# Bases de données :
# Exercices

**IUT de Villetaneuse — R&T 1ère année**

Laure Petrucci

24 mars 2018

# 1 Diagramme de classes, Schéma relationnel, Normalisation

### Exercice 1.1 : Informatisation d'une bibliothèque

Le texte suivant a été rédigé par une bibliothécaire ne possédant pas de compétence particulière en informatique (aidée par un « spécialiste ») :

> *Grâce à cette informatisation, un abonné doit pouvoir retrouver un livre en connaissant son titre. Il doit aussi pouvoir connaître la liste des livres écrits par un certain auteur, ou la liste par éditeur ou encore la liste par genre (bd, sf, policier, etc...). Chaque livre est acheté en un ou plusieurs exemplaires. On souhaite également mettre en place une procédure de recherche documentaire par mots clés. Chaque ouvrage peut être associé à plusieurs mots clés. La gestion des prêts implique la possibilité de connaître à tout moment la liste des livres détenus par un abonné, et inversement, que l'on puisse retrouver le nom des abonnés détenant un livre absent des rayons.*

**Question 1 :** Modifier ce texte en essayant de préciser certains termes, de façon à obtenir un cahier des charges précis.

**Question 2 :** Relire ce cahier des charges et surligner d'une couleur les mots devant conduire à des classes et d'une autre couleur ceux devant conduire à des associations ou des classes-associations.

**Question 3 :** Déduire de ce cahier des charges le diagramme de classes correspondant.

**Question 4 :** Déduire du diagramme de classes le schéma relationnel correspondant.

**Question 5 :** Quelles sont les clés primaires et étrangères de ces relations ?

**Question 6 :** Le schéma relationnel obtenu est-il en $3^{\text{ème}}$ forme normale ?

**Question 7 :** Quelqu'un propose un schéma relationnel contenant la relation suivante :
`LIVRE(id_livre,id_exemplaire, titre, dateCréation, dateAcquisition, dateEdition)`
Peut-on accepter cette proposition ?

# 2 Algèbre Relationnelle

### Exercice 2.1 : Informatisation d'une bibliothèque

On souhaite effectuer des requêtes sur la bibliothèque dont le modèle relationnel a été conçu au TD1.
Ces requêtes permettent à l'utilisateur d'interroger la base de données.
Dans un premier temps, nous exprimerons les requêtes sous forme d'opérations de l'algèbre relationnelle.

Pour imaginer facilement le résultat des opérations que l'on souhaite effectuer, voici quelques instanciations des relations de la base :

MOT_CLÉ

| id_motclé | motclé |
|---|---|
| 1 | mathématiques |
| 2 | fiction |
| 3 | théorie |
| 4 | univers |
| 5 | statistiques |
| 6 | cinéma |

GENRE

| id_genre | nomGenre |
|---|---|
| 1 | BD |
| 2 | roman |
| 3 | théâtre |
| 4 | SF |
| 5 | guide |
| 6 | technique |
| 7 | policier |
| 8 | biographie |
| 9 | poésie |

AUTEUR

| id_auteur | nom | prénom |
|---|---|---|
| 1 | Franquin | André |
| 2 | Eddings | David |
| 3 | Volkoff | Vladimir |
| 4 | Bond | Edward |
| 5 | Hugo | Victor |
| 6 | Brumark | Annika |

LIVRE

| id_livre | titre | dateCréation |
|---|---|---|
| 1 | Idées noires | 1931 |
| 2 | Le grand tsar blanc | 1912 |
| 3 | Lear / La mer | 1953 |
| 4 | Le pion blanc des présages | 1895 |
| 5 | Métro pour l'enfer | 1964 |
| 6 | La reine des sortilèges | 1845 |
| 7 | Alexandre Nevsky | 1820 |
| 8 | L'homme qui rit | 2001 |
| 9 | Les travailleurs de la mer | 1842 |
| 10 | Hernani | 1830 |
| 11 | Le réseau | 1974 |
| 12 | Le réseau | 1982 |

DÉCRIT

| id_motclé | id_livre |
|---|---|
| 2 | 1 |
| 4 | 2 |
| 3 | 3 |
| 2 | 4 |
| 2 | 5 |
| 2 | 6 |
| 4 | 7 |
| 2 | 8 |
| 2 | 9 |
| 3 | 10 |
| 3 | 11 |
| 3 | 12 |

CORRESPOND

| id_genre | id_livre |
|---|---|
| 1 | 1 |
| 4 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 2 | 4 |
| 4 | 5 |
| 4 | 6 |
| 1 | 7 |
| 2 | 8 |
| 8 | 8 |
| 2 | 9 |
| 3 | 10 |
| 5 | 11 |
| 2 | 12 |

ÉCRIT

| id_auteur | id_livre |
|---|---|
| 1 | 1 |
| 3 | 2 |
| 4 | 3 |
| 2 | 4 |
| 3 | 5 |
| 2 | 6 |
| 3 | 7 |
| 5 | 8 |
| 5 | 9 |
| 5 | 10 |
| 2 | 11 |
| 6 | 12 |

