TP1 : SQL Avancé sous Oracle et Langage PL/SQL

L'objectif de ce premier TP est de vous montrer les capacités du SGBD Oracle en terme de programmation (langage PL/SQL) et de modélisation orientée-objet.

Initialisation de la base de données

Une bibliothèque gère un ensemble d'exemplaires de livres qui peuvent être empruntés, à une date donnée, par une personne. Ces livres sont décrits par un titre et un ou plusieurs auteurs. Les auteurs sont des personnes qui peuvent avoir eu le prix goncourt. Les personnes sont décrites par un nom, un prénom et leur père.

Le schéma entité-association de ces données est proposé sur la figure **??**.

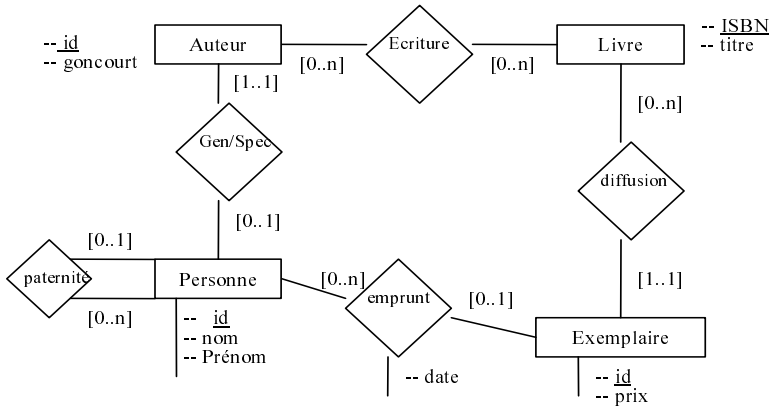

FIGURE 1 – schéma entité-association des données

Les scripts permettant de créer les tables du modèle logique dérivé de ce schéma entité-association est disponible à l'adresse `http://www.lisi.ensma.fr/ftp/enseignement/Oracle/CAUC`.

Création et peuplement des tables de la base de données

Questions

A-1. Exécuter le script `create_table.sql` pour créer ce modèle logique.

A-2. Exécuter le script `insert.sql` pour insérer des données dans votre modèle logique. Vérifiez que votre base de données comprend 6 livres, 11 personnes, 5 auteurs et 15 exemplaires.

SQL avancé sous Oracle

Pour chacune des questions suivantes, créez une requête SQL permettant d'obtenir les résultats à cette question. Pour chaque requête une indication du nombre de résultats à trouver est donnée.

A-3. Quel est le nombre d'exemplaires disponible pour chaque titre de livre ? Vous retournerez la valeur `0` lorsqu'un livre n'a pas d'exemplaires. (6 résultats).

A-4. Depuis combien de jours les différents exemplaires de la bases de données ont-ils été empruntés ? Vous retournerez la valeur `0` lorsque l'exemplaire n'est pas emprunté. Vous pourrez pour cela utiliser l'opérateur `NVL` disponible sous Oracle. (15 résultats).

A-5. Quels sont les noms et prénoms des auteurs ? (5 résultats).

A-6. Quel sont les noms et prénoms des personnes qui ne sont pas des auteurs ? Vous utiliserez l'opérateur de différence de l'algèbre relationnel (`MINUS` sur Oracle) pour définir cette requête. (6 résultats).

A-7. Quels est le père et le grand père de Jens Hawking ? Vous devez utiliser pour cela une requête hiérarchique décrite dans l'aide ci-dessous. Pensez également à soigner l'affichage en utilisant l'opérateur `CASE`. (2 résultats).

Aide

Syntaxe des jointures externes : `FROM table1 LEFT OUTER JOIN table2 ON condition`.

`NVL` est une fonction de signature : `NVL(expression, valeur)`. Si `expression` est évaluée à la valeur `NULL`, le retour de cet opérateur est `valeur`.

L'opérateur `SYSDATE` permet d'obtenir la date du jour. La soustraction de deux dates retourne le nombre de jours entre les deux dates. Le résultat est un réel, l'opérateur `ROUND` permet de l'arrondir à l'entier le plus proche.

L'opérateur `CAST` à la syntaxe suivante :

```
CAST(<valeur> AS TYPE)
```

Il permet de convertir `valeur` dans le type de données `type`.

