

TP PL/SQL

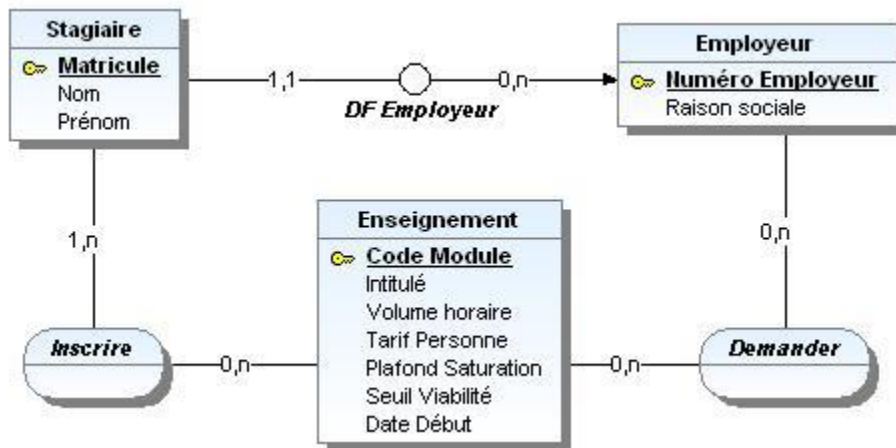
PRÉSENTATION

Il s'agit d'aider un Centre de Formation à développer une application pour la gestion d'inscriptions de stagiaires à des modules d'enseignement par des employeurs.

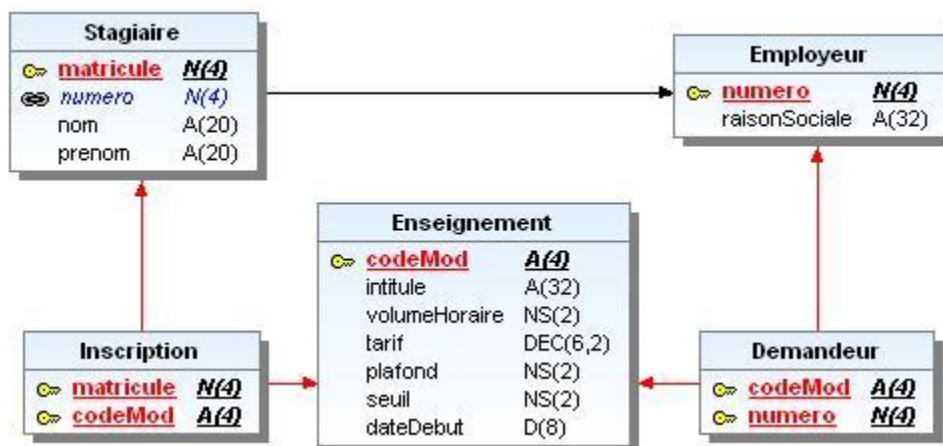
Un module d'enseignement dispensé dans le centre est caractérisé par son code, un intitulé (*De MERISE à UML, Développement WEB*, etc.), son volume horaire, le tarif par personne inscrite, un plafond de saturation et un seuil de viabilité en nombre d'inscrits, et la date de début prévue. Un stagiaire est inscrit à des modules par son employeur : une entreprise cliente décrite par son numéro et sa raison sociale.

Les informations nécessaires à cette activité du centre ont été structurées dans les modèles de données conceptuels (MCD) et logiques (MLD) donnés ci-après.

MCD



MLD



CONTENU DES TABLES

Table Stagiaire

```
SELECT * FROM Stagiaire;
```

MATRICULE	NUMERO	NOM	PRENOM
1	1	Dubois	Paul
2	1	Dupont	Patrick
3	2	Durand	Pierre
4	2	Duteuil	Yves
5	3	Dunois	Luc

Table Employeur

```
SELECT * FROM Employeur;
```

NUMERO	RAISON SOCIALE
1	Oracle
2	IBM
3	Microsoft

Table Inscription

```
SELECT * FROM Inscription;
```

Matricule	Codemod
3	WEB
5	SQL
1	UML
2	UML
4	UML
1	WEB

Table Enseignement

```
SELECT * FROM Enseignement;
```

CODEMOD	INTITULE	VOLUME HORAIRE	TARIF	PLAFOND	SEUIL	DATE DEBUT
UML	De Merise à UML	30	100	6	2	07/05/05
WEB	Développement Web	20	200	4	2	07/06/05
SQL	Langage SQL	40	150	5	3	07/07/05

Table Demandeur

```
SELECT * FROM Demandeur;
```

CODEMOD	NUMERO
WEB	1
WEB	2
UML	1
UML	2
SQL	3

QUESTION 1

Constituer un fichier *question1.sql* contenant un script SQL qui permet de créer la base de données de travail contenant les données indispensables à la gestion des inscriptions des stagiaires à des modules d'enseignement dispensés dans le Centre de Formation. Ce fichier est à constituer à partir des trois fichiers listés ci-dessous, disponibles sur '*Celene*' au sein de la ressource intitulée '*Centre de formation*' qui vous est accessible. Ce fichier doit contenir les définitions des **clés primaires et étrangères en tant que contraintes de tables** ainsi que celles d'autres contraintes qui vous semblent judicieuses. On nommera explicitement toutes les contraintes, sauf 'NOT NULL'.