EXEMPLAIRE

| id_exemplaire | date_acquisition | année_édition | id_livre | id_éditeur |
|---|---|---|---|---|
| 1 | 1995 | 1990 | 4 | 4 |
| 2 | 1995 | 1991 | 6 | 4 |
| 3 | 1995 | 1963 | 5 | 4 |
| 4 | 1996 | 1995 | 2 | 3 |
| 5 | 1997 | 1997 | 2 | 2 |
| 6 | 1999 | 1998 | 3 | 6 |
| 7 | 2000 | 1996 | 7 | 5 |
| 8 | 2001 | 1987 | 10 | 2 |
| 9 | 2002 | 2001 | 1 | 1 |
| 10 | 2003 | 2002 | 8 | 2 |
| 11 | 2003 | 2002 | 9 | 2 |
| 12 | 2003 | 2002 | 9 | 7 |
| 13 | 2004 | 2003 | 11 | 9 |
| 14 | 2000 | 2000 | 12 | 8 |

ÉDITEUR

| id_éditeur | nom | adresse |
|---|---|---|
| 1 | Fluide Glacial | 33 avenue du Maine, Paris 15 |
| 2 | Livre de Poche | 43 quai de Grenelle, Paris 15 |
| 3 | Fallois | 22 rue de la Boétie, Paris 8 |
| 4 | Presses Pocket | 12 avenue d'Italie, Paris 13 |
| 5 | Lombard | 15-27 rue Moussorgski, Paris 18 |
| 6 | Arche | 6 rue Bonaparte, Paris 6 |
| 7 | Casterman | 66 rue Bonaparte, Paris 6 |
| 8 | Anne Carrière | 66 rue Bonaparte, Paris 6 |
| 9 | IRMA | 22 rue Soleillet, Paris 20 |

ABONNÉ

| id_abonné | nom | date_naissance | date_adhésion |
|---|---|---|---|
| 1 | Dupond | 15/05/1973 | 10/09/2000 |
| 2 | Schmidt | 01/04/1959 | 01/10/2001 |
| 3 | Thomas | 29/02/1964 | 20/01/2002 |

EMPRUNTE

| id_abonné | id_exemplaire | date_retour |
|---|---|---|
| 1 | 9 | 25/01/2010 |
| 1 | 7 | 25/01/2010 |
| 1 | 12 | 25/01/2010 |
| 3 | 8 | 10/02/2010 |
| 3 | 11 | 10/02/2010 |

**Question 1 :** Quels sont les livres de la bibliothèque intitulés *Le réseau* ?

**Question 2 :** Quelles sont les *clés* des exemplaires empruntés par l'abonné numéro 3 ?

**Question 3 :** Quelles sont les *clés* des exemplaires acquis par la bibliothèque en 2003 ?

**Question 4 :** Quels sont les noms des *éditeurs* ayant publié les exemplaires que la bibliothèque possède ?

**Question 5 :** Trouver les exemplaires du livre dont le titre est *Le grand tsar blanc*.

**Question 6 :** Quelles *bandes dessinées* trouve-t-on à la bibliothèque ?

**Question 7 :** Quels sont les livres de la bibliothèque dont l'auteur est *David Eddings* ?

**Question 8 :** Quels sont les noms des auteurs ayant écrit à la fois des romans et des bandes dessinées ?

**Question 9 :** Quels sont les noms des auteurs ayant écrit des romans ou des bandes dessinées ?

**Question 10 :** Quels sont les noms des auteurs de romans n'ayant pas écrit de bande dessinée ?

**Question 11 :** Quels sont les titres des romans se déroulant dans le monde du cinéma que la bibliothèque possède ?

**Question 12 :** Quels sont les abonnés ayant emprunté un livre intitulé *Le réseau* et dont l'auteur est *Annika Brumark* ?

