TP : Interrogation d'une base de donnée relationnelle

On considère la base de données suivante fournie par Oracle

EMP ***de clé primaire EmpNo et de clé étrangère DeptNo***

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SMI	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10
7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
[illegible]	FORD	ANALYST	7566	03/12/81	3000		20
[illegible]	MILLER	CLERK	7782	23/01/82	1300		10

SalGrade

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

1. Connectez-vous au compte SMI et affichez les tables, les triggers et les procédures possédées par SMI.
2. Donner la liste des employés ayant des salaires supérieurs à 1000
3. Donner la liste des employés travaillant dans le département SALES
4. Donner le nom de chaque employés ainsi que son salaire total (incluant sa commission)
5. Afficher les noms des employés et leurs commissions. Si un employé n'a pas de commission, afficher sans commission. Etiqueter cette colonne Commission (alias).
6. Afficher les noms des employés, leur date d'embauche et la date de la première négociation de leur salaire, qui est le premier Lundi après 6 mois de travail, nommer cette colonne « Date Négociation ».
7. Écrire une requête qui donne le résultat suivant, pour chaque employé :
   < nom_employé > gagne < salaire >euros par mois, mais souhaite < salaire * 3 > dhs.
   Nommer cette colonne : Salaire souhaité. (L'opérateur || (double barre verticale) permet de concaténer des champs de type caractères)

8. Afficher pour chaque employé le nom, la date d'embauche et le jour de la semaine auquel il a commencé à travailler. Ordonner le résultat suivant le jour.
9. Un fichier **fichier1.sql** que vous créerez à l'aide de l'éditeur sql (et non un fichier texte), sera appelé pour exécution par '**@fichier1**' avec son chemin d'accès complet. Ecrire la requête qui affiche le nom de chaque employé et le nom et le code de son chef. Appeler les colonnes Employé, NoEmp, Chef, NoChef. Insérer cette requête dans fichier1.sql à exécuter.
10. Afficher le nom, le métier, le nom du département, le salaire et la catégorie du salaire de chaque employé. (On utilisera aussi la table SALGRADE).
11. **Afficher par métier** la somme des salaires pour chaque département, ainsi que le total des salaires de tous les départements, en utilisant 2 méthodes : GROUP BY, DECODE. On ne considèrera que les départements qui contiennent des employés des départements : 10, 20 et 30.

Job	Dep10	Dep20	Dep30	Total
Analyst				
Manager				

12. Obtenir les numéros des départements où il n'y a pas d'employés, en utilisant la clause MINUS.
13. Donner le nom du plus ancien employé.
14. Afficher par ordre croissant les noms des 3 employés qui ont les salaires les plus élevés.
15. Afficher des informations complètes concernant les départements où il n'y a aucun vendeur (SALESMAN)
16. Afficher le numéro et le nom des employés qui gagnent plus que le salaire moyen et qui travaillent dans le même département qu'au moins un employé dont le nom contient la lettre 'T'.

TP : PL/SQL

A. Triggers

1. Créer un trigger qui avant de mettre à jour l'âge d'un étudiant donné, le nouvel âge saisi doit être supérieur à l'ancien.
2. Créer un trigger qui avant de mettre à jour le salaire d'un employé donné, le nouveau salaire saisi doit être supérieur à l'ancien.
3. Définir un trigger en insertion permettant d'implémenter une numérotation automatique de la colonne **NUM_ETUDIANT** de la table Etudiant. Le premier numéro doit être égal à 1.
4. Il s'agit de réaliser la gestion d'une base de données commerciale des produits informatiques. Les tables de la base de données sont définies ci-dessous :
   PRODUIT(**NO_PROD**, DESIGN_PROD, PRIX_UNITE, STOCK), COMMANDE(**NO_CMD**, NO_PROD, DATE_CMD, QUANTITE, MONTANT).
   Pour des raisons de simplicité, on considère qu'une commande concerne un seul produit, et qu'on peut commander un produit plusieurs fois.
   Définir un trigger qui doit mettre à jour automatiquement les quantités en stock de la table produit.
   NB : On ne pourra de plus pas enregistrer une commande que l'on ne pourrait satisfaire pour cause de stock insuffisant.
5. Donner la requête qui permet d'activer/désactiver tous les triggers liés à la table ETUDIANT.
6. Donner la requête qui permet d'activer/désactiver un trigger nommé « Trigger_SMI ».

