Module ITSR43 - Programmation

TPs PL-SQL

Benoît Darties - benoit.darties@u-bourgogne.fr
Univ. Bourgogne Franche-Comté

Année universitaire 2015-2016

Avant-propos : Support de Travaux Pratiques (TP) dispensés à l'Université Bourgogne Franche-Comté, enseignements en Ingéniérie des Systèmes d'Information partie PL/SQL. La totalité de ce document a été rédigée uniquement à partir des connaissances de son auteur, et en utilisant un matériel personnel. L'utilisation / réutilisation partielle ou complète d'éléments de ce document est soumise à l'approbation de son auteur. Les corrections détaillées de chaque exercice sont réservées au personnel académique et disponibles par mail sur simple demande.

Consignes :

Les exercices présentés dans ce document sont à réaliser durant les séances TP PL-SQL du module ITC43 - PL/SQL. Ce document regroupe l'ensemble des TPs de cette partie. Les exercices sont à réaliser en binôme ou monôme exclusivement. Le travail à réaliser se doit d'être votre propriété exclusivement. Les enseignants de TP sont là pour vous aiguiller durant les séances prévues à cet effet. Les exercices pourront être complétés en dehors de ces séances en cas de manque de temps.

Il vous sera demandé de remettre un compte rendu de TP reprenant les réponses aux différents exercices proposés et dans lequel vous détaillez la démarche que vous avez adoptée. Ce compte rendu se présente plus sur la forme d'un rapport permettant de suivre le cheminement de votre démarche en la repositionnant dans le contexte de l'exercice, plutôt qu'une simple liste de réponses aux exercices. Il devra contenir une introduction, une table des matières, une liste des figures, une conclusion, et le cas échéant une bibliographie. Chaque figure doit être numérotée et référencée dans le texte à l'endroit adéquat. Le document pourra être rédigé en Word, LaTeX, ou tout autre éditeur de votre choix. Mais il devra être soumis en un seul fichier `pdf` dont la taille, images comprises, ne devra pas excéder *30 Mo*

Au terme de la dernière séance de TP PL/SQL, vous disposerez d'une semaine pour finaliser votre compte-rendu, puis le déposerez sur le serveur de dépôt à l'adresse suivante :

`http://depot.infotro.fr/ITSR43/`

Des éléments complémentaires pourront, le cas échéant, vous être communiqués par mail.

En cas de fortes similitudes avec un travail réalisé par un binôme de votre promotion ou d'une promotion précédente, une note de 0 non négociable sanctionnera votre travail, ainsi que celui du binôme ayant communiqué son travail à un tiers si ce dernier fait partie de la même promotion. Il vous appartient donc de protéger l'intégrité et la confidentialité de votre travail.

Avant-Propos

Pour une meilleure visualisation des tables sous `sqlplus`, vous pouvez exécuter les commandes suivantes (à exécuter à nouveau si vous quittez puis relancez `sqlplus`.

```
set colsep '|'
set linesize 167
set pagesize 30
set pagesize 1000
```

1 TP : organisation du Gala de l'ESIROUM

Contexte

Chaque année, l'ESIROUM organise son grand Gala. Cet événement célèbre les diplômés de l'année courante et permet aux anciens de se retrouver autours d'une table, de partager un bon repas, et d'échanger autours d'un verre ou deux au cours de la soirée. La commission soirée est en charge des commandes des boissons pour les participants.

1.1 Création d'une table sur la base de donnée `butor`

Les boissons disponibles ainsi que le prix HT de chaque boisson sont présentées dans la table `Boisson`. Les tuples de cette relation sont décrits ci-après :

`Boisson :`

- **ID** : identifiant de la boisson. Nombre. Clé primaire de la relation
- **nom** : nom de la boisson. chaine de 80 caractères au plus.
- **format** : verre, bouteille 75cl, magnum, géroboan, ... chaine de 40 caractères au plus.
- **alcoolise** : définit si la boisson est alcoolisé ou non. Booléen
- **degre_alcool** : degré d'alcool de la boisson. Nombre décimal
- **caractéristique** : définit si la boisson est pétillante, aromatisée, autre,...
- **prixHT** : prix de la boisson en euro. Nombre décimal

Un exemple de cette table est présenté ci-après :

ID	nom	format	alcoolise	degre_alcool	caracteristque	prixHT
1	bière blonde	25 cl	oui	6.5		2
2	Coca coca	33 cl	non	0	boisson gazeuse	2
3	Bière génépi	verre 25 cl	non	7		3
4	Vin rouge	bouteille 75 cl	oui	14 .2		16
5	Champagne	bouteille 75 cl	oui	13	boisson pétillante	23
6	Fanta orange	verre 25 cl	non	0	boisson gazeuse	2

1. Créer sur la base de données `butor`, la structure de cette table sans la remplir à l'aide de la requête SQL adéquate :
   - Les types de chaque attribut sont à définir à partir des domaines de chacun, annoncés précédemment
   - Oracle ne dispose pas de l'instruction mysql `auto increment`. Il n'est pour l'instant pas possible d'affecter automatiquement la valeur de la clé. Cette limitation sera vue par la suite.
2. Pour remplir cette table, définissez une procédure `ajouterBoissson` qui prend en paramètre 5 valeurs représentant les cinq attribiuts (ID, nom, format, degre_alcool, caracteristique), et ajoute dans la table une boisson avec ces attributs.