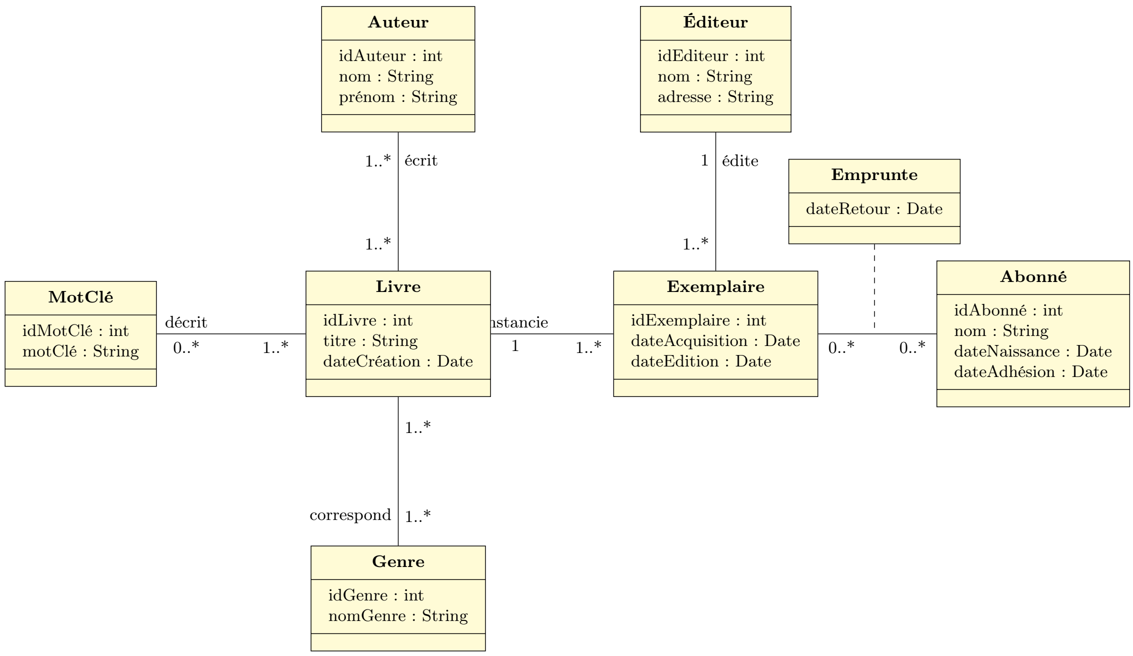
**Question 13 :** Quels sont les titres des livres publiés par les éditeurs ayant publié *Les travailleurs de la mer* ?

# 3 Utilisation d'un logiciel de modélisation

### Exercice 3.1 : Utilisation du logiciel `PowerAMC` sous Windows

Le but de cet exercice est d'utiliser un logiciel pour créer un diagramme de classes UML puis de passer, par étapes successives, de ce diagramme de classes au script SQL permettant de créer les différentes tables d'une base de données.

**Question 1 :** Création d'un diagramme de classes :
Il s'agit de créer un sous-ensemble du diagramme de classes du TD1 :




Remarque : les multiplicités 1 n'apparaissent pas sur le diagramme de classes.

Pour simplifier votre travail, vous ne représenterez pas les classes MotClé et Genre ainsi que les relations Décrit et Correspond.

Lancer, sous Windows, le logiciel `PowerAMC` (`VosLogiciels\IUT_CRIT\Logiciels\AMCModélisation\PowerAMC11`).
Ouvrir un nouveau *modèle orienté objet* : (`Fichier\Nouveau\ Modèle Orienté Objet`)
Vérifier que dans la fenêtre qui s'ouvre, le champ « Premier Diagramme » contienne bien « Diagramme de classes ».
Créer le diagramme de classes du TD1 en utilisant les icônes de la palette d'outils.
Un double-clic sur une classe permet de la renommer.
Un clic droit permet de rajouter les attributs en les nommant, en choisissant leur type dans une liste.
Il faut, en vue du passage au schéma relationnel, que les identifiants (idAuteur, idAbonné, etc...) soient déclarés identifiant primaire (icône à gauche « propriété de l'attribut » onglet « détails » : cocher la case « identifiant primaire »).

Il faut enfin mettre à jour les multiplicités des associations lorsque cela est nécessaire (double-clic sur l'association, propriétés de l'association, détails).
Pour créer une classe-association, il faut 1/ créer une association normale ; 2/ créer une classe « normale » (la future classe-association) ; 3/ créer une nouvelle association entre cette classe et la première association.

**Question 2 :** Vérifiez votre modèle (`Outils\Vérifier le modèle`) et, si nécessaire, modifiez votre diagramme de classes pour ne plus obtenir d'erreur lors de cette vérification. Enregistrez votre modèle objet.

**Question 3 :** Générez le *modèle physique* correspondant à votre *modèle objet* (`Outils\Générer le modèle physique`).
Le SGBD cible doit être `PostgreSQL7.3`.
Toutes les classes sont transformées en tables ainsi que certaines associations.
Quelles associations ont disparu ? Pourquoi ? Sous quelle forme sont-elles quand même prises en compte ?
Enregistrez votre modèle physique.

**Question 4 :** À partir du *modèle physique*, générez le *script SQL de création des tables* (`SGBD\Générer la base de données`). Ce script se présente sous forme d'un fichier ayant pour extension `.sql`.
Éditez-le et étudiez-le. L'ordre de création des tables est-il quelconque ou obéit-il à une logique, et si oui, laquelle ?
Enregistrez le script SQL sur une clé.

**Question 5 :** On souhaite maintenant procéder en sens inverse, c'est-à-dire obtenir le diagramme de classes à partir du script SQL de création de la base.
Fermer tous les modèles ouverts. À partir du script, générer le modèle physique (`Fichier`, `Reverse Engineering`→`Base de données`), puis le diagramme de classes (menu `Outils`).

