vous appuierez sur une des questions.

Si la valeur entrée pour database est invalide, la boite de texte affichera le message suivant lorsque vous appuierez sur une des questions.

Si la valeur entrée pour username ou password est invalide, la boite de texte affichera le message suivant lorsque vous appuierez sur une des questions.