TD

version 18 avril 2018 (TD 1b-10b)

TD/TP 1b

1 JDBC : exécution d'ordre SQL non select

1. Ecrire un programme complet et autonome Java/JDBC qui effectue l'action 1 du cahier des charges (employé : créer un village). Pour simplifier il n'y aura aucune lecture au clavier.
   Indication : Inspirez-vous de l'exemple du cours.

2. *Structure du TD.* Le but du TD est de programmer en Java/JDBC les dix actions du CC. Pour simplifier toutes les actions seront faites sur le compte du programmeur, et on ne vérifiera ni les contraintes, ni les paramètres.

   (a) Récupérez et exécutez le corrigé du dernier TD qui permet en particulier de peupler la base.

   (b) Récupérez sur la page du cours, compilez et exécutez *Menu.java* (il a été distribué sur papier en cours). Entrez les différents choix possibles pour vous familiariser avec.

   (c) Ce TD va consister a écrire le corps des fonctions Java du menu ci-dessus, et à en rajouter d'autres bien sûr. Attention : cela peut nécessiter d'adapter leurs paramètres et leurs valeurs de retour. Dans tout le TD JDBC on n'écrira aucune fonction PL/SQL, mais on appellera celles qu'on a écrites lors des TD sur PL/SQL quand l'énoncé le demandera explicitement.

   (d) Attention : Dans tous les exercices, les éventuels paramètres des actions seront lus au clavier dans la fonction (donc pas passés en paramètres dans *Menu.java*), et leurs valeurs de retour affichées (donc pas renvoyées).

   (e) Attention : il est interdit de rattraper les erreurs (c'est à dire *try-catch* interdit), sauf si c'est demandé dans l'énoncé.

   (f) Testez bien sûr chaque fonction, y compris en vérifiant sous un client interactif.

3. *Pb MP : exécution d'ordre SQL (non curseur).* Ecrivez la fonction Java effectuant l'action des employés : créer un village. Les valeurs des colonnes de *Village* sont lues au clavier (comme indiqué plus haut), sauf *idv* bien sûr.
   Indication : Il s'agit de l'exercice ci-dessus, mais maintenant dans une fonction du menu. Récupérez les parties utiles de votre code et adaptez-les. Placez dans le menu comme dans l'exemple du cours le chargement du pilote et la connexion, sans utiliser la fonction *connexion*.

TD/TP 2b

2 JDBC : compte-rendu restreint d'exécution, connexion, portabilité

Ce TD n'utilise pas le matériel du Bloc 2a (contraintes SQL, séquences).

1. *Pb MP : compte-rendu restreint d'exécution d'ordre SQL (non select).* Ecrivez la fonction *traitement3* effectuant l'action des employés : traitement 3 (comme indiqué plus haut, n'appelez pas la procédure PL/SQL correspondante).

2. *Problème du mode programme : connexion.*

   (a) Ecrivez la fonction Java de connexion a votre compte SGBD, et appelez-la en début du *main*. Pour simplifier son utilisation, codez en dur dedans les paramètres de connexion. Attention : il est interdit d'utiliser un champ statique.

   (b) Ecrivez la fonction Java de déconnexion.

3. *Pb MP : connexion (et ordre SQL non select).* Ecrivez une fonction qui insère un village dans votre table *Village*, et qui en insère aussi un autre dans celle du compte de l'enseignant chargé de cours. Attention : vous n'avez pas les droits sur cette dernière table, utilisez donc son compte SGBD, y compris pour tester sous le client interactif.

4. *Pb MP : portabilité des programmes Java/JDBC entre différents SGBD.* On considère MySQL (ou un autre SGBD au choix).

   (a) *Sur table.* Adaptez l'exemple de l'exercice ci-dessus pour vous y connecter, en conservant tel quel le reste de votre programme.

   (b) *Facultatif.* Si vous avez ce SGBD ou un autre sur votre ordinateur personnel, faites tourner cet exercice. Constatez qu'aucune modification du code Java/JDBC n'est nécessaire hors la spécification du pilote et du compte SGBD, puis que l'exécution de l'ordre SQL fonctionne telle quelle.

5. *Pb MP : exécution d'ordre SQL (non select).* Ecrivez la fonction Java effectuant l'action des employés : modifier un village, comme indiqué ci-dessous. Remarque : les concepts et leur mise en œuvre sont les mêmes que pour les exercices 1.3 et 2.1 (il n'y a pas de nouveau concept ni outil dans cet exercice), et c'est la dernière des actions sans interrogation du CC concernées par JDBC (remarque : l'action "système" ne l'est pas).

   (a) On suppose trois paramètres : l'identifiant, la capacité et l'activité.

   (b) *Facultatif.* Prévoyez toutes les variantes vues dans le corrigé du Bloc 1a (mises à jour).