— Le booleen `alcoolise` sera déterminé à partir du degré d'alcool. Si ce dernier est égal à 0, la boisson n'est pas alcoolisé.

— Le prix sera défini a 0 lors de l'ajout

3. Appelez la procédure `ajouterBoissson` plusieurs fois de sorte à reconstituer la table présentée précédemment. Ajoutez ensuite quelques entrées de votre choix.

1.2 Affinage des données saisies par utilisation de triggers

La procédure mise en place pour l'ajout de boissons souffre de quelques lacunes :Il semble en effet possible d'ajouter des boissons avec un degré négatif, ce qui ne devrait pas être possible. Pour corriger ce problème, nous allons mettre en place un trigger.

4. Essayer d'ajouter une boisson avec un degré d'alcool négatif et vérifiez que l'ajout est bien possible.
5. Supprimez ensuite la ou les boissons avec degré d'alcool négatif
6. Créez enfin un trigger `triggerVerifDegre` qui, à chaque ajout d'une ligne sur la table `Boisson`, vérifie le degré d'alcool :

— Si le degré d'alcool est positif ou nul, la boisson est correctement ajoutée à la table

— Sinon, une exception est levée, un message d'erreur est affiché sur la console et le trigger se termine

Par ailleurs, afin d'éviter à devoir saisir un identifiant unique à chaque fois que l'on ajoute une boisson, nous allons émuler un fonctionnement similaire à l'instruction `auto increment` que vous avez déjà vu en `mySQL`. Cette fonction est implémentée depuis Oracle 12c sous la syntaxe :

```
ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY
```

Nous allons utiliser un méthode couramment utilisée dans les versions antérieures d'Oracle.

7. Créez une sequence nommée `seqIdBoissons` . Cette séquence sera un réservoir à ID dans lequel nous allons piocher à chaque ajout de boisson. La syntaxe de création d'une séquence est la suivante :

```
CREATE SEQUENCE nom_sequence
  START WITH 1
  INCREMENT BY 1;
```

Attention à positionner le début de la séquence de sorte à ne pas créer de conflits avec les données déjà existantes de la table.

8. Ajoutez un trigger nommé `majID`, qui affecte la valeur `seqIdBoissons.nextval` à l'attribut `ID` de chaque nouvelle ligne ajoutée à la table. `nextval` est ici un mot clé renvoyant la prochaine valeur de la séquence.
9. Vérifiez que les nouveaux identifiants sont correctement incrémentés à chaque fois que vous ajoutez une nouvelle boisson, non plus en fonction du paramètre passé dans la procédure `ajouterBoissson` , mais par rapport à la valeur déterminée par le trigger.
10. Mettez alors à jour la procédure `ajouterBoissson` de sorte à ne plus avoir à saisir l'identifiant de la boisson, et effectuez quelques ajouts de boisson pour tester la mise à jour.

1.3 Sauvegarde de la table dans un fichier

Le *dump* d'une d'une base de données consiste à créer un ensemble d'instructions en SQL à partir d'une base existante, de sorte à pouvoir dupliquer ou recréer la base originale en cas de problème. Cette notion de dump est très répandue dans le domaine, aussi bien pour ce qui concerne la sauvegarde et la restauration des données, mais également leur vol par des pirates informatiques.

Le dump d'une base se présente sous la forme d'un fichier contenant un ensemble d'instructions SQL. Ces instructions peuvent se limiter à la structure d'une ou plusieurs tables, à leur donnée seulement, aux procédures et triggers. *Créer un dump de la base* consiste à créer ce fichier à partir

d'une base existante, tandis qu'*injecter un dump* dans une base de données consiste à restaurer la base à partir des données du fichier.

Nous allons dans un premier temps réaliser un fichier *dump* de la table `boisson`,. De nombreux outils d'administration permettent de réaliser des fichiers dumps d'une base existante.

