Université Paris 13 — Année 2016-2017
ING2 — Bases de Données

TP3
SQL3

Objectif : Utiliser l'objet-relationnel introduit dans SQL3
On désire implémenter la base de données Ecole en SQL3.

1 Création de types et de tables

1. Créez un type `adresse_type` avec un numéro de rue, un nom de rue et un nom de ville.

   NB : Il faut terminer la définition du type par une ligne finale qui ne contient qu'un "/". Ne terminez pas la définition par un ";". Si vous avez des erreurs de compilation d'un type, vous pouvez faire afficher une description des erreurs en tapant "show errors" dans sqlplus.
2. Créez un type `personne_type` avec un nom et un prénom.
3. Créez les types `activites_type` et `cours_type`.
4. Créez les tables `personnes`, `activites` et `cours` associées à ces 3 types (`personne_type`, `activites_type` et `cours_type`). N'oubliez pas les contraintes d'intégrité (au moins les clés primaires).
5. Utilisez `describe` pour voir les descriptions des types et tables que vous venez de créer.

2 Ajout et modification de données, requêtes

1. Ajoutez des données dans les trois tables (`personnes`, `activites` et `cours`) en utilisant les mêmes données que celles de la base en SQL2.
2. Vérifiez qu'il s'agit bien de tables objets et non de tables relationnelles en consultant les tables `user_tables` et `user_object_tables`.
3. Ecrire les requêtes suivantes :
   (a) Liste des cours avec toutes les informations associées
   (b) Nombre d'équipe par activité
   (c) Liste des cours dont le nombre d'heures est supérieure ou égale à 25

4. Ajouter une activité `ski` pour l'équipe `Ace Club` (niveau 1)
5. Passer l'équipe `Avs80` au niveau 3 en `Volley ball`

3 Héritage

On désire définir les types `eleve_type` et `professeur_type`. Ceux-ci hériteront du type `personne_type`.

1. Le type `personne_type` doit être modifié (la clause `NOT FINAL` doit être ajoutée pour permettre l'héritage - spécialisation du type). On désire également pouvoir respecter la contrainte de couverture (aucune personne ne peut être ni étudiant, ni professeur).

   Comme un type ne peut être modifié si des tables ou d'autres types l'utilisent, vous devez donc supprimer la table `personnes` avant de modifier le type `personne_type`.

   Proposez une nouvelle définition du type `personne_type`
2. Créez un type `professeur_type` qui hérite de `personne_type` et qui possède les attributs `specialite`, `date_entree`, `der_prom`, `salaire_base` et `salaire_actuel`
3. Créez un type `eleve_type` qui hérite de `personne_type` et qui possède les attributs `date_naissance`, `poids`, `annee` et `adresse` (`adresse` étant du type `adresse_type`).
4. Créez les tables `eleves` et `professeurs`
5. Ecrivez les triggers permettant d'assurer la contrainte de partition sur les tables `eleves` et `professeurs` (on ne veut pas qu'un élève et un professeur est le même numéro). Testez le trigger avec des insertions.
6. Insérez des données dans les tables `eleves` et `professeurs`.
7. Affichez la liste des professeurs avec toutes les informations associées.

4 Collections

1. On désire définir une UE qui est composée d'un nom et de plusieurs cours (au plus 5).

   Créer le type `UE_type` en utilisant des tableaux pré-dimensionnés (`VARRAY`)
2. Créez la table `UE` et insérez des données

3. Modifiez le type `eleve_type` et la table `eleves` de manière à disposer d'un attribut `resultat`. Cet attribut est une table imbriquée (`nested table`) composée de deux attributs `nom_cours` et `points`.
4. Insérez des données dans la nouvelle table `eleves` en utilisant les données de la base en SQL2.
5. Ajoutez à chaque étudiant, un cours `Service Web` ainsi qu'une note associée à ce cours
6. Affichez les étudiants dont la note du cours d'Analyse est supérieure ou égale à 10.