❽ Périgée : Logiciel de gestion des notes de l'IUT de Saint Denis

On s'intéresse ici au logiciel Périgée qui permet de gérer les notes des étudiants du département Informatique de l'IUT de Saint-Denis de la Réunion. Ce logiciel repose sur une base de données Oracle dont le schéma relationnel vous est communiqué ci-dessous.

PROMOTIONS (idPromotion, nomPromotion, *nbEtudiantsPromotion*)
GROUPES (idGroupe, idPromotion#)
ETUDIANTS (idEtudiant, nomEtudiant, prenomEtudiant, sexeEtudiant, dateNaissanceEtudiant, idGroupe#)
SEMESTRES (idSemestre, dateDebutSemestre, dateFinSemestre, idPromotion#)
MODULES (idModule, nomModule, idSemestre#, *coefficientModule*)
MATIERES (idMatiere, nomMatiere, idModule#, coefficientMatiere)
NOTES (idEtudiant#, idMatiere#, note)

La table *Promotions* recense les différentes promotions d'étudiants de l'IUT (en l'occurrence, la promotion de première année et celle de seconde année) et la table *Groupes* inventorie tous les groupes de TD de ces promotions (par exemple, les groupes T1, T2 et T3 en première année, et les groupes D1 et D2 en seconde année). Dans la table *Etudiants*, on peut trouver toutes les informations qui concernent les étudiants qui sont affectés à ces groupes de TD.

La table *Semestres* nous indique à quelles dates commencent et se terminent les semestres 1, 2, 3 et 4 et à quelle promotion ils sont rattachés ; par exemple, les semestres 1 et 2 sont rattachés à la première année, et les semestres 3 et 4 à la seconde année.

Dans la table *Modules* on peut trouver tous les modules dispensés pendant les différents semestres. Par exemple, le module 111 "Initiation Informatique" est dispensé au premier semestre (de la première année) ; alors que le module 211 "Programmation Web" est dispensé au troisième semestre (de la seconde année).
La table *Matières* permet de connaître les matières qui sont enseignées (et donc évaluées) à l'intérieur d'un module. Par exemple, le module 212 "Culture Générale Informatique" (du semestre 3 de la deuxième année) comprend les matières « UE15 Refactoring Agile », « UE16 Qualité & Tests » et « UE17 eXtreme Programming ». Enfin, la table *Notes* permet de connaître les notes qu'ont obtenues les étudiants dans les différentes matières.

Vous trouverez sur le moodle le fichier « Périgée.sql » qui vous permettra de générer les tables de la base de données sur votre schéma.

Dans les tables *Promotions* et *Modules* figurent respectivement les attributs `nbEtudiantsPromotion` et `coefficientModule` qui sont indiqués en italique dans le schéma relationnel. Ces attributs sont dits dérivés (au sens UML), c'est-à-dire qu'ils sont calculables à partir de requêtes SQL. Ils engendrent donc de la redondance dans la base de données mais ont tout de même été maintenus car leur contenu est relativement stable. Toutefois, dans le script « Périgée.sql » leur valeur n'a pas encore été initialisée et sur toutes les lignes, ces attributs ont pour le moment la valeur `NULL`.

Première partie – Les Blocs PL/SQL :

1) Bloc PL/SQL simple et paquetage `DBMS_OUTPUT`.
   - Ecrire un bloc PL/SQL qui, grâce au paquetage `DBMS_OUTPUT`, affiche le nombre d'étudiants qui se trouvent dans le groupe qui a pour identifiant 'T1'.

     On rappelle que pour que le paquetage `DBMS_OUTPUT` fonctionne dans l'interface iSQL*Plus, il faut l'activer avec la commande `SET SERVEROUTPUT ON` avant la première instruction du bloc PL/SQL (cette option sera valable durant toute la session iSQL*Plus).
   - Résultat attendu :

```
Il y a 6 étudiant(s) dans le groupe T1
```

2) Interactivité grâce aux variables de substitution.
   - En s'inspirant du bloc PL/SQL écrit lors de la question précédente, écrire un nouveau bloc qui affiche le nombre d'étudiants d'un groupe dont l'identifiant a été saisi au clavier.

     Pour cela on utilisera les variables de substitution vues en cours. On rappelle qu'à l'intérieur du `BEGIN … END`, les variables de substitution qui font référence à une chaîne de caractères doivent être encadrées par des 'quotes'. Par exemple, `'&s_idGroupe'`
   - Résultats attendus :

```
Entrer l'identifiant du groupe : T1
Il y a 6 étudiant(s) dans le groupe T1

Entrer l'identifiant du groupe : T2
Il y a 5 étudiant(s) dans le groupe T2

Entrer l'identifiant du groupe : T4
Il y a 0 étudiant(s) dans le groupe T4
```

3) Les Exceptions : l'exception `NO_DATA_FOUND`.
   - Dans la question précédente, si l'identifiant du groupe n'existe pas (par exemple le T4), il est indiqué qu'il y a 0 étudiant dans le groupe en question. Et on aurait eu la même réponse pour un groupe qui existe bien mais qui n'a aucun étudiant affecté.

     En s'inspirant du bloc PL/SQL écrit lors de la question précédente, écrire un nouveau bloc qui indiquerait le cas échéant que le groupe n'existe pas.
   - Pour cela, vous utiliserez l'exception prédéfinie `NO_DATA_FOUND` qui est levée automatiquement lorsque une requête `SELECT … INTO` ne retourne rien (il est à noter que si la requête retourne plusieurs lignes, c'est alors l'exception `TOO_MANY_ROWS` qui est levée).
   - Résultats attendus :

```
Il y a 6 étudiant(s) dans le groupe T1

Il n'y a pas de groupe T4
```

4) Structures de contrôle : la conditionnelle (`IF … THEN … ELSE … END IF`).
   - Ecrire un nouveau bloc PL/SQL qui fait exactement la même chose que la question précédente mais en utilisant cette fois-ci la structure de contrôle (`IF … THEN … ELSE … END IF`) à la place de la section `EXCEPTION`.

5) Les variables `%ROWTYPE`.
   - Ecrire un bloc PL/SQL qui permet d'afficher toutes les informations nominatives de la table *Etudiants* qui concernent l'étudiant E1.

     Pour se simplifier la tâche, et pour faciliter les éventuelles futures maintenances de l'application, on récupèrera les données qui concernent l'étudiant E1 dans une variable `%ROWTYPE`.
   - Résultat attendu :

```
Identifiant étudiant : E1
Nom étudiant : Alizan
Prénom étudiant : Gaspard
Sexe étudiant : M
Date naissance étudiant : 14.07.1997
Groupe étudiant : T1
```