**TP sur les triggers**

**Exercice 1 :**

Créer un trigger « employe\_insert » qui permet de vérifier si le salaire d’un employé est <5000. Si le cas se présente, il insère automatiquement 5000.

**Exercice 2 :**

Créer un trigger « employé\_embauche» qui permet de vérifier, lors de l’insertion d’un employé, si la date d’embauche est inférieure à la date système, dans le cas échéant, afficher un message d’erreur «La date d’embauche est supérieure à la date système».

Insérer dans la base l’enregistrement suivant pour vérifier votre solution.

**Exercice 3 :**

Écrire un trigger qui permet de contrôler les ajouts et les modifications des salaires. Si le JOB n’est pas ‘PRESIDENT’, alors Lors de l’ajout, modification d’un nouvel employé, nous devons vérifier si le salaire de celui-ci se retrouve dans la fourchette des salaires. Les salaires sont en fonction de JOB et sont dans la table Grille.

**Exercice 4 :**

Pour archiver l’historique des opérations de mise à jour de la table Employees, on a crée une table History (employee\_id, operation\_date, operation\_type). Ecrire un trigger permettant, après l’exécution de chaque opération sur Employees, d’ajouter un tuple concernant cette opération à la table History.

**Exercice 5 :**

Soit le schéma relationnel suivant :

ETUDIANT (numEtud, nom, prenom , datenaiss, civilite, numsecu);

MODULE (codMod, nomMod, effecMax);

EXAMEN (codMod#, codExam, dateExam);

INSCRIPTION (numEtud#, codMod#, dateInsc);

PREREQUIS (codMod#, codModPrereq, noteMin);

RESULTAT (codMod#, codExam#, numEtud#, note)

1. **Créer la base de données qui correspond au schéma relationnel ci-dessus tout en prenant en considération les contraintes décrites dans le tableau suivant :**

|  |  |
| --- | --- |
| **Colonne** | **Contrainte** |
| numsec | non nul |
| effecMax | valeur par défaut 30 |
| dateInsc | valeur par défaut est la date système |
| noteMin | non nul |
| Civilite | appartient à cette liste ('Mr', 'Mme', 'Mlle') |

1. **Créer les triggers qui permettent de répondre à ces règles de gestion**
2. Il ne doit pas être possible de modifier la note min dans la table prerequis.
3. Dans un module, il ne doit pas y avoir plus de effecMax élèves inscrits.
4. On ne peut créer un examen pour un module que s’il y a des élèves inscrits dans ce module.
5. Un élève ne peut passer un examen que si sa date d’inscription est antérieure à la date de l’examen.
6. Un élève s’inscrivant à un module doit avoir eu au moins la note min à tous les modules pré-requis.
7. Créer une vue de résultats, qui contiendra le nom et la moyenne de chaque étudiant s’il a passé les examens de tous les modules dans lesquels il est inscrit.
8. Revenez sur la première contrainte : il ne doit être possible de modifier une note min dans la table prerequis que s’il n’existe pas d’élève dont une inscription serait invalidée.
9. Il ne doit être possible de modifier effecMax que si des étudiants ne se retrouvent pas avec une inscription invalidée.