La commande définie par la norme SQL pour faire un test à la syntaxe suivante :

```
CASE WHEN <condition> THEN <valeur> ELSE <valeur> END
```

Une requête hiérarchique permet de parcourir récursivement les tuples d'une table lorsqu'une colonne de celle-ci référence un autre tuple de cette table.

La syntaxe générale d'une requête hiérarchique est la suivante :

```
SELECT LEVEL, ...
FROM ...
WHERE ..
START WITH <condition>
CONNECT BY PRIOR <condition>
```

La condition qui suit `START WITH` permet de définir la racine du parcours. `LEVEL` retourne le niveau de parcours par rapport à une racine donnée. La condition après `CONNECT BY PRIOR` définie le lien entre deux tuples de la table. Le membre gauche (resp. droit) de cette condition est la colonne du tuple de niveau inférieur (resp. supérieur) dans le parcours.

Langage PL/SQL

PL/SQL est un langage procédural proche de l'ADA. L'exécution du code écrit dans ce langage s'effectue sur le serveur de base de données. L'étude du PL/SQL nécessiterait un cours à part entière. Aussi, cette partie n'a pour but que de vous faire découvrir quelques fonctionnalités de ce langage par sa mise en pratique sur les trois unités de programmes possibles : blocs anonymes, procédures stockées et déclencheurs.

B-Blocs anonymes

Manipulation de chaînes de caractères

Un bloc PL/SQL a la structure suivante :

```
DECLARE
/* Facultative : déclaration de variables, types et constantes */
BEGIN
/* Obligatoire : instructions du programme terminées par ; */
EXCEPTION
/* Facultative : gestion des exceptions */
END;
```

Les fonctions et opérateurs de manipulation de chaînes de caractères les plus utiles sont :

- `dbms_output.put_line(str)` : affichage de la chaîne `str` ;
- `length(str)` : longueur de la chaîne ;
- `upper(str)` : mise en majuscule d'une chaîne ;
- `substr(str,m,n)` : sous chaîne de `str` commençant au caractère `m` de longueur `n` (ou jusqu'à la fin de `str` si ce paramètre n'est pas renseigné) ;
- `str1 || str2` : concaténation de chaînes ;

Questions

B-1. Ecrivez un bloc PL/SQL qui affiche « `Hello World !` ». Pour voir l'affichage, vous devrez cliquer sur le bouton `ENABLE` du panel `DBMS OUTPUT` de SQLDeveloper.

B-2. Complétez le bloc suivant pour afficher `'Les identifiants sont constitués des trois premières lettres du nom et des trois premières du prénom ; ainsi, JENNIFER RORAMLE a pour identifiant RORJEN'`. Vous devez, pour cela, utiliser les deux variables `nom` et `prenom`.

```
1 DECLARE
2  nom VARCHAR(30) := 'Roramle';
3  prenom VARCHAR(30) := 'Jennifer';
4 BEGIN
5 ...
6 END;
```

Variables

Les types utilisés en PL/SQL pour les variables sont généralement `NUMBER`, `VARCHAR(N)` et `BOOLEAN`. Les variables sont souvent utilisées pour manipuler les données contenues dans une base de données. Aussi, PL/SQL permet de référencer le type d'une colonne d'une table par l'opérateur `%TYPE` et la syntaxe `<nom_table>.<nom_attribut>%TYPE`.

PL/SQL permet également de déclarer des types articles (`RECORD` en ADA, `STRUCT` en C). Ils permettent notamment de déclarer des variables qui peuvent contenir un tuple d'une table. Par exemple, une variable du type `t_personne` défini ci-dessous permet de contenir une personne de notre base de données :

```
TYPE t_personne IS RECORD (
   id NUMBER(8,0),
   nom VARCHAR(32),
   prenom VARCHAR(32),
   pere  NUMBER(8,0));
```

Pour simplifier ceci,l'opérateur `%ROWTYPE` par la syntaxe `<nom_table>%ROWTYPE` permet de définir un type article contenant un champs par colonne de la table. Ainsi, une variable devant contenir un tuple de la table `Personne` peut se déclarer par :

```
ma_personne t_personne; /* Equivalent à : */
ma_personne Personne%ROWTYPE;
```

La variable `ma_personne` est de type `Personne`.

Questions

B-3. Quel(s) avantage(s) voyez-vous à utiliser les opérateurs `%TYPE` et `%ROWTYPE` plutôt que directement les types de la table ?