## Exercice 3.2 : Création et initialisation de votre base de données sous linux

Quittez Windows et redémarrez votre machine sous linux.
Chacun d'entre vous possède une seule base de données sur la machine aquanux.
Vous pouvez y accéder en utilisant l'interpréteur de commandes psql : tapez :

```
psql -h aquabdd -d promotion17
```

Le mot de passe qui vous est demandé est votre numéro INE.

**Question 1 :** Utilisez le script obtenu sous Windows pour créer les tables de la bibliothèque en utilisant la commande :

```
\i votreScript.sql
```

Vous devrez faire quelques modifications avant que le script ne soit accepté.
Pour vérifier que les tables ont bien été créées :

```
\d
```

Pour vérifier la structure de chaque table :

```
\d nomtable
```

Insérez quelques tuples dans vos tables et vérifiez ces insertions.

Supprimez ensuite toutes les tables de votre base.
L'ordre de destruction des tables est-il important, et, si oui, quelle est sa logique ?

**Question 2 :** Vous trouverez dans le sous-répertoire `scripts_creation` du répertoire `/home/usager/TPGTR/GTR1/BD` le fichier `creerBiblio.sql` permettant de créer les tables de la bibliothèque. Editez ce fichier, étudiez-le.
Créez les tables dans votre base de données en utilisant ce fichier.

**Question 3 :** Utiliser les fichiers d'extension `.txt` se trouvant dans dans le sous-répertoire `fichiers_texte` du répertoire `/home/usager/TPGTR/GTR1/BD` pour remplir les tables.
Exemple : pour remplir la table motcle, vous utiliserez la commande :

```
\copy motcle from motcle.txt with delimiter ';'
```

## Exercice 3.3 : Sauvegarde de la base

**Question 1 :** Dans un shell, utilisez la commande `pg_dump` permettant de sauvegarder le contenu de votre base. Vous redirigerez le résultat dans un fichier de nom `sauve_bibli.sql` :

```
pg_dump -h aquabdd -n nom_base promotion17 > sauve_bibli.sql
 (nom_base est votre numéro d'étudiant.)
```

Éditez le fichier et examinez son contenu.

**Question 2 :** Détruisez les tables que vous avez créées. Vérifiez qu'il n'y a plus rien.

**Question 3 :** Recréez les tables et leur contenu à partir de `sauve_bibli.sql`.

**Question 4 :** Recréez une dernière fois les tables et leur contenu à partir du fichier `creerBiblio.sql`.
Dans la suite des tps de bases de données, en cas de problème, régénérez votre base en utilisant ce fichier.

# 4 Langage de requêtes SQL : requêtes simples

Remarque préliminaire : l'éditeur de psql est malcommode. Nous vous conseillons donc de rédiger vos requêtes dans un fichier texte requetes.sql puis de les soumettre à psql :

```
\i requetes.sql
```

Lorsqu'une requête est mise au point, mettez-la en commentaires (/* ... */) et passez à la suivante.

### Exercice 4.1 : Requêtes sur la bibliothèque

Écrire les requêtes SQL permettant de répondre aux questions suivantes.

**Question 1 :** Quels sont les livres de la bibliothèque intitulés *Le réseau* ?

**Question 2 :** Quelles sont les *clés* des exemplaires empruntés par l'abonné numéro 3 ?

**Question 3 :** Quelless sont les *clés* des exemplaires acquis par la bibliothèque en 2003 ?

**Question 4 :** Quels sont les noms des *éditeurs* ayant publié les exemplaires que la bibliothèque possède ?

**Question 5 :** Trouver les exemplaires du livre dont le titre est *Le grand tsar blanc*.

**Question 6 :** Quelles *bandes dessinées* trouve-t-on à la bibliothèque ?

**Question 7 :** Quels sont les livres de la bibliothèque dont l'auteur est *David Eddings* ?

**Question 8 :** La bibliothèque possède-t-elle des *BDs* associées au mot clé « fiction » ?

**Question 9 :** Quels sont les noms des auteurs ayant écrit à la fois des romans et des bandes dessinées ?

**Question 10 :** Quels sont les noms des auteurs ayant écrit des romans ou des bandes dessinées ?

**Question 11 :** Quels sont les noms des auteurs de romans n'ayant pas écrit de bande dessinée ?

**Question 12 :** Quels sont les abonnés ayant emprunté un livre intitulé *Hernani* et dont l'auteur est *Victor Hugo* ?

# 5 Requêtes SQL complexes

### Exercice 5.1 : Base de données de la bibliothèque

**Question 1 :** Quels sont les noms des éditeurs ayant publié au moins un livre entre 1995 et 1999 ?

**Question 2 :** Quel est le nombre d'exemplaires de chaque livre ?

**Question 3 :** Quel est le nombre de livres écrits par chaque auteur ?

**Question 4 :** Quel est l'année moyenne d'acquisition des exemplaires par la bibliothèque ?

**Question 5 :** Quel est, pour chaque mot clé, le nombre de livres décrits par ce mot clé ? On indiquera la clé du mot clé ainsi que le nombre de livres.

