Soit la base de données d'une entreprise gérant les employés de différents services affectés à des projets

Employé (Nemp, Nomemp, Salairemp, Postemp, Datemb, *Nserv*)
Service (Nserv, Nomserv, Localisationserv, *Directeurserv*)
Projet (Nproj, Titreproj, *Responsableproj*, Datedeb, Budgetproj, *Nserv*)
Affectation (Nemp, Nproj, Date-aff, Montant-Indemnité)

NB : Les clés étrangères sont en italique.

Questions :

1. L'administrateur de la BD créé un utilisateur Ut1 qui peut lire et modifier le contenu de la relation Projet et un autre utilisateur Ut2 qui peut lire et modifier les relations Projet et Employé peut modifier le schéma de ces relations et il peut même transmettre ce pouvoir à d'autres utilisateurs. La table Service peut être accédée en lecture par tous les utilisateurs de la base. Donner les requêtes utilisées par l'administrateur pour répondre à ce besoin ?

Create user Ut1 Identified by xxx
Grant SELECT, UPDATE on Projet to Ut1 (1pt)
Create user Ut2 Identified by yyy
Grant SELECT, UPDATE, ALTER on Projet, Employé to Ut2 With Grant Option (1pt)
Grant SELECT ON Service to PUBLIC (0.5pt)

2. L'utilisateur Ut1 peut-il accéder à la relation Employé ?

L'utilisateur n'a pas le droit d'accéder à la table Employé sauf si Ut2 lui transmette les privilèges qu'il a sur cette table. (1pt)

3. On veut rajouter le nombre d'employés affectés à un projet. Donner l'instruction SQL permettant de satisfaire ce besoin.

Alter Table Projet Add (nb_empnumber). (0.5pt)

4. Exprimer la contrainte permettant de mettre à jour automatiquement ce nombre d'employés lors de l'affectation d'un employé à un projet ou lorsqu'il quitte un projet.

```
CREATE OR REPLACE TRIGGER MAJ_NB_EMP (2pt)
AFTER DELETE OR INSERT ON Affectation
FOR EACH ROW
Begin
IF INSERTING THEN
Update Projetset NB_Emp=NB_Emp+1 where projet.Nproj= :NEW.Nproj;
ELSIF DELETING THEN
Update Projet set NB_Emp=NB_Emp-1 where projet.Nproj= :OLD.Nproj;
END IF
End
```

5. Supposons que le SGBD sur lequel la BD est créée ne supporte pas les contraintes type clés étrangères. Proposez une solution permettant, lors de la création d'un projet dans un service de vérifier que cenuméro de service existe bien dans la table Service. Dans le cas contraire, l'opération d'ajout est rejetée. Donner la requête correspondante.

La solution consiste à créer une assertion ou un trigger qui lors de l'insertion vérifie que le numéro de services mentionné existe bien dans la table service (0.5 pt)

Solution Avec Assertion

```
ASSERT CI-ClEEtrangère (SELECT Nserv
FROM Projet)
IS IN (SELECT Nserv
 FROM Service);
```

Solution Avec Trigger

```
CREATE OR REPLACE TRIGGER Verif_Foreign_Ref (1.5pt)
 BEFORE INSERT ON Projet
FOR EACH ROW
SrvcService.Nserv%Type.
Begin
Select Nserv into srvc from Service where Nserv=:NEW.Nserv
If (srvc IS NULL) thenraise_application_error('-20055, Insertion impossible, le projet est affecté à un service inexistant!');
End If
End
```

6. Ecrire la contrainte permettant de spécifier qu'un employé ne peut être affecté qu'à un projet du même service.

```
CREATE OR REPLACE TRIGGER Verif_Service (1.5pt)
 BEFORE INSERT ON Affectation
 FOR EACH ROW
SrvcService.Nserv%Type.
Begin
Select Nserv into srvc from Employé E, Projet P where E.Nserv=P.Nserv and E.Nemp=:NEW.Nemp and P.Nproj=:NEW.Nproj
If (srvc IS NULL) then raise_application_error('-20055, Insertion impossible, le projet est affecté à un service inexistant!');
End If
End
```

7.

a. Donner la requête permettant de s'assurer que l'indemnité accordée à un employé lors de son affectation à un projet ne dépasse pas le budget du projet. (1pt)

Solution Avec Assertion

```
ASSERT CI-Ind on Affectation A : Montant-Indemnité<select budgetproj  from Projet where Nproj=A.Nproj
```

Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département d'Informatique

Concours d'accès au Doctorat LMD Informatique, 2014/2015

USTHB le 15/10/2014

Solution Avec Trigger

```
CREATE OR REPLACE TRIGGER Verif_Budget (1pt)
 BEFORE INSERT ON Affectation
 FOR EACH ROW
Bdg Projet. Budgetproj %Type.
Begin
Select Budgetproj into Bdg from Projet where Nprj=:NEW.Nprj;
If (Bdg< :NEW.Montant-Indemnité) then raise_application_error('-20055, 'Insertion impossible,
l'indemnité accordée dépasse le budget du projet !');
End If
End
```

b. A quelle moment doit-on définir cette requête pour qu'elle soit la plus efficace. (0.25 pt)

Il est conseillé de définir cette contrainte dès la création de la relation Affectation avant toute insertion pour ne pas avoir un tuple déjà inséré où l'indemnité de l'employé est supérieur de au budget du projet

c. Pouvons-nous la définir à d'autres moments ? (0.25 pt)

Oui puisque la définition des contraintes est dynamique. On peut la définir à tout moment de la vie d'une BD

d. Dans ce cas la requête peut-elle être rejetée ? (0.25 pt)

Oui, pour une assertion, elle pourra être rejetée s'il y a un employé qui a été inséré avec un salaire supérieur au budget du projet où il travaille avant que la ci n'ait été définie.

Pour un triggers, il commence la vérification après sa création, donc, il ne sera pas rejeté.

e. Justifier votre réponse. (0.25 pt)

Voir plus haut

f. Si elle est acceptée, qu'est-ce qui permet au SGBD de la vérifier ? (0.5pt)

La définition de la CI (assertion ou trigger) est insérée dans un catalogue et à chaque insertion ou modification d'une affectation, le SGBD accède au catalogue et ajoute cette contrainte à ces commandes.