

# TP3 : PL/SQL (séance 2)

## 1. Schéma de base de données

---

Nous allons poursuivre sur la base de données `Communes` de `France`

## 2. Evolution du schéma de base de données

---

Le code de la question 2.2 du TP précédent est donné. Il fait appel à une construction de curseur particulière (dynamique) qui rend possible le passage de paramètres à l'exécution. Vous aurez plusieurs tâches à réaliser :

- créer la table `Population` (si non déjà fait lors du TP précédent)
- alimenter la table `Population` à l'aide des fonctions définies au sein du script fourni
- créer une table `Commune_Old` à partir de `Commune` : même structure, mêmes tuples; afin de conserver une structure de table dotée de plus de 40 colonnes et de plusieurs milliers de lignes pour des tests de performance futurs (même si cette table `Commune_Old` n'est pas conforme à la théorie de la normalisation).

```
create table commune_old as select * from commune;
```

Listing 1 – ordre create

- Supprimer de `Commune`, les colonnes allant de `pop_1975` à `pop_2010` Vous pouvez choisir soit d'exécuter 36 fois la commande `alter`, soit de faire jouer du SQL dynamique au travers de la commande `execute immediate`

```
alter table commune_old drop column pop_2010;
```

Listing 2 – ordre alter

## 3. Constructions de fonctions

---

Vous définirez une fonction de calcul de distance kilométrique. Vous gérerez pour cette fonction différentes exceptions de manière à prévenir une grande majorité des erreurs pouvant impacter le schéma de la table `commune`. Le calcul de la distance en kilomètres entre deux points `a` et `b` ayant respectivement une latitude et une longitude notées `lat_a`, `long_a`, et `lat_b`, `long_b` (référentiel WGS84) se fait de la manière suivante :

```
6366*acos(cos(radians(lat_a))*cos(radians(lat_b))*cos(radians(long_b)-radians(long_a))  
+sin(radians(lat_a))*sin(radians(lat_b)));
```

Listing 3 – distance Km

Les valeurs des latitudes et longitudes de la table Commune sont en degrés. Il vous faudra convertir les degrés en radians en définissant la fonction radians qui effectue le calcul :

```
valeurEnDegre/57.295779513082;
```

Listing 4 – degre vers radian

La valeur 6366 correspond à une estimation en Km du rayon de la Terre. La fonction trigonométrique prend en considération la rotondité de la Terre. Vous pouvez construire différentes variantes de la fonction, avec une fonction qui prend pour arguments d'entrée les deux noms de communes considérées.

## 4. Constructions de paquetages

---

Vous construirez deux paquetages :

1. un paquetage nommé UrbanUnits qui visera à proposer des fonctions et procédures pour mieux caractériser les communes, les départements et les régions. La fonction de calcul kilométrique viendra se placer dans ce paquetage. Vous pourrez aussi y définir différentes fonctions de comptage des populations (reprenez notamment les fonctions demandées dans la question 2.3 du TP précédent). Vous proposerez des exemples d'utilisation du paquetage ainsi construit.
2. un paquetage nommé Supervision qui visera à proposer des fonctions et procédures pour "monitorer" la base de données. Notamment vous exploiterez les vues du méta-schéma pour :
  - (a) manipuler les vues structurelles user\_constraints, user\_tab\_columns et user\_cons\_columns pour définir une procédure qui renvoie pour une table donnée, toutes les informations sur ses attributs, le type de ces attributs, les contraintes qui s'appliquent éventuellement sur ces attributs et le type de ces contraintes. Un exemple de ce qui est attendu est donné ci-dessous :

```
COLUMN_NAME      DATA_TYPE      CONSTRAINT_NAME  C
-----
NUM              NUMBER         EMP_PK          P
N_SUP           NUMBER         EMP_FK1         R
EMBAUCHE        DATE
COMM            NUMBER
NOM             VARCHAR2
SALAIRE         NUMBER
N_DEPT          NUMBER
FONCTION        VARCHAR2

8 rows selected.
```

FIGURE 1 – Exploiter la fonction

- (b) **Utilisateurs connectés** Vous écrirez une fonction qui renvoie les utilisateurs connectés (ainsi que le nom du terminal sur lequel ils sont connectés et le type de client qu'ils utilisent)
- (c) **Suppression des éléments du schéma** L'idée est, ici, d'exploiter la notion de SQL dynamique au travers de la commande `execute immediate`. Il s'agit de permettre l'exécution de requêtes faisant le lien avec des objets qui ne sont pas encore connus au moment de la compilation et qui ne le seront qu'en tout dernier lieu à l'exécution. Ainsi il sera rendu possible d'interroger les vues du méta-schéma et de supprimer, par exemple, à la volée toutes les tables, vues ou autres objets du méta-schéma. Vous construirez deux procédures : une procédure qui supprimera une table dont le nom sera passé en paramètre, une procédure qui supprimera toutes les tables du schéma. Vous pouvez choisir de travailler sur un autre objet de la base (vue, trigger, ...) si vous préférez conserver vos tables.