TP2 : PL/SQL (séance 1)

1. Schéma de base de données

Nous allons travailler sur une version de la base de données Commune (créée et peuplée la semaine dernière). Le schéma relationnel vous est rappelé ci-dessous :

- Region(**reg varchar(4)**, chef_lieu varchar(46), nom_reg varchar(30))
- Departement(reg varchar(4), **dep varchar(4)**, chef_lieu varchar(46), nom_dep varchar(30))
  avec Departement(reg)⊆Region(reg)
- Commune(dep varchar(4),com varchar(4),article varchar(4),com_nom varchar(46),longitude float,latitude float,pop_1975 float,pop_1976 float, pop_1977 float,pop_1978 float,pop_1979 float,pop_1980 float,pop_1981 float,pop_1982 float,pop_1983 float,pop_1984 float,pop_1985 float,pop_1986 float,pop_1987 float,pop_1988 float,pop_1989 float,pop_1990 float,pop_1991 float, pop_1992 float,pop_1993 float,pop_1994 float,pop_1995 float,pop_1996 float,pop_1997 float,pop_1998 float,pop_1999 float,pop_2000 float,pop_2001 float,pop_2002 float,pop_2003 float,pop_2004 float,pop_2005 float,pop_2006 float,pop_2007 float,pop_2008 float,pop_2009 float,pop_2010 float, **code_insee varchar(4)**)
  avec Commune(reg)⊆Region(reg)

2. Premiers triggers

1. Construisez un trigger qui vérifie que la population de 2010 (pop_2010 de Commune) est toujours positive (ordres insert et update).
2. Les triggers ont souvent une écriture très simple (règle ECA) et s'appuient sur des procédures qui en masquent la complexité. Vous reprendrez l'écriture du trigger ouvrable. Vous définirez une procédure JoursEtHeuresOuvrables sans argument qui vérifie que la date du jour n'est pas un samedi ni un dimanche et qui renvoie un message d'erreur autrement. Vous redéfinirez le trigger ouvrable qui fera appel à cette procédure, dans le contexte de la table Region. Vous en testerez les effets.
3. Les triggers ont souvent un rôle de monitoring auprès des administrateurs de bases de données (indicateurs de la bonne santé des bases de données dont ils ont la charge). Vous créerez une table historique (dateOperation, nomUsager, typeOperation) qui va permettre de conserver la trace de toutes les opérations réalisées sur la table *Region*. Vous créerez le trigger qui va permettre d'alimenter cette table.
   Le nom de l'usage et la date système sont connus au travers des descripteurs USER et SYSDATE (cf. select user, sysdate from dual ;)
4. Lors de la définition de contraintes de clés étrangères, Oracle autorise certaines fonctionnalités telle que la suppression en cascade des tuples dépendants au travers de la syntaxe réservée *on delete cascade*. A la différence de PostgreSQL, Oracle ne permet pas l'exploitation de la syntaxe

on update cascade qui permettrait de modifier les tuples dépendants en conséquence.
Vous construirez un trigger nommé *cascade* qui porte sur la table Région et qui se charge à chaque évènement de suppression ou de modification d'une région (reg) dans *Region* de supprimer ou de modifier dans la table *Departement*, les tuples de département dépendants de cette région. Pensez ensuite à annuler les effets des suppressions ou modifications par rollback sur la transaction.

2.1 Curseurs

1. Vous écrirez un curseur implicite qui permet d'afficher pour le département de l'Hérault (34), le nombre d'habitants en 2010 (somme de tous les habitants de toutes les communes de l'Hérault).
2. Vous écrirez un curseur explicite qui permet de retourner le nombre d'habitants en 2010 département par département

2.2 Evolution de schéma

Nous avons vu lors du TP précédent quelques difficultés liées à de la réconciliation de schémas. Les données portant sur les communes proviennent d'une première source et contiennent seulement les informations des communes de métropole (plus Corse). Les données portant sur les départements et régions proviennent d'une seconde source avec quelques différences sur les identifications des départements en particulier pour la Corse et comprennent les informations sur les ROM-COM. Le schéma de données actuel pose lui aussi quelques problèmes en particulier sur la manière de modéliser les flux de population au regard des années. Les attributs allant de population_1975 à population_2010 sont en réalité des indicateurs qui agrègent une notion de population pour une année donnée. Il faut revenir à des attributs élémentaires pour faire en sorte de ne pas avoir à ajouter de nouveaux attributs à la relation Commune à chaque année qui s'écoule d'une part, mais aussi pour faciliter une exploitation efficace de l'information, d'autre part. Un exemple de curseur difficile à programmer en tant que tel vous est donné.

1. Vous écrirez un curseur explicite qui permet de retourner le nombre d'habitants année par année et département par département

Vous allez faire évoluer le schéma, en respectant le diagramme de classes UML (figure 1) proposé ci-dessous.

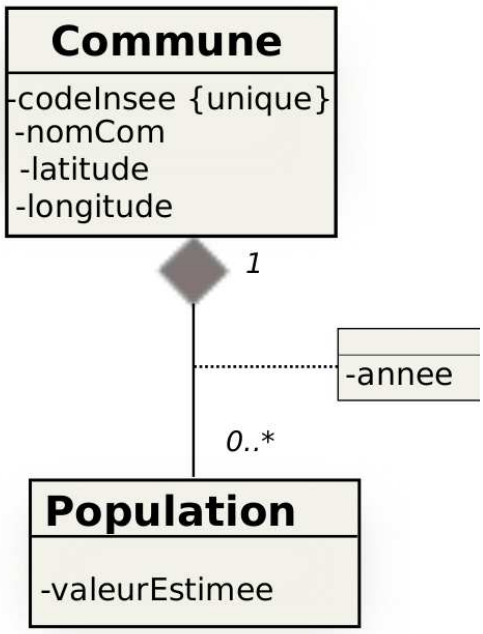

FIGURE 1 – Ajout de modélisation

Créez la table Population : Population(**codeInsee varchar(6), annee integer**, valeurEstimee float)
Vous alimenterez automatiquement cette nouvelle table provenant de la décomposition de Commune en vous aidant d'un programme principal ou d'une procédure PL/SQL.

2.3 Fonctions ou Procédures

1. Vous écrirez une fonction (ou une procédure) qui prend en entrée un numéro de département ainsi qu'une année (comprise entre 1975 et 2010) et qui retournera le nombre d'habitants du département sur l'année considérée (pensez à gérer les exceptions possibles).
2. Vous écrirez une fonction (ou une procédure) qui renvoie le nombre (et le pourcentage) d'utilisateurs connectés à la base. A cet effet, vous exploiterez les requêtes suivantes qui expriment respectivement : donner le nom des usagers "non système" de la base de données et donner le nom des usagers (qui en possèdent un) connectés.
   select username from dba_users where username <> '%SYS%'
   select username from v$session where username is not null

2.4 Objets du schéma

Vous exploiterez les tables du méta-schéma : USER_TRIGGERS, USER_PROCEDURES, USER_OBJECTS et USER_SOURCE pour avoir une vue d'ensemble sur les différents objets PL/SQL que vous venez d'ajouter à votre schéma. Vous supprimerez les déclencheurs construits.