TP 5 : Procédures, Fonctions et Triggers

Exercice 1 :

Nous considérons le schéma relationnel suivant modélisant les activités d'une agence de location des voitures.

Clients (**idclt**, nom, adresse, nbvoiturelouee)

Voitures (**idv**, dateconstruction, #idmarque)

Marques (**idmarque**, nomMarque, #idpays)

Locations (**idloc**, dateloc, duree, #idclt, #idv)

1. Définir un programme PL/SQL nommé Q1 qui ajoute une ligne à la table locations.
2. Définir un bloc PL/SQL nommé Q2 qui affiche pour chaque client le numéro et la marque de la dernière voiture louée.
3. Définir un bloc PL/SQL nommé Q3 qui affiche les numéros des clients qui ont réservé seulement des voitures d'une seule marque.
4. Définir un programme PL/SQL nommé Q4 permettant de savoir si deux clients ont loué exactement les mêmes voitures. Retourne oui si c'est le cas non dans le cas contraire.
5. Définir un bloc PL/SQL nommé Q5 affichant les noms des clients qui ont effectué des locations les plus longues.
6. Lorsque la table locations est manipulée, les ***nbvoiturelouQee*** des clients doivent rester cohérents avec les données existant dans la table locations. Ecrire le déclencheur assurant cette cohérence dans le cas suivants :
    a. Lorsqu'on ajoute un tuple.
    b. Lorsqu'on supprime un tuple.
    c. Lorsqu'on modifie les valeurs des attributs
7. Définir un bloc PL/SQL nommé Q7 permettant de compter le nombre des clients effectuant des locations des voitures des toutes les marques.

Exercice 2 :

Nous considérons le schéma relationnel suivant modélisant les activités d'un hôtel.

Hotel (**idHotel**, nom, ville)

Chambre (**idChambre**, #idHotel, #idtype, price, Nb_réservation)

Types (**idtype**, nom, tarif)

Réservation (**idChambre**, DateDebut, DateFin, #idVisiteur)

Visiteur (**idVisiteur**, nom, ville)

1. Définir un programme PL/SQL nommée Q1 permettant de savoir si deux visiteurs ont réservé exactement des chambres des mêmes types.

2. Définir un programme PL/SQL nommée Q2 qui affiche les noms de trois premiers visiteurs qui ont effectué le plus grand nombre des réservations.
3. Définir un bloc PL/SQL nommée Q3 retournant le nombre des chambres qui n'ont pas été réservé depuis 1 Juin 2009.
4. Définir un bloc PL/SQL nommée Q4 qui affiche les numéros des visiteurs qui ont réservé des chambres de tous les types.
5. Définir un bloc PL/SQL nommée Q5 affichant pour chaque chambre, le nom du visiteur qui a effectué la dernière réservation.
6. Lorsque la table **Réservation** est manipulée, les ***Nb_réservation*** des chambres doivent rester cohérents avec les données existant dans la table **Réservation**. Ecrire le déclencheur assurant cette cohérence dans les cas suivants :
    a. Lorsqu'on ajoute un tuple.
    b. Lorsqu'on supprime un tuple.
    c. Lorsqu'on modifie les valeurs des attributs
7. Définir un programme PL/SQL nommée Q7 permettant de compter le nombre des visiteurs effectuant plus que deux réservations.

Exercice 3 :

Nous considérons le schéma relationnel suivant modélisant coupe du monde.

Equipe(**CodeEquipe**, Nationalite)

Stade(**CodeStade**, NomStade,NbPlaces)

Individu(**NoIndividu**, Nom, Prenom, #CodeEquipe)

Match(**IdMatch**,NbSpectateur,Date, #NoIndividu, #CodeStade)

Jouer(**#IdMatch, #CodeEquipe**)

But(**IdBut**, Minute, Type,#IdMatch,#NoIndividu)

A. Procédures stockées : Ecrire pour chaque question une procédure nommée <<P*numDeQuestion*>>

1. Afficher les différents pays qui ont participé à la coupe mondiale.
2. Afficher la liste des arbitres
3. Afficher les noms des stades qui ont pu accueillir plus de N spectateurs
4. Afficher pour chaque match le nombre de places vacantes dans le stade.
5. Afficher tous les noms des joueurs marocains qui ont participé à la coupe mondiale.
6. Afficher noms ordonnés des arbitres des matchs joués par une équipe donnée.
7. Afficher les noms des joueurs qui ont marqué au moins N buts d'un type donné.
8. Afficher pour une équipe le nombre de buts marqués.
9. Afficher le nom du joueur marocain qui a marqué le maximum de buts.
10. Ajouter à tous joueur marocain deux buts.

