Contrôle du 26 janvier 2016

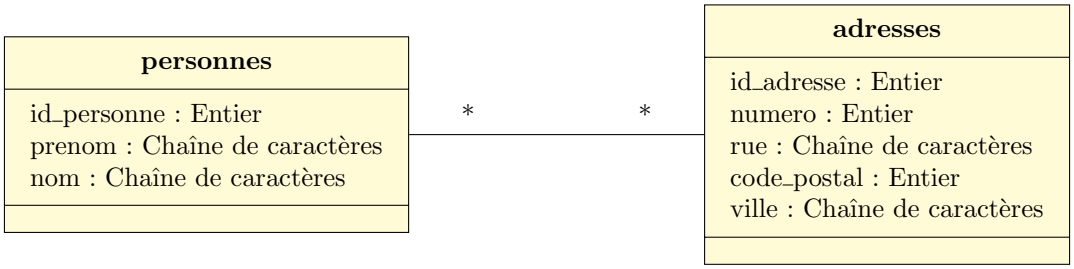
Durée 2h

Les documents sont autorisés. Lisez tout l'énoncé avant de commencer. Respectez bien les instructions. Les exercices sont indépendants, les questions au sein d'un exercice sont en revanche à faire dans l'ordre.

À l'issue du TP, envoyer les fichiers `Repertoire.sql` et `Hierarchie.java` à l'adresse `mathieu.sassolas@u-pec.fr` avec « [MERIT] » dans le sujet.

Exercice 1. Création de tables

Ou souhaite créer un répertoire, c'est à dire une base de donnée stockant des personnes et des adresses selon le schéma suivant :



1. Écrire un fichier `Repertoire.sql` qui crée en PostgreSQL les tables nécessaires. Attention, ce schéma
   — fait apparaître des cardinalités multiples (« ∗ ») ;
   — indique les types de donnée de manière générale et non propre à PostgreSQL ;
   Une première phase de modification du schéma est nécessaire.
   Vous pouvez tester votre script en tapant `psql merit` depuis le dossier contenant votre script et en exécutant `\i Repertoire.sql`.
2. Ajouter les commandes pour insérer dans vos tables les données suivantes :
   — Jean-Claude et Marie-Gertrude Dusse, habitant au 42 rue Victor Hugo, 94000 Créteil.
   — Jean-Claude Dusse (le même !) habite également au 1 rue de la Paix, 01000 Bourg-en-Bresse.
3. *(Cette question est plus difficile que les autres)*
   Construire une vue `repertoire` qui donne les habitants avec leurs adresses.

Exercice 2. La hiérarchie dans l'entreprise

On souhaite coder un programme qui nous donne le nom du supérieur hiérarchique d'un employé, en se basant sur la base `classicmodels`.

1. *(Dossier de travail)* Créer un dossier `ControleM7Ex2` pour cet exercice. Dans la suite, quand on demande de « Récupérer » un fichier, cela signifie qu'il faut le placer dans ce dossier ; Les fichiers en question sont disponibles sur la page `http://www.lacl.fr/~msassolas/enseignement/SQL-Prog_Licence/`, mais il est possible de récupérer des versions téléchargées lors de TP précédents.
2. *(Base de données)*
   a) Récupérer `classicmodelsDB.sql`.
   b) Ouvrir PostgreSQL en exécutant `psql merit`.
   c) Dans l'invité de commande, taper `\i classicmodelsDB.sql`.
   d) On gardera cette session `psql` ouverte afin de pouvoir regarder la structure de la base et des tables avec `\dt` et `\d`, et tester des requêtes. Il faut donc ouvrir un nouveau terminal de se re-placer dans le dossier de travail `ControleM7Ex2`.
3. Récupérer `postgresql-9.4.1207.jar` et `ConnexionMerit.java`.

   a) Compiler `ConnexionMerit` avec la commande `javac ConnexionMerit`.
   b) Exécuter la commande suivante (rien ne devrait s'afficher) :
   ```
   etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar ConnexionMerit
   ```
4. Créer un fichier `Hierarchie.java`. C'est dans ce fichier que l'on écrira le code.
5. On va utiliser le package `java.sql`. Importer toutes ses classes à l'aide de `import java.sql.*;`
6. Définir une classe Hierarchie. Cette classe n'a pas d'attributs.
7. Définir une opération principale `main` avec la signature adéquate. Pour l'instant son contenu est vide.
8. Compiler et exécuter ; pour l'instant, rien ne doit apparaître. Dans les questions suivantes, penser à tester en compilant et exécutant. Pour rappel, les commandes de compilation et d'exécution seront :
   ```
   etudiant@etudiant01$ javac Hierarchie.java
   etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie
   ```
9. Modifier l'opération `main` pour qu'elle vérifie que l'on a bien passé deux arguments. Si c'est le cas, afficher
   ```
   Vous avez recherché la personne suivante:
     Prénom: <premierArgument>
     Nom: <secondArgument>
   ```
   (Chaque commande `System.out.println("... ");` affiche une ligne ; il suffit d'en faire plusieurs pour avoir plusieurs lignes.) Sinon, un message d'erreur doit s'afficher. Par exemple :

```
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie
Erreur: veuiller donner un prénom et un nom!
etudiant@etudiant01$ java-classpath .:./postgresql-9.4.1207.jar Hierarchie Yoshimi Kato
Vous avez recherché la personne suivante:
  Prénom: Yoshimi
  Nom: Kato
```

