

Pour mettre en place l'environnement nécessaire à JDBC et à SQLJ, on mettra en place les variables d'environnement suivantes (si vous utilisez un autre shell que `csh` ou `tcsh`, remplacez `setenv` par `export` et le caractère espace entre nom et valeur des variables par un `=`) :

1. Pour les étudiants MIAGE :

```
setenv CLASSPATH $ORACLE_HOME/oc4j/sqlj/lib/translator.jar:\
                 $ORACLE_HOME/sqlj/lib/runtime12.jar:\
                 $ORACLE_HOME/jdbc/lib/classes12.zip:.
setenv PATH $ORACLE_HOME/oc4j/bin:$PATH
```

2. Pour les étudiants M1 :

```
setenv CLASSPATH $ORACLE_HOME/sqlj/lib/translator.jar:\
                 $ORACLE_HOME/sqlj/lib/runtime12.jar:\
                 $ORACLE_HOME/jdbc/lib/classes12.zip:.
```

On exécutera alors le traducteur SQLJ par `sqlj`. Les exercices sont indépendants, et chacun nécessitera l'écriture d'un programme Java ou SQLJ.

1 Curseurs et type ref cursor en JDBC

1. Dans SQL*Plus, créer une table `personne` contenant des noms et prénoms de personne. Insérer quelques valeurs dans cette table.
2. Écrire une fonction stockée PL/SQL renvoyant un curseur vers l'ensemble des noms de personne ayant pour prénom une valeur passée en argument. On déclarera le type du curseur au préalable avec :

```
CREATE OR REPLACE PACKAGE Types AS
  TYPE cursor_type IS REF CURSOR;
END Types;
/
```

3. Exécuter cette fonction stockée depuis JDBC, et parcourir le curseur renvoyé pour afficher l'ensemble des noms de personne ayant un prénom donné.

2 Gestions des erreurs en JDBC

1. Écrire une fonction Java `printException` prenant en argument un `SQLException` et affichant toutes les informations qu'il contient (`sqlstate`, `sqlcode`, `message`).
2. Écrire une fonction Java `safeExec` prenant en argument une chaîne de caractères contenant un ordre SQL, et exécutant cet ordre au travers d'un `try/catch`. Dans le cas où une exception est levée, `printException` sera appelé pour afficher l'exception. Dans le cas où des warnings sont présents, `printException` sera appelé pour les afficher également.
3. Écrire une fonction Java `main` contenant le nécessaire pour se connecter à la base Oracle.
4. À l'aide de `safeExec` et de `printException`, exécuter dans le `main` des ordres SQL renvoyant des erreurs et afficher les erreurs correspondantes. On essaiera de reproduire les types suivants d'erreur :
 - (a) une erreur de syntaxe
 - (b) une erreur de schéma (p.ex. table inexistante)

- (c) des erreurs de typage :
 - type de colonne incorrect
 - troncation de valeur
- (d) une erreur de violation de contrainte
- (e) un avertissement résultant d'une erreur de compilation d'une procédure
- (f) une erreur d'authentification lors de la connexion à la base
- (g) une erreur de privilèges ; on pourra par exemple essayer de créer une table appartenant à un autre utilisateur.
- (h) une erreur de concurrence ; on pourra par exemple provoquer un LOCK NOWAIT.
- (i) une erreur personnalisée, générée par un RAISE_APPLICATION_ERROR à partir d'un bloc PL/SQL.

3 Métadonnées en JDBC

1. Écrire une fonction Java permettant d'afficher le nom et type (table ou vue) de l'ensemble des tables et vues d'un utilisateur donné. On utilisera l'appel suivant :

```
String list[]={ "TABLE", "VIEW" };
ResultSet tables=dbmd.getTables(null,login,"%",list);
```

et on n'affichera pas les tables interne à Oracle, dont le nom commence par BIN\$. Tester.

2. Écrire une fonction Java prenant en argument un nom de table, et affichant cette table comme suit. Afficher sur une ligne le nom et le type de chaque colonne. Afficher ensuite, ligne par ligne, le contenu de cette table. Tester sur diverses tables.

4 SQLJ

On reprend les exercices du TD7 en JDBC, mais cette fois ils sont faits en SQLJ. Comparer les avantages et inconvénients pour chaque réponse. Récupérer `MenuSQLJ.sqlj` ; le but de l'exercice est de programmer les fonctions laissées vides, chaque fonction correspondant à une question. On reprendra le code de création des table, fonction et procédure du TD précédent.

4.1 Connexion

1. Écrire la fonction de connexion SQLJ `connexion`.
2. Écrire la fonction de déconnexion SQLJ `deconnexion`.

4.2 Exécution d'ordres BD

1. Requête. Dans la fonction `selectInto`, à l'aide d'une instruction `SELECT ... INTO`, afficher le nombre de lignes de la table `t`.
2. Ordres SQL non requête.
 - (a) Dans la fonction `miseAJour1`, écrire un ordre SQL incrémentant de 1 l'âge des personnes dont le nom est `toto` ou dont l'âge est inférieur à 22.
 - (b) Même question pour `miseAJour2`, mais le nom et l'âge sont lus au clavier.
3. Appel de procédures stockées

- (a) Dans la fonction `appelProc`, exécuter la procédure stockée PL/SQL *rajeunit* avec un paramètre lu au clavier.
- (b) Dans la fonction `appelFonct`, exécuter la fonction stockée PL/SQL *f*.

4.3 Curseurs

Dans la fonction `curseur`, afficher les noms des personnes ayant strictement plus de 20 ans (donc sans paramètre).

4.4 SQL dynamique

Dans la fonction `sqldyn`, détruire une table dont le nom est lu au clavier.