1. TP2_FP_DROPCREATE.SQL qui contient les ordres de destruction et de création des cinq tables **Demandeur**, **Employeur**, **Enseignement**, **Inscription** et **Stagiaire** **sans aucune définition de contrainte**.
2. TP2_FP_INSERT.SQL qui contient les ordres d'insertion de données dans les cinq tables **Demandeur**, **Employeur**, **Enseignement**, **Inscription** et **Stagiaire**.
3. TP2_FP_SELECT.SQL qui permet de visualiser la structure et le contenu des cinq tables ainsi que les contraintes d'intégrité que vous auriez définies explicitement sur les cinq tables.

QUESTION 2

1. Ecrire une fonction **NbDispo** (dans un fichier *question2_1.sql*) qui prend en entrée un code module et qui renvoie le nombre de places encore disponibles pour des inscriptions de stagiaires à ce module.

Scénario d'utilisation de la fonction :

```
Select NbDispo('UML') from Dual;
NBDISPO('UML')
-----
3
Select NbDispo('WEB') from Dual;
NBDISPO('WEB')
-----
2
```

2. Ecrire une fonction **NbManquant** (dans un fichier *question2_2.sql*) qui prend en entrée un code module et qui renvoie le nombre d'inscriptions manquantes pour qu'un module atteigne son seuil de viabilité.

Scénario d'utilisation de la fonction :

```
Select NbManquant('UML') from Dual;
NBMANQUANT('UML')
-----
0
Select NbManquant('SQL') from Dual;
NBMANQUANT('SQL')
-----
2
```

QUESTION 3

1. Ajouter une colonne **etatMod** de type **VARCHAR2(10)** à la table **Enseignement**, cette colonne ne pouvant prendre comme valeur que l'une des trois possibilités suivantes : 'NON VIABLE', 'VIABLE' ou 'SATURE'.
2. Ecrire et lancer un bloc PL/SQL qui remplit automatiquement la colonne **etatMod** de la table **Enseignement** selon la règle décrite ci-après.
 - **etatMod** vaut 'SATURE' si le nombre d'inscrits au module a atteint son plafond de saturation.
 - **etatMod** vaut 'VIABLE' si le nombre d'inscrits au module est en dessous du plafond de saturation et au moins égal au seuil de viabilité.
 - **etatMod** vaut 'NON VIABLE' si le nombre d'inscrits au module est en dessous du seuil de viabilité.

Il vous est conseillé d'utiliser un curseur dans une forme simplifiée. S'agissant d'une mise à jour par **UPDATE** via un curseur, on utilisera, lors de la déclaration du curseur, la clause **FOR UPDATE** de l'instruction **SELECT** et, lors de la mise à jour, la clause **WHERE CURRENT OF** de l'instruction **UPDATE**. Regrouper les réponses aux deux sous-questions 1 et 2 dans un même fichier *question3.sql*.

Scénario de test :

DESC Enseignement

Nom	NULL ?	Type
CODEMOD	NOT NULL	VARCHAR2 (4)
INTITULE		VARCHAR2 (32)
VOLUMEHORAIRE		NUMBER (2)
TARIF		NUMBER (6,2)
PLAFOND		NUMBER (2)
SEUIL		NUMBER (2)
DATEDEBUT		DATE
ETATMOD		VARCHAR2 (10)

SELECT * FROM Enseignement;

CODE	INTITULE	VOLUMEHORAIRE	TARIF	PLAFOND	SEUIL	DATEDEBU	ETATMOD
UML	De Merise à UML	30	100	6	2	24/05/05	VIABLE
WEB	Développement Web	20	200	4	2	24/06/05	VIABLE
SQL	Langage SQL	40	150	5	3	24/07/05	NON VIABLE

QUESTION 4

Ecrire un trigger lié à la table **Inscription** qui met à jour “automatiquement” l’état du module d’enseignement, dans la table **Enseignement**, à chaque nouvelle inscription.

L’ajout d’une nouvelle inscription peut faire basculer l’état d’un module :

- Soit de l’état **NON VIABLE** à l’état **VIABLE** ;
- Soit de l’état **VIABLE** à l’état **SATURE**.

Dans le cas d’une nouvelle inscription concernant un module déjà saturé, l’inscription doit être refusée. Utiliser la procédure **RAISE_APPLICATION_ERROR** pour traiter ce cas d’erreur.

Ecrire le script PL/SQL de création du TRIGGER dans un fichier *quesiont4.sql*.

Pour vérifier le bon fonctionnement du TRIGGER, on pourra effectuer les opérations suivantes :

```
Insert Into Inscription Values(1,'SQL');
Insert Into Inscription Values(2,'SQL');

Select * From Enseignement;
```

CODE	INTITULE	VOLUMEHORAIRE	TARIF	PLAFOND	SEUIL	DATEDEBU	ETATMOD
UML	De Merise à UML	30	100	6	2	24/05/05	VIABLE
WEB	Développement Web	20	200	4	2	24/06/05	VIABLE
SQL	Langage SQL	40	150	5	3	24/07/05	VIABLE

```
Insert Into Inscription Values(2,'WEB');
Insert Into Inscription Values(4,'WEB');

Select * From Enseignement;
```

CODE	INTITULE	VOLUMEHORAIRE	TARIF	PLAFOND	SEUIL	DATEDEBU	ETATMOD
UML	De Merise à UML	30	100	6	2	24/05/05	VIABLE
WEB	Développement Web	20	200	4	2	24/06/05	SATURE
SQL	Langage SQL	40	150	5	3	24/07/05	VIABLE

```
...
Insert Into Inscription Values(5,'WEB');
...
ORA-20000: Le module de code WEB est déjà saturé.
...
```