6. *Facultatif. Pb MP : connexion ; pb BD : confidentialité.* Dans quelques blocs, lorsque la confidentialité aura été traitée en BD2, adaptez votre programme Java/JDBC pour utiliser les comptes employés et clients décrits dans le CC.

TD/TP 3b

3 JDBC : gestion des erreurs, curseurs (1/2), séquences

Ce TD utilise le matériel du Bloc 2a (contraintes SQL, séquences).

1. *Pb MP : gestion des erreurs.*
   (a) Exécutez votre programme en faisant quelques tests en lui entrant des valeurs créant des erreurs, par exemple :
      i. déconnexion puis traitement 3,
      ii. traitement 3 après avoir détruit la table séjour,
      iii. insertion d'une chaîne trop longue.
   (b) On reprend la fonction Java effectuant le traitement 3, mais cette fois il est interdit d'utiliser *"throws"*. Affichez bien sûr le message d'erreur de Java, précédé d'un message personnel. Testez avec les mêmes exemples que ci-dessus.
   Remarquez qu'ici les erreurs ne sont pas "gérées" à proprement parler, mais plutôt qu'on tient compte du fait qu'elles peuvent survenir. Dans toute la suite, à nouveau il est interdit de rattraper les erreurs (c'est à dire *try-catch* interdit), sauf si c'est demandé dans l'énoncé.
2. *Pb MP : exécution d'ordre SQL : curseur (1/2).*
   (a) Ecrivez la fonction effectuant l'action des clients *authentification* : consulter son (éventuelle) ligne dans la table *Client*, à partir de son nom et de son prénom et afficher le message correspondant (partez du corrigé du TD1a).
   (b) Ecrivez la fonction effectuant l'action des employés : consulter les villages (n'appelez pas la procédure PL/SQL).
   (c) Ecrivez la fonction effectuant l'action des clients : traitement 1 (n'appelez pas la procédure PL/SQL).
   (d) Ecrivez la fonction effectuant l'action des clients : consultation des lignes lui correspondant dans les tables *Séjour* et *Village* (n'appelez pas la procédure PL/SQL correspondante).
   (e) *Entraînement personnel.* Ecrivez la fonction effectuant l'action des clients : consultation des villages sans séjours. (Et quand le chapitre correspondant aura été traité, vous utiliserez bien sûr la vue.)
3. *Pb BD : contraintes et séquences.* Recodez tous les exercices JDBC des TDs 1b et 2b en utilisant l'outil *séquences* du Bloc 2a sur les contraintes SQL chaque fois que c'est pertinent : Exercice 3 du TD1b (on ignore l'exercice 1), et Exercice 2 du TD2b.

TD/TP 4b

TD/TP 4b

4 JBDC : appel de procédures stockées, curseurs (2/2)

1. *Pb MP : exécution d'ordre BD non SQL : appel de procédure stockée.*

   (a) Récupérez sur la page de l'UE les procédures stockées PL/SQL suivantes, afin de pouvoir les appeler depuis JDBC.

      i. La procédure *créerVillage* qui crée un village à partir de ses paramètres, sans rien renvoyer (en particulier pas le nouvel identifiant).

      ii. La version simplifiée (en-tête correct mais corps "vide") de la *fonction traitement3*, et de la *procédure traitement3_ out*

   (b) Ecrivez la fonction Java qui appelle la procédure stockée PL/SQL *créerVillage*.

   (c) Ecrivez la fonction Java qui appelle la *fonction* stockée PL/SQL *traitement3*.

   (d) Ecrivez la fonction Java qui appelle la *procédure* stockée PL/SQL *traitement3_ out*.

   (e) Ecrivez la fonction Java qui appelle la procédure PL/SQL *traitement2*. (Si vous n'avez pas encore écrit cette procédure PL/SQL, écrivez-en vous-même une version avec l'en-tête correct mais un corps "vide" afin de pouvoir l'appeler.)

   (f) *Entraînement personnel.* Ecrivez la fonction Java qui appelle la fonction PL/SQL *traitement1*.

2. *Pb MP : exécution d'ordre SQL : curseur (2/2).*

   (a) Ecrivez la fonction effectuant l'action des clients : traitement 2 (n'appelez pas la procédure PL/SQL correspondante).

TD/TP 5b

5 JBDC : exécution répétée d'ordres BD avec paramètres

1. *Pb MP : exécution répétée d'ordre SQL avec paramètres.*
   On reprend l'exercice 1.3 (création d'un village, TD 1b), mais maintenant on boucle pour créer des villages tant que l'utilisateur le souhaite. Ecrivez cette fonction Java.
   Indication : Ecrivez une version utilisant le nouveau matériel vu dans ce bloc, et une autre avec l'ancien matériel, puis comparez.