11. A titre purement informatif, recherchez sur internet le nom de l'outil permettant de faire des dump de tables sous mySQL en ligne de commande. Cet outil n'est pas disponible sous oracle avec sqlplus.
12. Sous oracle, l'utilitaire permettant de créer un dump de la base est la commande `exp`. Vous pouvez utiliser cette commande soit en passant des paramètres, soit sans paramètre, auquel cas un prompteur vous demandera les différentes informations nécessaires. Lancez cet utilitaire sans paramètres, en prenant soin d'être dans votre répertoire personnel pour garantir l'accès en écriture. Les questions suivantes vous sont alors posées :
    (a) Username : **tapez votre login**
    (b) Password : **tapez votre mot de passe (identique login)**
    (c) Enter array fetch buffer size : 4096 > **tapez 4096**
    (d) Export file : expdat.dmp > **tapez un nom de fichier pour l'export**
    (e) (2)U(sers), or (3)T(ables) : (2)U > **tapez 3 (pour exporter les tables)**
    (f) Export table data (yes/no) : yes > **tapez yes**
    (g) Compress extents (yes/no) : yes > **tapez yes**
    (h) Table(T) or Partition(T :P) to be exported : (RETURN to quit) > **tapez le nom de la table à sauvegarder, ici** `boisson`
    (i) Table(T) or Partition(T :P) to be exported : (RETURN to quit) > **validez simplement pour teminer le dump**

    En cas de succès, le programme aura créé le fichier dont vous aurez précisé le nom, et y aura mis la structure de la table `boisson`, ses données (compressées) et les triggers définis sur cette table. Attention, les procédures ne sont pas stockées dans ce fichier dump.
13. Visualisez le contenu du fichier, par exemple en utilisant la commande `cat`. Vérifiez la présence des instructions permettant de créer la table et le trigger. Les données ne sont pas clairement identifiables car compressées.

1.4 Restauration de la table depuis un fichier

Une fois la table sauvegardée, nous allons tester le fichier de sauvegarde, et sans filet.

14. Dans `sqlplus`, supprimez la table `boisson` avec l'instruction SQL `drop table boisson;`
15. Quittez ensuite `sqlplus` et appelez alors la commande `imp`. cette commande permet d'injecter un fichier dump dans la base. Les questions suivantes vous sont alors posées :
    (a) Username : **tapez votre login**
    (b) Password : **tapez votre mot de passe (identique login)**
    (c) Import data only (yes/no) : no > `no`
    (d) Import file : expdat.dmp > **tapez le nom de fichier à importer**
    (e) Enter insert buffer size (minimum is 8192) 30720> **tapez 30720**
    (f) List contents of import file only (yes/no) : no > **tapez no**
    (g) Ignore create error due to object existence (yes/no) : no > **tapez no**
    (h) Import grants (yes/no) : yes > **tapez yes**
    (i) Import table data (yes/no) : yes > **tapez yes**
    (j) Import entire export file (yes/no) : no > **tapez no**
    (k) Username : **tapez votre login**
    (l) Enter table(T) or partition(T :P) name or . if done : **tapez le nom de la table à injecter, ici** `boisson`

(m) Enter table(T) or partition(T :P) name or . if done : **tapez . pour terminer l'import**

Le programme se termine en mentionnant normalement que la table a été restaurée. Vérifiez sous `sqlplus` son contenu et si le trigger a bien été recréé avec la commande :

```
select * from all_triggers where TRIGGER_NAME IN ('triggerVerifDegre', 'majID') ;
```

1.5 Manipulation d'une table depuis un système distant

La table `boisson` a été définie sur le SGBD `butor`. Nous allons ensuite l'interroger depuis le SQBG `eluard`.

16. Tout en gardant une connexion ouverte sur le premier SGBD `butor`, ouvrez une seconde connexion sur le SGBD `eluard` en procédant de manière identique que pour `butor`, et vérifiez que la table `boisson` est inexistante.
17. Le lien de `eluard` vers `butor` a été défini par le nom `dl_e2b`. Depuis `eluard`, listez le contenu de la table `boisson` de `butor` avec l'instruction :

    ```
    select * from boisson@dl_e2b;
    ```

18. Depuis `eluard`, insérer deux nouvelles boissons dans la table `boisson`, et supprimez une autre boisson.
19. Listez à nouveau le contenu de la table `boisson` depuis `eluard`, puis depuis `butor`. Que constatez-vous ?
20. Depuis `butor`, supprimez une autre boisson et listez à nouveau le contenu de la table `boisson` depuis `eluard`, puis depuis `butor`. Que constatez-vous ?
21. Lorsqu'une modification est faite, cette dernière est en fait locale, et ne sera pas propagée tant que l'instruction `commit;` n'est pas exécutée. Cette instruction permet de valider une transaction (ensemble de modifications) pour qu'elles puissent être propagées depuis l'extérieur. Exécutez cette commande sur `butor`, et listez à nouveau le contenu de la table `boisson` depuis `eluard`, puis depuis `butor`. Que constatez-vous ?
22. Exécutez ensuite cette commande sur `eluard`, et listez à nouveau le contenu de la table `boisson` depuis `eluard`, puis depuis `butor`. Que constatez-vous ? A ce stade, les deux résultats doivent à nouveau être identiques.
23. Lorsqu'une modification locale a été effectuée, mais que l'on ne souhaite plus la propager, il est possible de revenir en arrière, avec la commande `rollback;`. Testez cette commande en supprimant une boisson, en affichant la table, en exécutant `rollback` et en affichant à nouveau la table.

