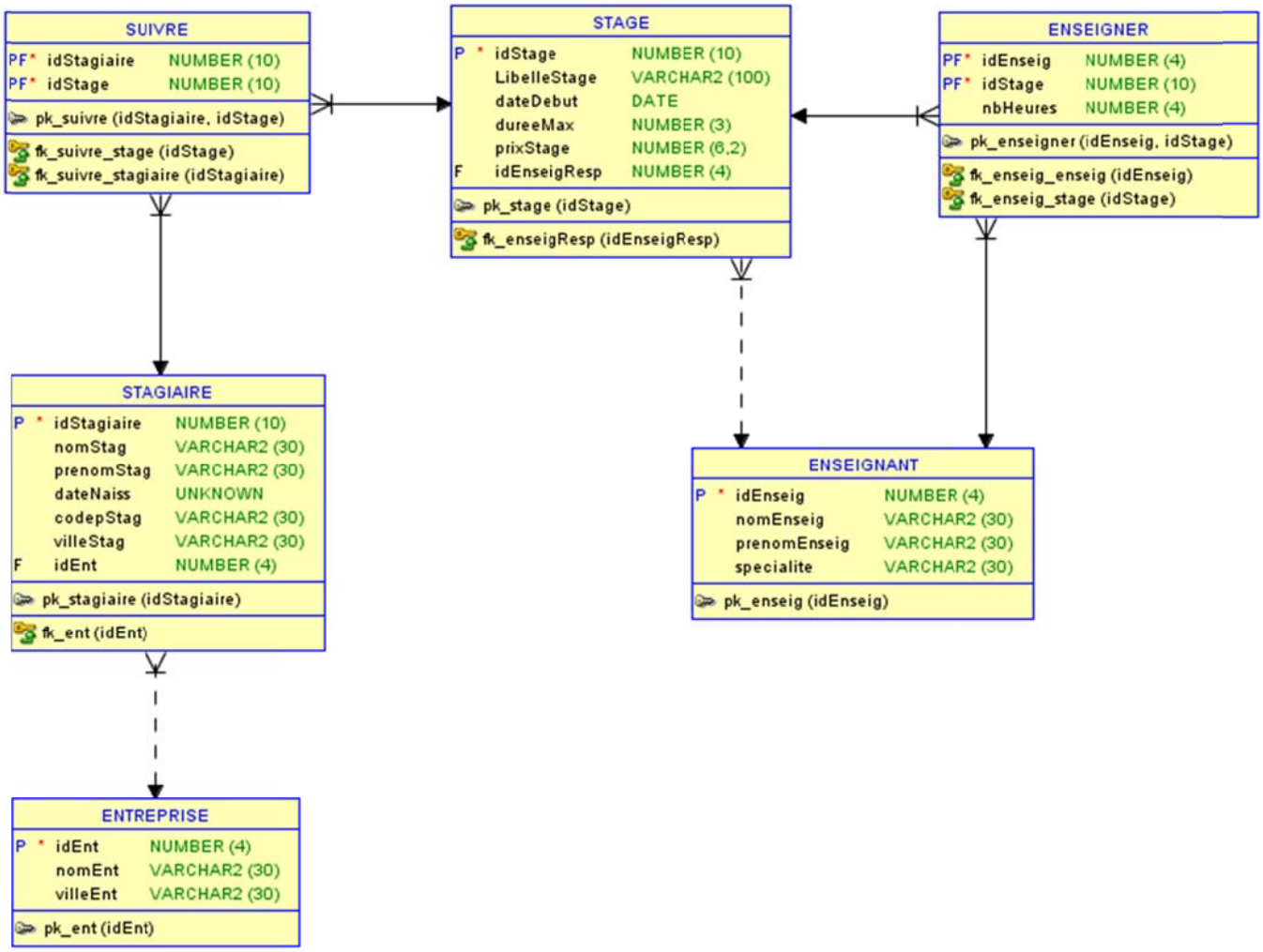
EISTI Ecole Internationale des Sciences du Traitement de l'Information	**EXAMEN DE** **TRAITEMENT ET ADMINISTRATION DES DONNÉES** **Durée : 2 heures** **Tous les documents papiers sont autorisés**
Rédigé par : L'équipe pédagogique	Ref : *SE- TAD*
A l'intention de : Élèves ingénieurs 2ème année	Créé le : 08/12/2016

DCF est une société spécialisée dans l'organisation de stages professionnels pour les entreprises (formation continue). Elle accueille des stagiaires provenant de diverses entreprises qui en assurent le paiement. Plusieurs enseignants encadrent ces stages. Ils peuvent intervenir sur plusieurs stages. Les heures réalisées sont variables et dépendent des stages. Un enseignant responsable est désigné pour chaque stage. Ainsi, un enseignant peut-être responsable de plusieurs stages ou d'aucun.