2. *Pb MP : exécution répétée d'ordre BD non SQL avec paramètres.*
   Ecrivez une fonction Java qui boucle en appelant les procédure stockées PL/SQL suivantes.

   (a) La procédure *créerVillage*.

   (b) La fonction *traitement3*.

   (c) La procédure *traitement3_out*.

   Indication : Utilisez le nouveau matériel vu dans ce bloc.

JBDC : entraînement personnel global

1. Assurez-vous que vous avez terminé tous les exercices obligatoires de tous les TD.

2. Reprenez le CC de la Bibliothèque et écrivez les trois traitements "directement" (sans appel de procédure stockée), puis en appelant les procédures stockées correspondantes (pour ces appels écrivez vous-même une version des procédures PL/SQL avec l'en-tête correct mais un corps "vide" afin de pouvoir les appeler.)

TD/TP 6b

6 Appel en JBDC de procédures PL/SQL renvoyant un curseur

6.1 PL/SQL : procédures renvoyant un curseur

On considère la procédure *consulter_sejours*, qui prend en paramètre l'identifiant d'un client et affiche ses séjours. On veut l'adapter pour qu'elle *renvoie* le curseur au lieu d'afficher son contenu. Récupérez-en la version 2 (*consulter_sejours2*) dans le corrigé du TD4a sur la page du cours (on pourrait indifféremment utiliser toute autre version de *consulter_sejours*, mais dans ce TD c'est celle-là qu'on utilisera).

1. Ecrivez une *fonction* (pas une procédure) *renvoyer_sejours* qui prend l'identifiant d'un client et renvoie ses séjours dans un curseur (*"refcursor"*). Compilez-la (sans erreurs...), et vous la testerez grâce à la procédure de la question suivante.
2. Ecrivez une procédure *consulter_sejours_refcurseur* qui prend en paramètre l'identifiant d'un client, appelle *renvoyer_sejours*, et affiche les lignes du curseur renvoyé par *renvoyer_sejours*. Testez.
3. *Entraînement personnel.* Reprenez les deux questions ci-dessus, mais en écrivant une *procédure* *renvoyer_sejours_proc* au lieu d'une fonction.

6.2 JBDC : appel de procédures renvoyant un curseur

1. On considère la *fonction* (pas la procédure) PL/SQL *renvoyer_sejours* de l'Exercice 6.1.1 ci-dessus, qui renvoie un curseur contenant les séjours d'un client. Insérez dans votre menu une fonction Java qui appelle cette fonction et affiche les lignes qu'elle renvoie.

TD/TP 7b

7 PL/SQL : SQL dynamique, méthode 1

1. Ecrivez ce qui suit.
   (a) Une procédure *détruire* qui prend en entrée un nom de table et détruit cette table (*drop*).
   (b) Un bloc éphémère qui ne lit rien au clavier et détruit les tables du cahier des charges.
2. Ecrivez une procédure *générique1* qui prend en entrée un nom de procédure p et un entier n, et appelle p sur n.

   Indications : procédez comme suit.
   (a) Identifiez l'exemple du cours le plus proche de cet exercice.
   (b) Modifiez-le pour qu'il réponde à cet exercice : paramètres, ordres BD concernés (SQL et appels de procédure).
   (c) Testez en passant en entrée *consulter_informations* (voir corrigé TD 4a) et un entier.
3. Ecrivez une procédure *affiche_gen*, qui prend en entrée le nom d'une table et de deux de ses colonnes, supposées respectivement de type *varchar2(10)* et *int*, et affiche les valeurs de ces colonnes pour toutes les lignes de cette table. Testez (par exemple avec *Client, nom, age*).

TD/TP 8b

8 PL/SQL : SQL dynamique, méthode 2

1. *Appel dynamique de procédure/fonction PL/SQL.*

   (a) Ecrire une procédure PL/SQL qui prend en entrée un nom de fonction *f:int→ int* et un entier *n*, et appelle *f* sur *n*. Testez avec *traitement3*.

   (b) Même question en prenant un nom de procédure. Testez avec *traitement3Proc*.

2. *Répétition d'ordre SQL dynamique avec variables.* Ecrivez une procédure *duplique_ gen* qui prend en entrée le nom d'une table et de ses quatre colonnes, supposées respectivement de type *int, varchar2(10), int* et *int*, et fait ce qui suit. Elle crée d'abord une table de même nom et même colonnes mais "en ajoutant *2* partout", par exemple : *t(a,b,c,d)* → *t2(a2,b2,c2,d2)*, puis elle recopie dedans le contenu de la première table, ligne par ligne (sans utiliser *create table as select* ni *insert into table select*). Testez-la sur la table *Client*.

3. On considère la procédure *traitement2*. On veut maintenant écrire une procédure *traitement2_ gen*, qui prend les mêmes paramètres, ainsi qu'une chaîne de caractères *l* qui représente un nom d'utilisateur SGBD, et appelle, sur ces paramètres, la procédure *traitement2* de l'utilisateur de nom *l*. Attention : vous devez *appeler* cette procédure, et non la réécrire.