10. On veut pouvoir chercher le numéro d'un employé à partir de son nom et prénom. (On parle bien de son `employeenumber`.) On rappelle que pour accéder à la table `employees` il faut utiliser `classicmodels.employees` car elle se trouve dans le schéma `classicmodels` ;
    a) En commentaire, donner la requête qui permette de trouver le numéro d'employé de Yoshimi Kato.
    b) Écrire une fonction `reqEmpNumFromName` qui prend en argument deux chaînes de caractères (le prénom et le nom de famille) et qui retourne la requête qu'il faudrait exécuter (sans le point-virgule à la fin) pour trouver le numéro d'employé de la personne désignée par le prénom et le nom donnés en arguments. Par exemple, appeler `reqEmpNumFromName("Yoshimi","Kato")` doit retourner la même requête que vous avez écrit en commentaire. Dans les deux cas, il est possible de tester la requête en la copiant dans l'invité de commande `psql` que vous avez laissé ouvert.
11. Modifier l'opération `main` pour qu'elle affiche également la requête de recherche. Par exemple (ici la requête est remplacée par des points d'interrogation pour ne pas donner la réponse à la question précédente !) :

```
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Yoshimi Kato
Vous avez recherché la personne suivante:
  Prénom: Yoshimi
  Nom: Kato
La requête qu'il faudrait exécuter est:
  ??????????????????????
```

12. On veut écrire une fonction qui récupère le numéro d'employé à partir de son prénom et de son nom. Si on ne trouve pas l'employé, on retourne -1. Le principe est le suivant :
    — On récupère la requête à l'aide de la fonction `reqEmpNumFromName` codée précédemment.

— On récupère un `Statement` avec les commandes suivantes :
```
ConnexionMerit db = ConnexionMerit.obtain("merit","etudiant");
Statement st = db.getStatement();
```
— On exécute la requête sur ce `Statement`. Attention, l'exécution de la requête peut générer une `SQLException` qu'il faut attraper.
— On regarde si il y a des lignes dans le résultat avec `next()` qui retourne alors `true`.
— S'il n'y a pas de résultats, on retourne -1.
— S'il y a des résultats, on retourne le numéro d'employé.

13. Modifier l'opération `main` qui affiche le numéro de l'employé donné en entrée s'il est trouvé et un message d'avertissement sinon. On commentera (mais sans effacer !) les lignes qui affichaient la requête. Par exemple :

```
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Jean Dupont
Vous avez recherché la personne suivante:
  Prénom: Jean
  Nom: Dupont
  Employé non trouvé dans la base.
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Yoshimi Kato
Vous avez recherché la personne suivante:
  Prénom: Yoshimi
  Nom: Kato
  Numéro d'employé: 1625
```

14. *(Cette question est plus difficile que les autres)*
On veut procéder de la même manière pour trouver le nom et prénom du responsable (`reportsto`) d'un employé à partir de son numéro.
a) Écrire en commentaire la requête qui trouve le nom et prénom du supérieur de l'employé au numéro 1625. Attention, le numéro d'employé est une chaîne de caractères (`varchar`)
b) Écrire une opération `reqBossName` qui crée cette requête à partir du numéro d'employé, passé en argument.
c) Écrire une `getBossName` qui effectue cette requête et retourne la concaténation du prénom et du nom du supérieur. Si l'employé ou son supérieur n'est pas trouvé, cette fonction retourne la chaîne vide `""`.
d) Modifier l'opération `main` pour qu'elle affiche également le nom du supérieur, s'il a été trouvé, et un avertissement sinon. Par exemple :

```
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Yoshimi Kato
Vous avez recherché la personne suivante:
  Prénom: Yoshimi
  Nom: Kato
  Numéro d'employé: 1625
  Supérieur: Mami Nishi
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Diane Murphy
Vous avez recherché la personne suivante:
  Prénom: Diane
  Nom: Murphy
  Numéro d'employé: 1002
  Supérieur non trouvé dans la base.
etudiant@etudiant01$ java -classpath .:./postgresql-9.4.1207.jar Hierarchie Jean Dupont
Vous avez recherché la personne suivante:
  Prénom: Jean
  Nom: Dupont
  Employé non trouvé dans la base.
```