B-4. Dans un bloc PL/SQL, déclarez une variable du type de la colonne `nom` de la table `Personne`. Quelle valeur par défaut est attribuée à cette variable ? Quel(les) particularité(s) à cette valeur ?

B-5. Dans un bloc PL/SQL, déclarez une variable de type `Livre`. Initialisez cette variable par l'`ISBN 100` et le titre `Los Angeles`. Enfin, affichez la valeur de cette variable à l'écran.

Manipulation des données de la base

PL/SQL permet de faire toutes les opérations de manipulation de données (`INSERT`,`UPDATE`,`DELETE` et `SELECT`) mais pas de modification de schéma (`CREATE`,`ALTER` et `DROP`). Les opérations `INSERT`,`UPDATE` et `DELETE` s'utilisent directement comme des instructions. Pour les opérations `SELECT`, il faut utiliser des variables pour récupérer le résultat de la requête. Ainsi, après les colonnes de la clause `SELECT` et avant la clause `FROM` de la requête, PL/SQL autorise la syntaxe :

```
INTO <var1>,<var2> ...
```

pour pouvoir récupérer les valeurs du premier tuple retourné.

Questions

B-6. Inserez un nouveau livre sans titre puis modifiez le pour lui attribuer le titre `A la vitesse de la lumière` dans un bloc PL/SQL.

B-7. Modifiez le bloc de la question B-5 pour que votre variable livre soit initialisée avec les valeurs du livre de numéro `ISBN` 1 de votre modèle logique.

Une requête pouvant ne ramener aucun résultat ou plusieurs, l'utilisation de variables pour récupérer le résultat de requêtes n'est pas très souple. PL/SQL introduit pour cela le concept de curseur. Avant de présenter cette notion, il faut aborder les structures de contrôle de flot de données offertes par ce langage.

Contrôle du flot de données

La syntaxe du `IF` est la suivante :

```
IF <condition_1> THEN <instructions>
ELSIF <condition_2> THEN <instructions>
ELSE <instructions>
END IF;
```

Trois types de boucles peuvent être utilisés :

```
LOOP
EXIT WHEN <condition>;
END LOOP
-----------------------
WHILE <condition> LOOP
END LOOP;
-----------------------
FOR <var> IN <indice_debut>..<indice_fin> LOOP
END LOOP;
```

B-8. Écrivez un bloc PL/SQL qui crée un livre. En utilisant la structure de contrôle `IF`, faites en sorte que le titre du livre soit stocké en majuscule si celui-ci débute par la lettre `'t'`.

B-9. Utiliser un des 3 types de boucles pour insérer 100 personnes dans votre modèle logique.

Curseurs

Un curseur est une variable qui permet le parcours des tuples d'une table ou du résultat d'une requête. La déclaration d'un curseur se fait selon la syntaxe suivante :

```
CURSOR <nom_curseur> IS
<requête>
FOR UPDATE;
```

L'instruction `FOR UPDATE` est optionnelle. Elle doit être spécifiée si le curseur doit permettre de modifier les données de la table.

Une fois déclaré, le curseur s'utilise par les opérations suivantes :

```
/* avant toute utilisation du curseur */
OPEN <nom_curseur>
/* placement du curseur sur le prochain tuple */
FETCH <nom_curseur> INTO <var1>,<var2>
/* le curseur est-il sur un tuple vide ? */
<nom_curseur>%NOTFOUND /* boolean */
/* Mise à jour d'une table en se basant sur la position d'un curseur
portant sur celle-ci */
UPDATE <nom_table> ... WHERE CURRENT OF <nom_curseur>
/* Libération de la mémoire occupée par le curseur */
CLOSE <nom_curseur>
```

Questions

B-10. Utilisez un curseur pour afficher l'ensemble des personnes de votre modèle logique.

B-11. Utilisez un curseur pour augmenter le prix des exemplaires du livre d'`ISBN 1` de `10`.

Exceptions

Les traitements d'exceptions se font par la syntaxe :

```
WHEN <nom_exception> THEN <traitement>
```

Parmi les très nombreuses exceptions prédéfinies, `dup_val_on_index` est levée dès qu'une instruction essaie de créer une valeur existante dans une colonne sur laquelle une contrainte d'unicité est définie.

