Gestion des consommations de café

Objectifs: Ce TP a pour but de vous familiariser avec l'API JDBC (package `java.sql` de l'API JavaSE), vous aborderez ainsi les points suivants :

- Ouvrir une connexion JDBC
- Retrouver l'information de manière sélective
- Insérer des données dans des tables
- Faire une requête quelconque et Obtenir la méta information sur les types de données du résultat

Considérons le scénario suivant :

Il s'agit d'écrire une application Java permettant de suivre la consommation de café des programmeurs d'un projet. Chaque semaine, le chef de projet relève le nombre de tasses de café consommées par les différents programmeurs. Ces informations sont stockées dans un SGBD dans deux tables :

- `PROGRAMMEURS` qui stocke les informations d'identification de chaque programmeur du projet
- `CONSOS_CAFE` qui stocke les consommations de café semaine par semaine des différents programmeurs.

La structure de ces tables est la suivante :

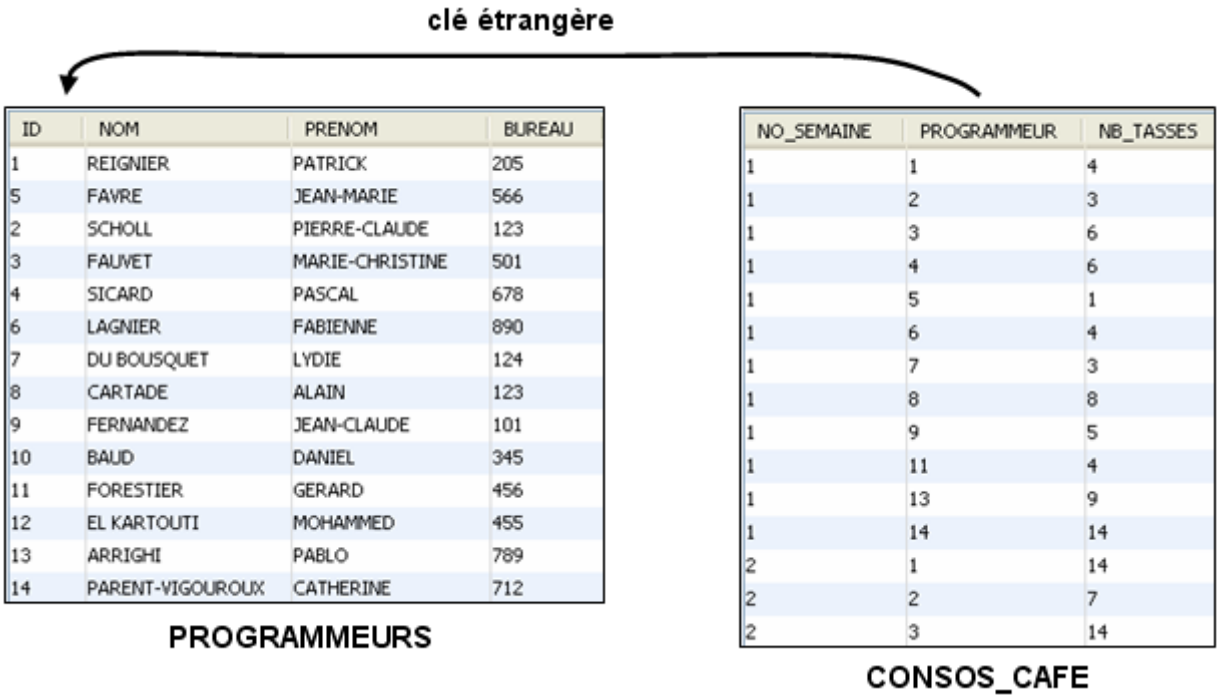

Figure 1: Les tables pour la gestion des cafés.

0. Préambule: Récupérer un pilote JDBC pour Oracle

Le SGBD utilisé pour ce TP, sera la base de données Oracle installé votre serveur en locale. Pour pouvoir dialoguer avec ce serveur depuis vos programmes Java vous avez besoin d'un pilote (*driver*) JDBC pour Oracle.