Ces stages concernent l'informatique (logiciels), mais aussi la communication écrite ou orale, le management, … Un stagiaire peut suivre 1 ou plusieurs stages. On ne le rentre dans la base (et son entreprise) que lorsqu'il réalise effectivement un stage. La durée d'un stage est donnée en jours. Un jour correspond à 6 heures de stage.

La base de données de DCF est réalisée selon le schéma relationnel suivant. Vous trouverez en annexe un script SQL de création de la base. On suppose cette base créée sous Oracle dans le schéma « dcf » précédemment créé. « dcf » possède tous les droits d'administration sur l'ensemble de son schéma.
On suppose enfin que vous êtes connecté en tant que « system ».



I. Administration Base de Données (6 points)

1. Créer deux utilisateurs **scol** et **compta** avec le mot de passe par défaut DCF.
2. Créer les rôles suivants :
    a) **RScol** : accès en lecture aux tables stage, enseigner, enseignant et tous les droits aux tables stagiaire, entreprise et suivre.
    b) **Rcompta** : accès en lecture à toutes les tables.
3. Créer les vues suivantes :
    a) La vue « **administration** » qui donne la liste des stages (tous les champs de la table avec les nom et prénom de l'enseignant responsable), le nombre d'élèves inscrits et le montant total généré.
    b) La vue « **chargetravail** » qui permet de visualiser, par enseignant, la liste des stages pour lesquels il intervient (libellé, date de début et responsable), ainsi que le nombre d'heures correspondantes et le nombre d'élèves inscrits.
    c) La vue « **nowebdev** » qui donne la liste des entreprises (tous les champs) dans la base n'ayant pas de stagiaire qui ait suivi une formation « Développement Web ».

II. PL/SQL (6 points)

1. Pour quelles tables est-il utile de créer une séquence ? Écrire les séquences correspondantes.
2. Écrire une procédure qui édite une fiche récapitulative d'un stage contenant les informations suivantes : libellé, date de début, durée, tarif, nom et prénom de l'enseignant responsable et nombre de stagiaires inscrits.
3. Écrire une fonction qui retourne la charge horaire d'un enseignant. On supposera que la responsabilité d'un stage occupe l'enseignant 2 heures par jour de formation.
4. Écrire une procédure qui affiche pour chaque enseignant son nom, son prénom et sa charge horaire. Cet affichage doit se terminer par celui de la charge horaire moyenne.

III. JDBC (3 points)

Supposons que nous avons à notre disposition une classe Java MyConnectorJDBC pour établir une connexion à la base de données de DCF de la façon suivante :

```
import java.sql.*;
public class Dcf {


  public static void main(String [] args) {
        MyConnectorJDBC connector = new MyConnectorJDBC();

        try {
                MyConnectorJDBC.setParametersOracleLocal("system","password");
                Connection  connexion = connector.getConnection();
                System.out.println("Connexion OK");

                /* À compléter... */

                connexion.close();
                }
                catch(Exception e) {
                        e.printStackTrace();
            }
   } // fin main
} // fin classe Dcf
```

Réalisez les opérations suivantes :

1. Complétez le code de la classe Dcf pour obtenir le nom et le prénom des enseignants spécialistes d'une spécialité passée comme argument du programme exécuté en ligne de commande.
2. Justifiez le type de requête utilisée dans la question 1 et expliquez la différence entre une requête SQL paramétrée et non paramétrée en Java.
3. Nous voulons ajouter au nom de chaque spécialité le suffix '_eisti'. Par exemple, si l'argument du programme est la spécialité 'Informatique', elle sera changée à 'Informatique_eisti' dans la base de données. Ajoutez le code pour mettre à jour les n-tuples concernés et montrez le nombre de n-tuples modifiés.