B. PL/SQL et Curseurs

1. On souhaite insérer un nouvel enregistrement dans la table BUDGET sans avoir à saisir le numéro de l'opération et le nouveau solde. Écrire un programme PL/SQL qui ajoute un nouvel enregistrement à la table BUDGET avec les valeurs suivantes :
   - NUM_OPERATION : calculé automatiquement en ajoutant 1 à la plus grande valeur de NUM_OPERATION présent dans la table BUDGET.
   - NOM_OPERATION : Courses
   - CATEGORIE : Depense
   - DATE_OPERATION : 14/01/2002
   - MONTANT : 500 (en dirhams)
   - SOLDE : calculé automatiquement en fonction du dernier solde de la table BUDGET (celui pour lequel NUM_OPERATION est le plus grand).
2. On souhaite à présent effectuer des traitements sur plusieurs colonnes en même temps. L'objectif de cet exercice est de construire une table COURSES, qui contient toutes les opérations pour lesquelles le nom de l'opération est 'Courses', à partir de la table BUDGET.
   La table COURSES doit contenir les champs suivants : NUM_OPERATION, NOM_OPERATION, CATEGORIE, DATE_OPERATION et MONTANT.

Pour cela, il faut définir un curseur qui va parcourir toute la table BUDGET. Pour chaque enregistrement que va traiter le curseur, le programme devra déterminer s'il doit être inséré dans la table COURSES ou non.

3. On souhaite créer une table BUDGET_EURO avec toutes les opérations réalisées depuis le passage à l'Euro (c'est-à-dire depuis le 01/01/2002) avec les montants des opérations exprimés en Euros et non plus en dirhams (le taux de conversion est 11).
   Créez une nouvelle table BUDGET_EURO, définissez un curseur qui traite toutes les opérations réalisées depuis le 01/01/2002 et insérez dans la table BUDGET_EURO ces opérations avec le montant exprimé en Euros.
   Attention : comme il s'agit d'une nouvelle comptabilité, les numéros des opérations sont réinitialisés (numérotés à partir de 1).
4. Créer une table BUDGET_SEUIL(NUM_OPERATION, DATE_OPERATION, MONTANT) dont on enregistre tous les opérations de débit qui contiennent un montant dépassant un seuil (saisi par l'utilisateur), en utilisant :

   - Un curseur paramétré.
   - Un curseur implicite.

C. Procédures, Fonctions et Packages

1. Ecrire une fonction qui permet de retourner le nom du département, prenant en paramètre le numéro du département.
2. Refaire la même question en utilisant une procédure, prenant en paramètre le numéro du département et le nom du département.
3. Ecrire une procédure (en utilisant les curseurs et la fonction de la question 1) qui permet d'afficher les noms des employés sous la forme suivante:
   L'employé <<Nom employé>> a la profession <<Profession>> dans le département <<Nom département >>.
4. Ecrire une fonction, prenant en paramètre le numéro de l'employé, qui permet d'afficher et de retourner le nom de l'employé (l'affichage est de même forme que la question 3).
5. Ecrire une fonction prenant en paramètre le numéro de l'employé et retournant le nom de son chef, s'il s'agit du chef, la fonction retourne "Aucun".
   NB: utiliser la question 4.
6. Ecrire une procédure qui permet d'augmenter les salaires (sans tenir compte la commission) des employés selon leurs grades.

   - 10% pour le grade 1 et grade 2.
   - 15% pour le grade 3.
   - 20% pour le grade 4 et grade 5.

7. Donner une fonction qui permet de retourner le salaire maximum pour un département donné
8. Ecrire un package gestion_emp permettant de regrouper les procédures et les fonctions créées dans les sept questions et tester. (**DEVOIR à rendre par email.)**

Gestion de Projet : TP
SMI S6
Travail à rendre

Objectif :

Réaliser une gestion complète de votre Projet de Fin d'Études (PFE) en utilisant MS Project. Vous devrez planifier, organiser et suivre les différentes phases de votre projet, en impliquant divers acteurs et en utilisant toutes les ressources.

Chaque étudiant doit envoyer par email, avant le **05/05/2024**, à l'adresse suivante : abdelouahed.sabri@usmba.ac.ma un dossier qu'il faut nommer « **VotreNom_VotrePrenom_VotreCNE** », compressé (ZIP ou RAR), et contenant :

1. **Un rapport (compte rendu) Word et pdf**
   - Une description de votre PFE
   - Le cycle de vie adopté pour la réalisation de votre projet
   - Des captures d'écran de votre plan de projet :
     - La liste des taches
     - La Liste des ressources
     - La planification : du démarrage du PFE jusqu'à la soutenance devant le Jury
     - Le diagramme de PERT
     - Le diagramme de GANTT
     - …
2. **Le fichier source de votre projet Ms Project**

Instructions :

- Vous aurez besoin d'au moins trois types d'acteurs: un Développeur, un Testeur, et un Concepteur. Il faut inclure également l'Encadrant comme une ressource
- Ajouter des ressources matérielles (par exemple, ordinateurs, logiciels spécifiques, …) nécessaires pour le projet
- Affecter ces ressources aux tâches correspondantes, en tenant compte de leur disponibilité et de leur capacité.

Remarques :

- Il faut noter que c'est un travail individuel. Toute ressemblance sera sanctionnée
- Pour les étudiants n'ayant pas de PFE cette année, il faut penser à un projet informatique pour la conception et le développement d'une application Web, Mobile, ou de bureau.