Exercice 0: télécharger le pilote JDBC pour Oracle

Le fichier `ojdbc7.jar` contient le bytecode des différentes classes d'un driver JDBC pour la base de données Oracle 12c que vous avez installé..

1. Créer la base de données en utilisant les outils intégrés à NetBeans

Avant de vous intéresser au programme de gestion des consommations de café proprement dit, il vous faut créer les différentes tables dans la base de données. Plusieurs moyens sont à votre disposition. Par exemple, vous pouvez utiliser directement l'interface `sqlplus` sur le serveur `oracle`.

A ce moyen plutôt primitif, on préférera l'utilisation des outils de à NetBeans qui offrent une interface autrement plus conviviale (et efficace). Ces outils basés sur JDBC (NetBeans est écrit entièrement en Java) permettent à un développeur d'accéder directement aux bases de données nécessaires au développement de ses applications sans quitter son environnement de programmation. Avec ceux-ci il est possible de :

- Consulter, créer, détruire des tables de la base de données.
- Visualiser et modifier le contenu des tables.
- Ecrire et exécuter des commandes SQL dans un éditeur spécialisé (coloration syntaxique, complétion automatique...).

Exercice 1:

Effectuez le tutoriel du document tuto1.pdf fourni qui vous permettra de créer les tables `PROGRAMMEURS` et `CONSOS_CAFE` et de découvrir les fonctionnalités de base des outils BD de Netbeans.

2. L'application GestionCafes

La base de données ayant été créée, il est maintenant temps de passer à la réalisation de l'application de gestion des consommations de cafés.

Exercice 2: Créer le projet java dans Netbeans

1. Dans Netbeans créez un nouveau projet d'application Java que vous nommerez par exemple `TP_JDBC`
2. Pour vous aider à réaliser ce TP, un squelette de l'application vous est fourni dans le fichier `GestionCafes.java`. Récupérez ce fichier et intégrez-le aux sources de votre projet.
3. Vérifiez que cette classe se compile et s'exécute correctement.

4. Regardez le code de la classe `GestionCafes` pour comprendre la structure de l'application.

2.1 Ouvrir la connexion JDBC avec Oracle

Le programme principal de l'application `GestionCafes` a la structure suivante:

```
Ouvrir une connexion JDBC à Oracle.
répéter
    afficher un menu proposant différentes opérations à
l'utilisateur
    effectuer l'opération choisie
tant que l'utilisateur veut continuer
Fermer la connexion
```

Pour le moment, les opérations liées à la Base de données, dont l'ouverture et la fermeture des connexions, ne sont pas encore implémentées.

Exercice 3: Ajoutez à la classe `GestionCafes` le code Java permettant d'ouvir une connexion à la base Oracle et de la fermer à la fin de l'exécution.

- Si la connexion n'a pu être ouverte. Le programme est interrompu et la cause de l'erreur est affichée sur l'écran.
- Si la connexion a pu être ouverte, un message le signale à l'utilisateur et le menu proposant les différentes opérations possibles est affiché.
- Ne pas oublier de fermer la connexion à la fin de l'exécution du programme.

Compiler et tester votre programme.

Attention, pour que votre programme fonctionne n'oubliez pas d'intégrer le pilote jdbc pour Oracle (`ojdbc7.jar`) dans le classpath de votre application lors des exécutions. Pour vous aider dans cette tâche vous pouvez consulter le tutoriel tuto2.pdf fourni.

2.2 : Rechercher de l'information dans la base de données

L'application `GestionCafes` permet à l'utilisateur de consulter les informations suivantes:

- **le programmeur ayant consommé le nombre maximum de cafés en une semaine et sa consommation pour cette semaine.**
  Une requête SQL permettant de récupérer cette information est

```
SELECT PROGRAMMEUR,PRENOM,NOM,NB_TASSES,NO_SEMAINE FROM CONSOS_CAFE c
       JOIN  PROGRAMMEURS p ON p.ID=c.PROGRAMMEUR WHERE
       c.NB_TASSES=(SELECT MAX(NB_TASSES) FROM CONSOS_CAFE)
```

- **pour un programmeur donné le nombre total de tasses de café consommées.**
  Une requête SQL permettant de récupérer cette information est

```
SELECT SUM(NB_TASSES) FROM CONSOS_CAFE WHERE PROGRAMMEUR=idProg
```

Où ***idProg*** est l'identifiant du programmeur concerné.