IV. JAXP (5 points)

Soit le document entreprises.xml suivant :

```
<entreprises>
  <entreprise idEnt="1">
        <nomEnt>Turbomeca</nomEnt>
        <villeEnt>Bordeaux</villeEnt>
  </entreprise>
  <entreprise idEnt="2">
        <nomEnt>Thales</nomEnt>
        <villeEnt>Paris</villeEnt>
  </entreprise>
  <entreprise idEnt="3">
        <nomEnt>Zara</nomEnt>
        <villeEnt>Madrid</villeEnt>
  </entreprise>
  <entreprise idEnt="4">
        <nomEnt>Gucci</nomEnt>
        <villeEnt>Paris</villeEnt>
  </entreprise>
</entreprises>
```

1. Créez une classe Java gestionEntreprise et une fonction chargerXML qui charge un fichier XML comme un arbre DOM. Créez la fonction main() pour lire le fichier entreprises.xml.
2. Ajoutez une fonction mutationSiege() qui, sur l'arbre DOM, change à Pau la ville de toutes les entreprises qui sont actuellement à Bordeaux.
3. Nous avons le fichier entreprises.xsd suivant pour valider notre document xml :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="entreprises ">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="entreprise" minOccurs="10" maxOccurs="unbounded">
         <xs:complexType>
                    <xs:sequence>
                        <xs:element name="nomEnt"/>
                        <xs:element name="villeEnt"/>
                    </xs:sequence>
            <xs:attribute name="idEnt" type="xs:int"/>
         </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Créez une fonction chargerXMLavecXsd() et une classe MyErrorHandler pour lire et valider le document entreprises.xml avec ce schéma. Quelle sera le résultat de la validation pour ce document XML ? Justifiez.

ANNEXES : Script SQL de création de la base de données DCF :

```
CREATE TABLE ENSEIGNANT (
    idEnseig NUMBER(4) NOT NULL CONSTRAINT pk_enseig PRIMARY KEY,
    nomEnseig VARCHAR2(30),
    prenomEnseig VARCHAR2(30),
    specialite VARCHAR2(30)
);

CREATE TABLE STAGE (
    idStage NUMBER(10) NOT NULL CONSTRAINT pk_stage PRIMARY KEY,
    LibelleStage VARCHAR2(100),
    dateDebut DATE,
    dureeMax NUMBER(3),  -- nombre de jours, sachant qu'un jour contient 6 heures de formation
    prixStage NUMBER(6,2),
    idEnseigResp NUMBER(4),
    CONSTRAINT fk_enseigResp FOREIGN KEY(idEnseigResp) REFERENCES ENSEIGNANT(idEnseig)
);

CREATE TABLE ENTREPRISE (
    idEnt NUMBER(4) NOT NULL CONSTRAINT pk_ent PRIMARY KEY,
    nomEnt VARCHAR2(30),
    villeEnt VARCHAR2(30)
);

CREATE TABLE STAGIAIRE (
    idStagiaire NUMBER(10) NOT NULL CONSTRAINT pk_stagiaire PRIMARY KEY,
    nomStag VARCHAR2(30),
    prenomStag VARCHAR2(30),
    dateNaiss DATE,
    rueStag VARCHAR2(30),
    codepStag VARCHAR2(30),
    villeStag VARCHAR2(30),
    idEnt NUMBER(4),
    CONSTRAINT fk_ent FOREIGN KEY(idEnt) REFERENCES ENTREPRISE(idEnt)
);

CREATE TABLE ENSEIGNER (
    idEnseig NUMBER(4) NOT NULL,
    idStage NUMBER(10) NOT NULL,
    nbHeures NUMBER(4),
    CONSTRAINT pk_enseigner PRIMARY KEY (idEnseig, idStage),
    CONSTRAINT fk_enseig_enseig FOREIGN KEY(idEnseig) REFERENCES ENSEIGNANT(idEnseig),
    CONSTRAINT fk_enseig_stage FOREIGN KEY(idStage) REFERENCES STAGE(idStage)
);

CREATE TABLE SUIVRE (
    idStagiaire NUMBER(10) NOT NULL,
    idStage NUMBER(10) NOT NULL,
    CONSTRAINT pk_suivre PRIMARY KEY (idStagiaire, idStage),
    CONSTRAINT fk_suivre_stagiaire FOREIGN KEY(idStagiaire) REFERENCES STAGIAIRE(idStagiaire),
    CONSTRAINT fk_suivre_stage FOREIGN KEY(idStage) REFERENCES STAGE(idStage)
);
```