1.6 Intéractions entre tables de systèmes distants

Nous ajoutons deux nouvelles tables à notre modèle. Ces deux tables seront disposées sur `eluard`, tandis que la table`boisson` reste sur `butor`.

La table `participant` recense la liste des personnes qui ont prévu de participer au gala et le budget qu'il va investir en boissons. Elle se présente de la façon suivante :

`Participant :`

- **ID** : identifiant du participant
- **nom** : nom du participant
- **prenom** : prénom du participant
- **budget** : budget prévu pour les boissons

Un exemple de cette table est présenté ci-après.

ID	nom	prenom	budget
1	Nemchi	Cyriaque	50
2	Allayaud	Thomas	54
3	Dupin	Nicolas	40
4	Morandini	Marc	30
5	Grastilleur	Julie	60

Enfin, nous ajoutons la table `commande`. Cette table résume l'ensemble des commandes d'alcool prévues par les participants. Cette table se présente de la façon suivante :

`Commande :`

— **IDParticipant** : Identifiant du participant

— **IDBoisson** : Identifiant de la boisson

— **Quantite** : quantité de boisson commandée

— **Etat** : état de la commande : en attente, validé ou refusé, en fonction du budget du participant

Un exemple de cette table est présenté ci-après.

IDParticipant	IDBoisson	quantité	Etat
1	2	3	'en attente'
1	1	3	'en attente'
4	2	5	'en attente'
4	5	6	'en attente'
3	2	4	'en attente'
2	1	1	'en attente'
1	3	2	'en attente'
2	2	1	'en attente'

24. De manière analogue à ce qui a été entrepris dans la table `boisson`, créez sur `eluard` la structure de cette table, une séquence nommée `seqIDParticipant`, un trigger `majIDParticipant`, une procédure `ajouterParticipant` qui ajoute un participant à partir de son nom, son prénom, et un budget donnés en paramètre, et ajoutez les participants mentionnés dans l'exemple ainsi que d'autres participants.
25. De manière analogue à ce qui a été entrepris dans la table `boisson`, créez sur `eluard` la structure de cette table
26. Ajoutez une procédure `ajouterCommande` qui ajoute une commande dans la table `commande` à partir de 3 paramètres que sont : l'identifiant d'un participant, l'identifiant d'une boisson, et une quantité. Par défaut, l'état de la commande doit être mis à 'en attente', et passez quelques commandes.
27. Ajoutez alors une procédure `verifierCommande` :
    — pour chaque ligne de commande ayant l'état 'En attente', on détermine le coût de la commande (égal au prix unitaire HT multiplié par la quantité et par 1.20 pour avoir le prix TTC.
    — On compare alors le coût de la commande avec le budget du participant.
    — Si le participant dispose d'un budget suffisant, on décrémente ce dernier du coût de la commande, et on met la commande en état 'validé'
    — Sinon, on refuse la commande en passant son état à 'refusé'

    Exécutez alors cette procédure.
28. Ajoutez enfin un trigger `triggerVerifierCommande`, qui effectue cette vérification en live dès qu'une commande est ajoutée et met à jour automatiquement l'état de la commande.

1.7 Intéractions avancées [question bonus]

Un événement inattendu est apparu : le fournisseur de Bière génépi vient déposer le bilan et ne sera pas en mesure de livrer la bière.

29. Afin de gérer cette situation et de prévenir toute situation similaire, mettez en place une solution à base de triggers et de procédures, agissant de la façon suivante :

    — Si une boisson venait à être supprimée, il faudra déjà supprimer les commandes refusées ou en attente relatives à cette boisson.

    — Pour les commandes qui avaient été validées, il faudra les supprimer en recréditant à chaque participant l'argent qui avait été dépensé pour cette commande.

    — Les participants ayant été recrédités disposent désormais de peut-être suffisamment d'argent pour que l'on puisse valider des commandes qui avaient été refusées. Il faudra donc revérifier pour chaque commande qui a été refusée à ce participant s'il est désormais possible de la valider ou pas. Si le participant dispose de suffisamment d'argent, la commande passe de refusé à validé et le budget du participant est débité du coût de la commande.

    Il convient de mettre en place des procédures et triggers ciblés, c'est à dire qui agissent sur un minimum de données (exemple seulement les participants concernés par une suppression de boisson, et pas tous les participants).