USTHB – Faculté d'Electronique et Informatique
Département Informatique

Administration des SGBD
TP Triggers

RELARQUE IMPORTANTE

Pour l'ensemble des TP, on demande de réaliser un compte-rendu du travail effectué. Ce compte-rendu comprendra un rapport (manuscrit ou réalisé sur traitement de texte) détaillant les points intéressants abordés à chaque question :

- requêtes SQL correspondant aux questions posées
- résultat de l'exécution de la requête
- réponses aux questions soulevées dans le sujet,
- explication des parties non triviales des requêtes complexes,
- analyse et commentaire des jeux d'essais.

TRIGGERS

Syntaxe de création d'un trigger

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[FOR EACH ROW]
WHEN (condition)
BEGIN
  --- Instruction PLSQL
END;
```

- ***CREATE [OR REPLACE] TRIGGER trigger_name*** *: pour créer ou écraser un trigger existant.*
- ***{BEFORE | AFTER | INSTEAD OF }*** : le moment du déclenchement du trigger (avant ou après l'opération de mise à jour).
- ***{INSERT [OR] | UPDATE [OR] | DELETE}*** : l'événement de mise à jour qui provoquera le déclenchement du trigger. Plusieurs événements séparés par OR sont possibles.
- ***[OF col_name]*** *: utilisé dans le cas de l'opération Update appliquée sur une colonne particulière.*
- ***[ON table_name]*** *: le nom de la table sur laquelle le trigger est défini*.
- ***[FOR EACH ROW]*** *: spécifie si le trigger est lancé pour chaque ligne affecté ou une seule fois.*
- ***WHEN (condition)*** *: le trigger est lance seulement lorsque la ligne affectée vérifie la condition.*

Remarque *: pour générer une exception et empêcher le programme de continuer, l'utilisateur peut lancer la procédure* ***raise_application_error (-Num_Message, 'Message à Afficher') ;*** *Num_Message est compris entre 20000 et 20999.*

Questions

Supposons que les tables des TP précédents sont crées et remplies.

1. Créez un trigger qui affiche **« un nouveau employé de type infirmier est ajouté»** après chaque insertion d'un infirmier. Répétez la même chose pour la modification ou la suppression.
2. Créez un trigger qui affiche **« un nouveau infirmier est affecté à un [Nom de service] »** après chaque insertion d'un infirmier.

3. Créer un triggers qui vérifie avant modification du **code service** dans la table **infirmier** que la nouvelle valeur existe réellement, sinon, il refuse l'opération.

4. Créer un trigger qui vérifie que lors de la modification du **salaire** d'un infirmier, la nouvelle valeur ne peut jamais être inférieure à la précédente.

5. L'administrateur veut, pour un besoin interne, avoir le total des salaires infirmiers pour chaque **service**. Pour cela, il ajoute un attribut : **total_salaire_service** dans la table service.
    a. Ajoutez l'attribut.
    b. Créez un trigger **TotalSalaire_Serivce_trigger** qui mit à jour l'attribut **total_salaire_service** après l'insertion d'un infirmier.
    c. Créez un trigger **TotalSalaireUpdate_trigger** qui mit à jour l'attribut **total_salaire_service** après la mise à jour d'un salaire.

6. Un infirmier peut changer de service. Créer un trigger qui mit à jour l'attribut **total_salaire_service** des deux services.

7. L'administrateur veut sauvegarder toutes les hospitalisations des patients dans le temps. A chaque fois un patient est hospitalisé une ligne sur les informations de l'hospitalisation est sauvegardée dans une autre table « **Hist_Hospit** ». La table « Hist_Hospit » est définie par **Hist_Hospit (<u>date_hospit,num_patient</u> code_service*)**. Où date_hospit est la date d'hospitalisation.