Travail de conception et d'exploitation d'une base de données
(INF5180 – Hiver 2018)
TP2

Le travail consiste à faire la conception, la construction et la mise en œuvre partielle d'une application de base de données avec le SGBD Oracle. Le travail doit se faire en **équipe** de **deux étudiants (les groupes d'un étudiant seront acceptés exceptionnellement!)**. Les détails sur les environnements de travail Oracle ont été présentés dans vos séances de laboratoire.

À REMETTRE DANS UNE ENVELOPPE FERMÉE (identifiée avec le sigle du cours, le groupe, le nom du professeur et les noms des membres de l'équipe)

- **ET SUR PAPIER**
  1) **Le document INF5180tp2_template.doc dûment rempli**

À REMETTRE SUR MOODLE

- **UN UNIQUE FICHIER ZIP INCLUANT**
  1) **Le document INF5180tp2_template.doc dûment rempli**
  2) **Tous les fichiers Oracle dans les répertoires « script/ » et « script/resultats/ »**
  3) **Le projet Eclipse\Netbeans avec tous les fichiers java et Hibernate sans le pilote ojdbc**.jar**

Le non-respect des spécifications (identification de l'enveloppe et du travail, Template du travail fourni, fichiers à fournir, etc.) du TP2 vous coûtera une pénalité négative de 10 points en moins par violation dans votre note finale.

La non-remise d'un des deux livrables (Document Papier ou Fichier zip sur Moodle) vous coûtera 50% de votre note finale.

La qualité du français constitue un critère d'évaluation (pour un maximum de 10%)

Les travaux remis en retard ne seront pas considérés.

Échéancier

Le 26 avril 2018 avant 16h00 sur Moodle et à la chute de courriers du département d'informatique au 4e étage du PK.

Références

Godin, R. (2006). *Systèmes de gestion de bases de données par l'exemple*. 2ième édition, Montréal, Canada

Voici un texte délimitant la portée du système et rappelant les règles de base du domaine :
Suite à votre phase de conception, une entreprise de consultation a fourni le schéma relationnel ci-dessous permettant de gérer les consultations entre des docteurs identifiés par leurs matricules d'employés (répertoire partagé des employés du centre médical) et des patients identifiés par leur numéro de dossier et associé à un docteur.

DOSSIERPATIENT (**numDos**, nomP, prenomP, genre, numAS, dateNaiss, dateC, matricule)

DOCTEUR (**matricule**, nomM, prenomM, specialite, ville, adresse, niveau, nbrPatients)

CONSULTATION (**CodeDocteur, numDos, dateC**, diagnostic, numOrd)

ORDONNANCE (**numOrd,** recommandations, type, dateC)

ORDONNANCECHIRURGIE (**numOrd, idChir**, rang)

CHIRURGIE (**idChir**, idType, idSalle, dateChirurgie, HeureDebut, HeureFin)

SALLE (**idSalle**, nom)

SPECIALISATIONSALLE (**IdType**, **idSalle,** , dateC)

TYPECHIRGIE (**IdType**, nom, Description)

ORDONNANCEMEDICAMENTS (**numOrd, idMed**, nbBoites)

MEDICAMENT (**idMed**, nomMed, prix, categorie)

SPECIALITE (**code**, titre, description)

CATEGORIES (**IdCategorie**, nom, Description)

Le patient est identifié par son numDos numéro de dossier et son *numAS* numéro d'assurance maladie est unique et son docteur traitant est *matricule*. Le docteur est identifié par *matricule* et il peut avoir une spécialité. Un patient consulte un docteur à une certaine date. Lors d'une consultation, le docteur effectue un diagnostic et lui prescrit une ordonnance dont l'identifiant est *numOrd*. Une ligne de l'ordonnance du docteur '*numOrd'* prévoit de donner un *nbBoites* du médicament *idMed*. Le médicament est identifié, il a un certain prix et correspond à une catégorie de médicament. La date *dateC* représente la date de création. HeureDebut et HeureFin sont des entiers positifs (Example, 900 pour 9h AM et 1300 pour 1h PM).

Attention à bien définir les clés primaires (**Champs en gras**) et étrangères (Champs soulignés).

Voici les autres contraintes portant sur ces tables.

- ✓ Les données *nomP*, *prenomP*, *nomM*, *prenomM*, nomMed, *nom*, *titre*, et *diagnostic* doivent toujours être connues.
- ✓ Toutes les valeurs (*nbBoites*, *prix*, *etc.*) ont une valeur par défaut à 0 et prennent des valeurs positives. Le champ *genre* peut avoir uniquement les valeurs 'F' et 'M'.
- ✓ Il ne peut pas y avoir deux médicaments avec le même nom et la même catégorie.
- ