

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 lancé 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 créées 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\_Service\_trigger** qui met à jour l'attribut **total\_salaire\_service** après l'insertion d'un infirmier.
  - c. Créez un trigger **TotalSalaireUpdate\_trigger** qui met 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 met 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 (date\_hospit,num\_patient code\_service\*)**. Où date\_hospit est la date d'hospitalisation.