**Question 6 :** Quels sont les auteurs dont la bibliothèque possède plusieurs exemplaires de leurs livres ?

### Exercice 5.2 : Quelques requêtes plus avancées

**Question 1 :** Quels sont les titres des livres publiés par les éditeurs ayant publié *Les travailleurs de la mer* ?

**Question 2 :** Quels sont les numéros des exemplaires édités plus tard que tous ceux publiés par l'éditeur *Presses Pocket*, classés par année d'édition croissante ?

**Question 3 :** Quels sont les numéros des exemplaires édités plus tard que l'un de ceux publiés par l'éditeur *Livre de Poche*, classés par date d'acquisition décroissante ?

**Question 4 :** Indiquer, à la place de la clé du mot clé, son libellé.

**Question 5 :** Quel est le nombre d'exemplaires de chaque livre avec la date moyenne de publication ?

**Question 6 :** Quels sont les auteurs dont la bibliothèque possède le plus d'exemplaires de leurs livres ?

**Question 7 :** Quels sont les noms des abonnés qui ont le plus d'emprunts ?

# 6 Contraintes, vues, règles et fonctions

## Exercice 6.1 : Contraintes de domaine, fonctions

**Question 1 :** Soit la relation suivante :

`ARTICLE(id_article, nom_article,categorie,prixHT)`

L'attribut `categorie` prend ses valeurs dans le domaine {1,2}.
Rédigez le script SQL de création de la table correspondant à cette relation, exécutez-le, puis insérez quelques tuples.

**Question 2 :** Compléter la fonction SQL ci-dessous recevant en paramètres le prix hors taxe et la catégorie d'un article et retournant le prix taxes comprises.
La taxe s'élève à 5% pour les articles de catégorie 1 et à 20% pour ceux de catégorie 2.

```
create function pTC(integer,numeric)
returns numeric
as '
    DECLARE
        ........ .......;
    BEGIN
    if ($1 = .)
        then
            ........ := $2 * ....;
        else
            ........ := $2 * ...;
    end if;
    return ........;
    END;
' LANGUAGE 'plpgsql';
```

x

**Question 3 :** Rédiger une requête SQL permettant d'afficher le prix hors taxe et le prix taxe comprise de chaque article.

## Exercice 6.2 : Base de données de la bibliothèque

**Question 1 :** Contraintes d'intégrité : Que se passe-t-il si l'on essaie de supprimer un auteur de la table `auteur` ?
Comment faire pour obtenir cette suppression ?

**Question 2 :** Construire une vue `auteurlivre` permettant d'obtenir, pour chaque livre, son titre, sa date de création, le nom et le prénom de son auteur.

**Question 3 :** Rédiger les fonctions `maximumAuteur()` et `maximumLivre()` retournant respectivement la valeur maximum de `id_auteur` dans la table `auteur` et la valeur maximum de `id_livre` dans la table `livre`.

Ces fonctions ont la forme suivante :

```
create function maximumAuteur()
returns integer
as '
    DECLARE
        ........ .......;
    BEGIN
        select ..............................................
    return ........;
    END;
' LANGUAGE 'plpgsql';
```

**Question 4 :** Créer une règle `ins_auteurlivre` permettant d'insérer un livre et son auteur dans la vue `auteurlivre`. Vérifier que la règle `ins_auteurlivre` est déclanchée lorsqu'on insère dans `auteurlivre` un nouveau livre de titre *Alexandra* écrit en 1929 par *Jacqueline Dauxois*.

**Question 5 :** L'insertion dans la vue `auteurlivre` ne se justifie que si l'auteur et le livre sont nouveaux. Que se passe-t-il si l'on insère par erreur dans la vue un auteur déjà existant ? Comment corriger le problème ?

# 7 Programmation C/SQL

### Exercice 7.1 : Gestion de la bibliothèque

**Question 1 :** Voici la liste des fonctions que vous devez rédiger :

```
PGconn* ouvrirConnexion();
PGresult* executerRequete(PGconn* connexion,char* requete);
void afficherResultat(PGresult* resultat);
void afficherRequete(PGconn* connexion, char* requete);
void afficherTable(PGconn* connexion, char* nomTable);
void supprimerTuple(PGconn* connexion,char* nomTable,char* nomId,char* valeurId);
void ajouterTuple(PGconn* connexion,char* nomTable,char** tabAttributs,int nb);
```

La fonction `ouvrirConnexion` retourne un pointeur de type `PGconn*` représentant une connexion à la base en cas de succès ; elle retourne `NULL` en cas d'échec.