- **pour une semaine donnée la liste des programmeurs triée dans l'ordre décroissant selon leur nombre de consommations.**
  Une requête SQL permettant de récupérer cette information est

```sql
SELECT PROGRAMMEUR,  PRENOM, NOM, NB_TASSES FROM
       CONSOS_CAFE c JOIN PROGRAMMEURS p ON c.PROGRAMMEUR=p.ID
       WHERE c.NO_SEMAINE=numSem
       ORDER BY NB_TASSES DESC
```

Où ***numSem*** est le numéro de la semaine concernée.

Exercice 4:

Modifiez le code des méthodes `plusGrosConsommateurs()` `nbreTotalDeTasses()` et `consomationsPourUneSemaine()` de la classe `GestionCafes` afin d'implémenter ces opérations de consultation.

Pour exécuter les instructions SQL `SELECT` depuis le code Java, utilisez la méthode `executeQuery` de l'interface `Statement` qui retourne des résultats sous la forme de lignes de données dans un objet `ResultSet`. Les résultats sont examinés ligne par ligne en utilisant les méthode `next()` et `getXXX()` de l'interface `ResultSet`.

2.3 : Insérer des informations dans la base de donnée

L'application `GestionCafes` doit permettre à l'utilisateur de saisir et d'enregistrer dans la base de données les consommations des programmeurs pour une semaine donnée.

Le programme permet de saisir un numéro de semaine et ensuite pour chaque programmeur de rentrer le nombre de tasses qu'il a consommées durant cette semaine.

Une requête SQL permettant d'enregistrer cette information est

```sql
INSERT INTO CONSOS_CAFE (NO_SEMAINE, PROGRAMMEUR, NB_TASSES)
       VALUES(numSem, idProg, nbTasses)
```

où

- ***numSem*** est le numéro de la semaine concernée.
- ***idProg*** est l'identifiant du programmeur.
- ***nbTasses*** est le nombre de tasses qu'il a consommé durant cette semaine.

Exercice 5: Modifiez le code de la méthode `saisirConsommations()` de la classe `GestionCafes` afin d'implémenter ces opérations d'insertion de données dans la base.

Pour exécuter les instructions SQL `INSERT` depuis le code Java, utilisez la méthode `executeUpdate` de l'interface `Statement` qui retourne un entier (le nombre de lignes modifiées dans la table).

Afin de ne pas reconstruire un nouvel objet `Statement`pour insérer les données de chaque programmeur, il peut être judicieux d'utiliser un objet de type `PreparedStatement`.

2.4 : Effectuer une requête libre et obtenir la méta information sur les types de données du résultat

L'application `GestionCafes` offre à l'utilisateur la possibilité d'exécuter n'importe quelle requête SQL qu'il peut entrer directement au clavier. Le résultat affiché dépendra de la nature de la requête.

- Si la requête est type `SELECT` les informations suivantes sont affichées:
  - le nombre de colonnes de la table résultat est affiché,
  - pour chaque colonne, son nom et le type des données est affiché,
  - le contenu de la table est affiché ligne par ligne.

Par exemple, si la requête est :

```
1 | SELECT * FROM PROGRAMMEURS ORDER BY ID
```

le résultat affiché sera :

```
le résultat contient 4 colonnes
-------------------------------
Colonne : 1
NOM ID TYPE : NUMBER
-------------------------------
Colonne : 2
NOM NOM TYPE : VARCHAR2
-------------------------------
Colonne : 3
NOM PRENOM TYPE : VARCHAR2
-------------------------------
Colonne : 4
NOM BUREAU TYPE : NUMBER
-------------------------------
Résultats de la requête

1 REIGNIER PATRICK 205
2 SCHOLL PIERRE-CLAUDE 123
3 FAUVET MARIE-CHRISTINE 501
4 SICARD PASCAL 578
5 FAVRE JEAN-MARIE 566
...
```

alors que la requête :

```
1 | SELECT SUM(NB_TASSES) FROM CONSOS_CAFE WHERE NO_SEMAINE = 1
```

conduira à l'affichage suivant :

```
le résultat contient 1 colonnes
-------------------------------
Colonne : 1
NOM SUM(NB_TASSES) TYPE : NUMBER
-------------------------------
Résultats de la requête

25
```

