GIS 2ème année - Devoir Surveillé S.G.B.D.

Durée 2 heures

1 Contraintes (5 points)

Soit la base de données suivante :

```
create table t1 (a integer primary key, b integer not null check (b> a));
create table t2 (a integer primary key, b integer unique);
create table t3 (a integer references t1, b integer references t2);
create table t4 (a integer references t1, b integer references t2,
                 primary key (a,b));
```

Pour chaque commande SQL suivante, précisez si l'exécution est possible ou non et, pour chaque exécution impossible, l'origine de l'erreur.

```
insert into t1 values (1,3) ;
insert into t1 values (2,3) ;
insert into t2 values (1,1) ;
insert into t2 (a) values (2) ;
insert into t3 values (1,1) ;
insert into t4 values (1,1) ;
insert into t3 values (1,1) ;
insert into t4 values (1,1) ;
insert into t3 values (2,1) ;
insert into t4 values (2,1) ;
delete from t1 where a=2 ;
delete from t2 where a=2 ;
update t1 set a=4 where a=1 ;
update t1 set a=2 where a=1 ;
update t3 set a=2 where a=1 ;
update t4 set a=2 where a=1 ;
```

2 Conception (9 pts)

Un capitaine d'une équipe sportive pour un championnat universitaire inter-promotions désire informatiser son système d'informations.
Les membres de l'équipe sont identifiés par un nom et peuvent éventuellement avoir un surnom. Les membres féminins ainsi que les membres 'profs' apportant chacun un bonus, il est donc nécessaire de disposer de ces deux informations.
Le championnat est décomposé en 5 épreuves : "volley-ball", "sports de raquettes", "football", "basket", et "biathlon". Une épreuve se déroule sur une journée. Certaines épreuves peuvent se décomposer en plusieurs activités. Ainsi l'épreuve "sports de raquettes" se décompose en les activités suivantes : "tennis", "tennis de table" et "badminton". Pour les épreuves ayant une seule activité, le nom de l'épreuve correspond au nom de l'activité. Chaque épreuve se déroule à un endroit précis de l'université. Pour chaque activité, un nombre maximum de membres est imposé. Le capitaine désire automatiser la phase d'inscription de ces membres aux différentes épreuves. Chaque membre (qui est également un utilisateur de la base) peut donc "alimenter" la base en s'inscrivant à telle ou telle épreuve.

2.1 Question 1

Proposez un schéma conceptuel (avec le modèle MERISE MCD ou UML) correspondant à ce système d'informations.

Pour des raisons de sécurité, le capitaine ne veut montrer (en lecture) à ses membres qu'une partie du système d'informations qui pourra avoir la forme suivante :

```
les_epreuves(nom_epreuve,nom_activité,          journée,   lieu)
membres_inscrits(nom_membre,          nom_activité)
```

En écriture, chaque membre n'a que la possibilité de s'inscrire.
Donner les différentes commandes SQL d'administration pour créer les tables et sécuriser le système.

2.3 Question 3

Le capitaine veut offrir un programme client qui effectue la phase d'inscription. Ce programme est écrit en Java et utilise le protocole JDBC. Il se lance de la manière suivante :

```
java  Inscription   nomMembre   motDePasseMembre     nomActivité
```

Le programme a pour but d'inscrire dans la base le membre `nomMembre` à l'activité `nomActivité`. Cela est possible si les conditions suivantes sont réunies :
– le membre est référencé dans la base.
– l'activité existe
– le nombre max de membres inscrits pour l'activité n'est pas atteint.
Ecrire le programme java `Inscription` en s'assurant bien que la base de données reste cohérente. En particulier, il faut prévoir le fait que plusieurs membres peuvent s'inscrire au même moment.

3 Procédure stockée (6 pts)

3.1 Sujet

Le MCD de la figure 1 décrit une entité *pièce* qui comporte deux attributs : un *numéro* qui permet d'identifier la pièce ainsi que son *prix*. Une pièce peut être une pièce simple ou alors une pièce complexe composée d'autres pièces également décrites dans l'entité pièce. L'association réflexive *composéDe* permet de décrire des pièces complexes. L'attribut *nombre* de l'association spécifie le nombre de "sous-pièces".

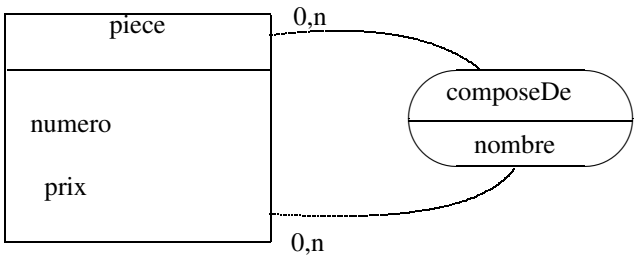

FIG. 1 – Modèle Conceptuel

3.2 Création de la base

Soit le cas de figure suivant :
– Les pièces 1 et 2 sont des pièces simples et valent respectivement 3 francs et 5,50 francs.
– La pièce n°3 est composée de 2 pièces n°1.
– La pièce n°4 est composée de 3 pièces n°3 et de 2 pièces n°2.
Voici un schéma relationnel conforme au schéma conceptuel précédent et au cas de figure ci-dessus :

```
create  table  piece  (numero  integer  primary  key,
                      prix  float)  ;
create  table  compose_de   (num_piece   integer   references   piece,
```

```
                        num_sous_piece    integer  references  piece) ;
insert into piece values (1,3.0) ;
insert into piece values (2,5.5) ;
insert into piece (numero) values (3) ;
insert into piece (numero) values (4) ;
insert into compose_de values (3,2,1) ;
insert into compose_de values (4,2,2) ;
insert into compose_de values (4,3,3) ;
```

3.3 Programmation

Ecrire, dans le langage plpgsql, la fonction `calcule_prix` à un paramètre de type `integer` correspondant à un numéro de pièce. Cette fonction a pour but de fixer le champ prix de l'entité pièce dont le numéro est passé en paramètre et de toutes ses pièces composantes. Ce prix est la somme des prix des pièces composantes. La fonction retourne le prix de la pièce. Pour une pièce simple, la fonction retourne le prix de la pièce, sans effectuer de mise à jour dans la base.
Par exemple, l'exécution de la commande SQL suivante a pour effet de positionner le prix de la pièce 3 à 6 francs, et le champ prix de la pièce 4 à 29 francs :

```
select calcule_prix(6) ;
calcule_prix
------------
          29
```