Question

B-12. Écrivez un bloc PL/SQL qui essaie d'insérer un livre dont le numéro `ISBN` existe déjà. Notez le résultat de l'exécution de ce bloc.

B-13. En définissant un traitement pour l'exception `dup_val_on_index`, modifiez ce comportement pour que le livre soit inséré avec la prochaine valeur de la séquence `SEQ_Livre` (pour l'obtenir, appeler `SEQ_Livre.NEXTVAL`).

C-Procédures et fonctions stockées

Définition de procédures stockées

Une procédure tout comme une fonction (une procédure qui retourne une valeur) a un nom, est stockée dans la métabase et peut être appelée par plusieurs utilisateurs.

La définition d'une procédure se fait par la syntaxe :

```
CREATE OR REPLACE PROCEDURE <nom_procedure> (<paramètres>) AS
BEGIN
EXCEPTION
END <nom_procedure>;
```

Les paramètres se déclarent comme des variables. La seule différence c'est qu'un mode peut être défini entre le nom de la variable et son type. Trois modes sont disponibles :

- `IN` : lecture seule ;
- `OUT` : écriture seulement ;
- `IN OUT` : lecture/écriture ;

L'éxécution du fichier dans lequel se trouve une procédure stockée provoque sa compilation. Lorsque celle-ci présente des erreurs vous obtiendrez le message « Avertissement : Procédure créée avec erreurs de compilation ». La commande `SHOW ERRORS` sous SQL*PLUS permet de voir ces erreurs. Lorsque la compilation sera correct, le message suivant apparaît « Procédure créée. »Vous pourrez alors la tester en écrivant un bloc PL/SQL qui y fait appel.

Questions

C-1. Inspirez-vous du bloc de la question B-11 pour écrire une procédure stockée qui augmente de `10` le prix des exemplaires des livres dont le numéro `ISBN` est passé en paramètre.

C-2. Écrivez un procédure qui permet de récupérer l'emprunteur de l'exemplaire le plus cher pour un numéro `ISBN` de livre donné.

Gestion des procédures stockées

Les procédures et fonctions contenues dans la base de données peuvent être retrouvées dans la table `user_objects` de la métabase. La colonne `object_type` pour une procédure prend la valeur `PROCEDURE`.

Questions

C-3. Quelle est la différence dans la métabase entre une procédure qui est compilée sans erreur et une avec ?

D-Déclencheurs (Triggers)

Un déclencheur est un cas particulier de bloc PL/SQL. Il définit un traitement à effectuer avant `BEFORE` ou après `AFTER` un évènement telle qu'une instruction `INSERT`,`UPDATE` ou `DELETE` sur une table. Ce traitement peut être exécuté une seule fois ou pour chaque ligne modifiée (`FOR EACH ROW`).

La syntaxe de définition d'un déclencheur est la suivante :

```
CREATE OR REPLACE TRIGGER <nom_trigger>
  <avant_ou_apres> <evt> ON <nom_table>
  REFERENCING NEW AS <nom_nouveau_tuple> OLD AS <nom_ancien_tuple>
  FOR EACH ROW WHEN (<condition>)
BEGIN
END;
```

Les variables `NEW` et `OLD` sur lesquelles un alias peut être défini dans la clause `REFERENCING` permettent de consulter ou de modifier le tuple sur lequel s'est déclenché le trigger. Dans la partie entre `BEGIN` et `END` ces variables doivent être préfixées par le caractère deux points.

Questions

D-1. Créez un déclencheur qui affiche un message dès que la table personne est modifiée. Vous le définirez au niveau de la table puis au niveau d'un tuple et illustrerez les différences que ceci implique.

D-2. Créez un déclencheur pour vérifier qu'à l'insertion et la mise à jour d'un exemplaire l'emprunteur d'un livre n'est pas un auteur. Si c'est le cas, vous exécuterez l'instruction `Raise_application_error(-20201, 'L''emprunteur d'un livre ne peut pas être un auteur');`

Aide

La table de la métabase contenant des informations sur les déclencheurs est `user_triggers`. Pour supprimer un déclencheur la commande est :

```
DROP TRIGGER <nom_declencheur>
```

Pour le désactiver (`DISABLE`) ou l'activer (`ENABLE`) :

```
ALTER TRIGGER <nom_declencheur> <activation_ou_desactivation>
```