- Sinon (la requête correspond à un `UPDATE`, `INSERT`, `DELETE`, `DROP TABLE`, `CREATE TABLE`, ..) le résultat affiché est alors le nombre de lignes modifiées dans la table concernée.

  Par exemple, si la requête est :

```
1 | INSERT INTO CONSOS_CAFE(NO_SEMAINE, PROGRAMMEUR, NB_TASSES) VAUES (41,2,10)
```

  le résultat affiché sera :

```
Nombre de lignes modifiées 1
```

Exercice 6: Modifiez le code de la méthode `requeteLibreEtMetaDonnees()` de la classe `GestionCafes` afin d'implémenter les opérations permettant de supporter ces requêtes libres.

Pour exécuter une requête SQL libre depuis le code Java, vous devrez utiliser la méthode `execute` de l'interface `Statement` qui retourne un booléen à **vrai** si la requête est de type `SELECT`, **false** sinon.

Si la requête a produit un `ResultSet`, vous devrez utiliser les méta-données de celui-ci (classse `ResultSetMetaData`) pour obtenir les descriptions des colonnes.

2.5: Chargement des données (Batch Updates)

Avant de disposer de l'application `GestionCafes` le manager utilisait un tableur pour saisir les consommations de cafés. Il souhaiterait pouvoir exploiter ces données avec l'application `GestionCafes`. Les données pour l'année 2013 ont été exportées dans deux fichiers CSV (Comma Separated Values), fichiers textes où les valeurs sont séparées par des virgules:

- `progs.data` pour la liste des programmeurs,
- `consos.data` pour la liste des programmeurs.

Le format des fichiers CSV est le suivant :

- La première ligne contient les descriptions des colonnes de la table sous la forme *NomDeLaColonne:TypeSQLDeLaColonne* séparées par des virgules. Par exemple pour la table PROGRAMMEURS on aura :

```
ID:NUMBER,NOM:VARCHAR2,PRENOM:VARCHAR2,BUREAU:NUMBER
```

- Les lignes suivantes contiennent les valeurs des enregistrements présents dans la table, valeurs séparées par des virgules. Par exemple un enregistrement dans la table PROGRAMMEURS sera écrit sous la forme :

```
3,FAUVET,MARIE-CHRISTINE,501
```

Exercice 7: Modifiez l'application `GestionCafes` pour lui ajouter une fonctionnalité d'import des données à partir des fichiers texte au format CSV

Il s'agit de lire les fichiers `progs.data` et `consos.data` et pour chaque ligne d'effectuer l'opération d'insertion correspondante dans la base de données.

Le nombre de données étant relativement important (1612 lignes pour `consos.data`), pour des raisons d'efficacité il vous est recommandé d'utiliser des Batch Updates plutôt que de faire les insertions avec de simple objets `Statement`et des messages `executeUpdate()` (pour en savoir plus, voir le cours).

Exercice 8: De manière symétrique, modifiez l'application `GestionCafes` pour lui ajouter une fonctionnalité permettant d'exporter les données d'une table vers un fichier texte au format CSV.

2.6 Configuration du driver de la base de données à partir d'un fichier de propriétés (*properties file*)

Il est dommage de "mettre en dur" dans le programme les paramètres de configuration de la base donnée (nom du driver, url de connexion). En général, il vaut mieux isoler ces paramètres dans un fichier de configuration afin de pouvoir les changer facilement et ne pas avoir à modifier et recompiler le code Java en cas de changement de la base de données (par exemple basculement d'une base de données de tests vers une base en production).

Exercice 9:

Modifiez votre programme afin qu'il puisse lire les chaînes décrivant la BD dans un fichier de propriétés (voir la classe `java.util.Properties`)