   Ecrivez la procédure *traitement2_ gen* (inspirez-vous de l'exercice antérieur sur la confidentialité des procédures PL/SQL). Testez-la en l'appelant avec le login de votre voisin. Vérifiez le résultat. A qui sont les tables modifiées ?

4. Dans cet exercice, on se pose la même question que ci-dessus, mais en appelant cette fois la fonction *traitement1* ; attention : il s'agit bien d'une *fonction*, et pas d'une procédure.

   Ecrivez une procédure *traitement1_ gen*, qui prend en paramètre un nom d'utilisateur SGBD *l*, et appelle la fonction *traitement1* de l'utilisateur de nom *l*. Testez avec votre voisin.

5. *Erreurs fréquentes.* Reprenez la liste de procédures fausses de la feuille d'exemples, comppilez-les et/ou exécutez-les, puis codez la version correcte.

TD/TP 9b

9 JBDC : SQL dynamique

L'un des objectifs principaux dans tout ce TD est de comparer soigneusement le code Java/JDBC écrit au code PL/SQL utilisant SQL dynamique écrit dans les deux TD précédents, afin d'identifier toutes les similarités et différences. Les numéros d'exercices cités ne font pas référence à la version papier distribuée, mais à la version des TD sur la page du module.

1. *Pb MP : SQL dynamique.*
   (a) On reprend l'exercice 7.1 (PL/SQL, SQL dynamique, méthode 1). Ecrivez les fonctions Java suivantes.
      i. Une sans paramètre qui détruit (*drop*) la table t (on suppose que cette table existe).
      ii. L'une qui lit un nom de table et détruit cette table (*drop*). Comparez au code PL/SQL du corrigé de cet exercice.
      iii. L'autre qui ne lit rien et détruit les tables du cahier des charges. Comparez au code PL/SQL du corrigé de cet exercice.
   (b) On reprend l'exercice 8.3 (PL/SQL, SQL dynamique, méthode 2). On considère la procédure PL/SQL *traitement2*. Ecrivez une fonction Java qui prend en entrée une chaîne de caractères l qui représente un nom d'utilisateur SGBD, et appelle la procédure PL/SQL *traitement2* de l'utilisateur de nom l (en lisant les paramètres de *traitement2* au clavier bien sûr).
   Comparez au code PL/SQL du corrigé de cet exercice.
   (c) On reprend les exercices 7.2 et 8.4. Ecrivez une fonction Java qui fait ce qui suit. Elle prend en entrée deux chaînes de caractères : l'une l qui représente un nom d'utilisateur SGBD, et l'autre f qui représente un nom de fonction PL/SQL supposés prendre en entrée une chaîne et un entier, et renvoyer un entier. Elle appelle la fonction f de l (en lisant ses paramètres au clavier bien sûr), et affiche son résultat. Appelez-la sur la fonction PL/SQL *traitement1*.
   Comparez aux codes PL/SQL des corrigés de ces exercices.
2. *Entraînement personnel. Pb MP : appel de procédure.* Pour varier le nombre de paramètres *in* et *out* des procédures PL/SQL, écrivez des procédures et des fonctions PL/SQL avec 0, 1 ou 2 paramètres *in* et/ou 0, 1 ou 2 parmaètres *out*, et appelez-les toutes depuis une seule fonction Java/JDBC.

TD/TP 10b

10 Révisions et synthèse

10.1 Rattrapage pour étudiants en retard

Avant d'aborder la section *Annales*, faites obligatoirement les exercices suivants, dans cet ordre. Vous devez bien sûr les taper, les compiler, les exécuter sur différents exemples.

1. Terminez le TD 8b.
2. Terminez le TD 9b.
3. *Travail personnel.* Dans les exemples de cours du Bloc 9b, une liste d'exemples bugués ont été fournis. Pour chacun, tapez-le, compilez-le et exécutez-le, et constatez le bug en comprenant le message d'erreur. Puis écrivez le code correct correspondant.

10.2 Annales

Attention : avant d'aborder cette partie, il est obligatoire d'avoir terminé de faire tourner seul la totalité des exercices indiqués ci-dessus.

1. Faites tourner les deux exercices de l'examen de 1ère session de l'an dernier en procédant obligatoirement comme suit. Rédigez sur papier comme en examen (ou sur machine, mais sans compiler) l'exercice, en vous chronométrant. Relisez soigneusement votre code. Puis compilez-le et exécutez-le, il doit fonctionenr correctement *du premier coup*. Sinon, chronométrez le temps que vous mettez à le débuguer.
2. *Entraînement personnel.* Même question pour la deuxième session de l'an dernier, puis les autres annales disponibles.