La fonction `executerRequete` exécute la requête `requete` et retourne un pointeur de type `PGresult*` sur le résultat ; elle retourne `NULL` en cas d'échec.

La fonction `afficherResultat` affiche le contenu du résultat pointé par `PGresult* resultat`.

La fonction `afficherRequete` affiche le résultat de la requête reçue en paramètre.

La fonction `afficherTable` affiche le contenu d'une table quelconque de la base de données.

La fonction `supprimerTuple` supprime dans la table de nom `nomTable` un tuple dont l'identifiant a pour nom `nomId` et pour valeur `valeurId`.

La fonction `ajouterTuple` ajoute un tuple à la table de nom `nomTable`.
Les différentes valeurs des attributs sont regroupées dans le tableau `tabAttributs`. Le dernier paramètre `nb` correspond au nombre d'éléments de `tabAttributs`.

Vous testerez ces fonctions en utilisant un programme principal proposant un `menu`. La connection à la base de données devra être effectuée dans le programme principal et passée en paramètre aux fonctions qui en ont besoin.

**Question 2 :** Écrire une fonction affichant la liste des livres empruntés par un abonné de la bibliothèque dont l'utilisateur fournit l'identifiant.

**Question 3 :** Écrire une fonction `void ajouterLivreAuteur(PGconn* connection, char* titre, char* date, char* nom, char* prenom)` permettant d'ajouter un livre et son auteur à la base.
Il faudra vérifier que le même livre du même auteur n'existe pas déjà. Si ce n'est pas le cas, il faudra rajouter le livre dans la table Livre.
Si l'auteur n'existe pas déjà il devra être créé.
Enfin, il faudra rajouter le tuple `(id_auteur,id_livre)` dans la table Ecrit.

# 8 Accès aux bases de données en PHP

### Exercice 8.1 : Base de données de la bibliothèque

Comme en programmation C, vous regrouperez vos fonctions dans une bibliothèque `fonctionsBD.inc`. Pour toutes les questions ci-dessous, après avoir rédigé la fonction demandée, vous écrirez un programme de test (l'équivalent du `main` en C) recevant des paramètres d'un client, appelant cette fonction, puis envoyant (par l'intermédiaire du serveur) le(s) résultat(s) au client.
Utilisez votre navigateur puis celui d'un voisin pour envoyer les requêtes au serveur et pour récupérer ses réponses.
Utilisez aussi le client en mode console `nc` depuis votre poste et depuis celui d'un voisin.

**Question 1 :** Écrire une fonction `etablirConnexion` à une base de données Postgresql : cette fonction recevra quatre paramètres. Tester cette fonction.

**Question 2 :** Écrire un programme permettant de consulter la liste de tous les exemplaires empruntés par un abonné. Le numéro de l'abonné sera reçu en paramètre. Tester ce programme.

**Question 3 :** Écrire une fonction `existeResultat($connexion,$requete)`
Cette fonction retourne `vrai` si le résultat de la requête reçue en paramètre contient au moins un tuple et `faux` sinon. Tester cette fonction.

**Question 4 :** Écrire un programme `nouveauLivre.php`.
Ce programme reçoit quatre paramètres : le titre et la date de création du nouveau livre ainsi que les nom et prénom de l'auteur.
Le programme doit insérer le nouveau livre dans la table livre et, si l'auteur ne figure pas dans la table auteur, il doit y être inséré. Tester ce programme.

# 9 Scripts PHP

### Exercice 9.1 : Base de données de la bibliothèque

On souhaite pouvoir gérer la bibliothèque à partir d'une interface web.

**Question 1 :** Écrire un programme proposant à l'utilisateur les fonctionnalités suivantes :

- Création d'un nouveau livre ;
- Liste de tous les livres ;
- Suppression d'un livre ;
- Recherche d'ouvrages ;
- Inscription d'un nouvel abonné ;
- Modification d'un abonné ;
- Liste de tous les abonnés ;
- Liste de tous les livres empruntés par un abonné ;
- Suppression d'un abonné ;
- Prêt d'un livre ;
- Liste de tous les livres empruntés ;
- Liste de tous les emprunteurs retardataires ;
- Retour d'un livre.

L'appel de la fonctionnalité montrera, pour l'instant, une page ne contenant qu'un titre correspondant au choix de l'utilisateur. Ces pages seront remplacées, dans les questions suivantes, pour réaliser effectivement la fonction souhaitée.

**Question 2 :** Écrire la fonction qui présente la liste de tous les abonnés sous forme de tableau. Cette liste devra être triée par ordre croissant de nom d'abonné.

**Question 3 :** Écrire la fonction fournissant la liste de tous les exemplaires empruntés, comprenant le numéro d'exemplaire ainsi que le titre du livre.

**Question 4 :** Écrire la fonction permettant d'afficher la liste des livres. Celle-ci comprendra le titre du livre, le ou les auteurs ainsi que le nombre d'exemplaires.