B. Les fonctions : Ecrire pour chaque question une fonction nommée <<F*numDeQuestion*>>

1. Ecrire une fonction qui prend en paramètre un numéro stade et qui renvoie une chaine de caractère contenant le nom du stade et le nombre de match joués sur ce stade.
2. Ecrire une fonction qui prend en entrée le nom d'un arbitre et produit une chaine contenant les matches qu'il arbitré.

C. Les triggers : Ecrire pour chaque question un trigger nommée <<T*numDeQuestion*>>

1. Lors de l'ajout d'un individu, on met son nom en majuscule ainsi que la première lettre de son prénom.
2. Lors de l'ajout d'un match on vérifie si le nombre de spectateurs est inférieur ou égal au nombre de places disponibles dans le stade. Si oui on l'ajoute sinon on affiche un message d'erreur.
3. Lors de l'insertion d'un but on vérifie si la minute est positive.
4. Lors de la suppression d'un joueur, on supprime aussi les buts qu'il a marqués.
5. Interdire de modifie la date d'un match.
6. Lors de la suppression d'une équipe on supprime les joueurs.
7. Lors de la suppression d'une équipe on supprime les joueurs et les buts.
8. Lors de l'ajout d'une équipe si on essaie d'insérer dans le champ nationalité :

   'Fr' alors on la modifie en : 'France'

   'Mr' alors on la modifie en : 'Maroc'

   'Br' alors on la modifie en : 'Brésil'

   'Esp' alors on la modifie en : 'Espagne'

- La table Equipe :

```
insert into Equipe values(0,'Maroc')
insert into Equipe values(1,'Tunisie')
insert into Equipe values(2,'Camerone')
insert into Equipe values(3,'Manga')
```

- La table Stade :

```
insert into Stade values(0,'Amerigo',1000)
insert into Stade values(1,'Orange',1000)
insert into Stade values(2,'Pricipaux',1000)
```

- La table Individu :

```
insert into Individu values(0,'N1','P1',0)
insert into Individu values(1,'N2','P2',0)
insert into Individu values(2,'N3','P3',0)
insert into Individu values(3,'N4','P4',0)
insert into Individu values(4,'N5','P5',0)
insert into Individu values(5,'N6','P6',0)
insert into Individu values(6,'N7','P7',0)
insert into Individu values(7,'N8','P8',1)
insert into Individu values(8,'N9','P9',1)
insert into Individu values(9,'N10','P10',1)
insert into Individu values(10,'N11','P11',1)
insert into Individu values(11,'N12','P12',1)
insert into Individu values(12,'N13','P13',1)
insert into Individu values(13,'N14','P14',1)
insert into Individu values(14,'NA','PA', null)
insert into Individu values(15,'NA2','PA2', null)
```

- La table Match :

```
insert into Match values(0,200,'23/07/2003',14,0)
insert into Match values(1,200,'23/07/2003',14,1)
insert into Match values(2,200,'23/07/2003',15,1)
```

- La table Jouer :

```
insert into Jouer values(0,0)
insert into Jouer values(0,1)
insert into Jouer values(2,1)
insert into Jouer values(2,2)
```

- La table But :

```
insert into But values(15,'Tir',0,1)
insert into But values(23,'Diving header',0,2)
insert into But values(44,'Freestyle',0,3)
insert into But values(79,'Long rage',0,7)
insert into But values(90,'Cross',0,9)
insert into But values(17,'Tir',0,1)
```

Exercice 4 :

Soit la table **Pilote** (NoPilote, Nompil, Adresse, Salaire, Prime, Embauche)

1. Ecrire le package pack_pilote comportant les procédures et fonctions publiques suivantes :
   a. Inser_pil (nom,adresse, datembauche) qui permet l'insertion d'un nouveau pilote dans la table avec numéro automatique, salaire égal à la moyenne du salaire des pilotes et commission à zéro.
   b. Sel_pil (nom,adresse,salaire) permettant de récupérer l'adresse et le salaire d'un pilote dont on fournit le nom.
   c. Surcharger la méthode Sel_pil pour n'obtenir que l'adresse d'un pilote à partir du nom.
   d. Prime_pil(nom,prime) permettant d'attribuer une prime à un pilote.
   e. List_pil() permettant d'afficher la liste de tous les pilotes avec adresse, salaire et prime.Sal_prim_pil() permettant d'obtenir le total des salaires avec les primes.
   f. Transformer le calcul du salaire moyen des pilotes en une fonction privée.
2. Ecrire un bloc PL/SQL pour tester et exécuter les méthodes du package.