**Question 5 :** Écrire la fonction fournissant la liste de tous les abonnés retardataires ainsi que des livres qu'ils auraient dû rendre. On indiquera le nom de l'abonné, ainsi que le titre du livre, le nom du ou des auteurs, le numéro d'exemplaire et la date de retour prévue.

**Question 6 :** Écrire la fonction permettant de consulter la liste de tous les exemplaires empruntés par un abonné. Il faudra indiquer le titre du livre, son ou ses auteurs. Le numéro de l'abonné sera fourni par l'utilisateur.

**Question 7 :** Écrire la fonction permettant de créer un nouvel abonné. Le numéro attribué à cet abonné sera le suivant du dernier déjà attribué.

**Question 8 :** Écrire la fonction permettant de modifier un abonné à partir de son numéro.

**Question 9 :** Écrire la fonction permettant d'insérer un nouvel exemplaire de livre. On proposera une liste de valeurs possibles pour les éditeurs et les mots-clés, à partir des données déjà présentes dans la base de données, tout en laissant à l'utilisateur la possibilité de rentrer une nouvelle valeur.

**Question 10 :** Écrire la fonction permettant de supprimer un exemplaire de livre à partir de son numéro d'exemplaire.

**Question 11 :** Écrire la fonction permettant de supprimer un abonné.

**Question 12 :** Écrire la fonction permettant d'effectuer le retour d'un livre. Pour cela, on utilisera le numéro de l'exemplaire retourné.

**Question 13 :** Écrire la fonction permettant d'emprunter des livres. On utilisera les numéros de l'abonné et des exemplaires qu'il souhaite emprunter.

**Question 14 :** On souhaite maintenant réaliser la recherche d'ouvrages dans la bibliothèque. On propose aux utilisateurs d'effectuer la recherche à partir :

- d'un nom d'auteur ;
- d'un titre ;
- d'un mot-clé ;
- du genre.

L'utilisateur pourra utiliser un ou plusieurs de ces critères. Les résultats indiqueront si les livres sont en rayon, ou dans le cas où ils seraient empruntés, la date de retour prévue.

# 10 Accès aux bases de données en java

### Exercice 10.1 : Base de données de la bibliothèque

Vous trouverez ci-dessous la *javadoc* de la classe `GestionBD` que vous devez rédiger. Nous vous fournissons le début de cette classe sous forme de fichier `.class` (version compilée). Elle contient :

- les trois attributs ;
- le constructeur ;
- les méthodes :

```
public void setFluxSortie(PrintWriter out)
public PrintWriter getFluxSortie()
public ResultSet executerRequete(String requete) throws SQLException
public int executerUpdate(String requete) throws SQLException
public String[] getTableauNomsAttributsResultSet(ResultSet rs) throws SQLException
public ArrayList <String>  getArrayListTuplesResultSet
    (ResultSet rs, String separateur1,boolean nomsAttributs) throws SQLException
public void fermerConnexion() throws SQLException
```

**Question 1 :** Compléter la classe de test minimale `TestBD` ci-dessous utilisant une instance de `GestionBD` et mettant en œuvre les différentes méthodes de la classe.

```
import java.sql.ResultSet;
import java.util.ArrayList;

public class TestBD {
    public static void main(String[] args) {
        try {
            // ouvrir une connection à la base de données
            GestionBD gt = new GestionBD("jdbc:postgresql://aquanux/........",
                "org.postgresql.Driver","........","........",null);
            System.out.println("connexion reussie");
            // exécuter une requête : récupérer la table livre
            ResultSet rs = ........
            // afficher les noms des attributs sur une ligne
            String[] tab = ........
            ........
            ........
            ........
            // exécuter une requête et afficher les tuples du résultat
            // on affiche les éditeurs
            ........
            ArrayList <String> al = ........
            ........
            ........
            // faire une mise à jour de la base de données
            // en insérant un nouvel éditeur
            String requete = .......
            ........
            // afficher de nouveau les éditeurs pour vérifier que l'insertion
            // a bien eu lieu
```

```
            ........
            ........
            ........
            ........
            // fermer la connection pour terminer
            ........
        } catch(Exception e){e.printStackTrace();}
    }
}
```

**Question 2 :** Compléter la classe `GestionBD`. Dans la mesure où vous ne disposez pas du source, vous devrez déclarer une classe `MaGestionBD` héritant de `GestionBD`. C'est dans cette classe que vous rajouterez les méthodes indiquées dans la javadoc mais pas fournies dans `GestionBD`. N'oubliez pas de tester, en complétant la classe `TestBD` de la question précédente.

**Question 3 :** Compléter la classe de gestion de la bibliothèque, `GestionBibli`, dont le squelette est fourni ci-dessous.

```
import java.util.*;
import java.io.*;

public class GestionBibli{
    public static void main (String[] args){
        MaGestionBD gt = null;
     Scanner sc = new Scanner(System.in);
        PrintWriter pw = new PrintWriter(System.out,true);
        int rep;
        String requete;

        // ouverture de la connexion à la base de données
    try {
         ...
        } catch (Exception e) {System.out.println(e);}

        // Tant que l'utilisateur veut faire des requêtes,
        // lui proposer un menu
        do {
            System.out.println("\n1/ liste de tous les livres");
            System.out.println("2/ suppression d'un livre");
            System.out.println("3/ liste de tous les abonnés");
            System.out.println("4/ inscription d'un nouvel abonné");
            System.out.println("5/ suppression d'un abonné");
            System.out.println("6/ liste des emprunts d'un abonné");
            System.out.println("7/ prêt d'un livre");
            System.out.println("8/ rechercher un auteur");
            System.out.println("9/ remplir une table à partir d'un fichier texte");
            System.out.println("0/ terminer");

            // lecture du choix de l'utilisateur
            ...

            // traitement cas par cas
            switch (rep) {
```

```
                case 1 : // liste de tous les livres
                         ...
                         break;
                case 2 : // suppression d'un livre à partir de son numéro
                         ...
                         break;
                case 3 : // liste de tous les abonnés
                         ...
                         break;
                ...
            }  // end switch
        } while (rep != 0);

        // fermeture de la connexion
        ...
    } // end main
} // end class
```

**ANNEXE**

public class **GestionBD** extends java.lang.Object

## Field Summary

| | |
|---|---|
| private java.sql.Connection | connexion l'objet connexion |
| private java.io.PrintWriter | out fichier texte où sont envoyés tous les résultats. (usage local ou distant) |
| private java.sql.Statement | stmt l'objet statement |

## Constructor Summary

| |
|---|
| GestionBD(java.lang.String url, java.lang.String pilote, java.lang.String gestionnaireBD, java.lang.String motDePasse, java.io.PrintWriter out) throws Exception<br>constructeur : ouvre une connexion puis une session. |

## Method Summary

| | |
|---|---|
| void | ajouterTuple(java.lang.String nomTable, java.lang.String[] tabAttributs) throws SQLException |
| void | copier(java.lang.String nomTable, java.lang.String separateur) throws SQLException<br>simulation de la méta commande \copy table from table.txt |
| void | envoyerNomsAttributsRequete(java.lang.String requete, java.lang.String separateur1, java.lang.String separateur2) throws SQLException<br>envoie dans le fichier **out** les noms des attributs d'une requête |
| void | envoyerNomsAttributsResultSet(java.sql.ResultSet rs, java.lang.String separateur1, java.lang.String separateur2) throws SQLException<br>envoie la méta-structure (les noms des attributs) du Resultset dans le fichier **out** |
| void | envoyerNomsAttributsTable(java.lang.String nomTable, java.lang.String separateur1, java.lang.String separateur2) throws SQLException<br>envoie dans le fichier **out** le nom des attributs d'une table. |
| void | envoyerTuplesRequete(java.lang.String requete, java.lang.String separateur1, java.lang.String separateur2, boolean nomsAttributs) throws SQLException<br>envoie dans le fichier out le résultat d'une requête |

| | |
|---|---|
| void | envoyerTuplesResultSet(java.sql.ResultSet rs, java.lang.String separateur1, java.lang.String separateur2, boolean nomsAttributs) throws SQLException<br>envoie dans le fichier out le ResultSet (avec ou sans les attributs) |
| void | envoyerTuplesTable(java.lang.String table, java.lang.String separateur1, java.lang.String separateur2, boolean nomsAttributs) throws SQLException<br>envoie dans le fichier out le résultat d'une table |
| ResultSet | executerRequete(String requete) throws SQLException |
| int | executerUpdate(String requete) throws SQLException |
| void | fermerConnexion() throws SQLException<br>fermeture de la connexion |
| PrintWriter | getFluxSortie() |
| java.lang.String[] | getTableauNomsAttributsRequete(java.lang.String requete) throws SQLException<br>retourne dans un tableau de String les noms des attributs d'une requête |
| java.lang.String[] | getTableauNomsAttributsResultSet(java.sql.ResultSet rs) throws SQLException<br>retourne le tableau de String de la méta-structure (les noms des attributs) du Resultset |
| java.lang.String[] | getTableauNomsAttributsTable(java.lang.String nomTable) throws SQLException<br>retourne dans un tableau de String les noms des attributs d'une table |
| ArrayList <String> | getArrayListTuplesRequete(java.lang.String requete, java.lang.String separateur1, boolean nomsAttributs) throws SQLException |
| ArrayList <String> | getArrayListTuplesResultSet(ResultSet rs, java.lang.String separateur1, boolean nomsAttributs) throws SQLException |
| ArrayList <String> | getArrayListTuplesTable(java.lang.String table, java.lang.String separateur1, boolean nomsAttributs) throws SQLException |
| void | setFluxSortie(java.io.PrintWriter out)<br>redirige la sortie vers un autre fichier |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |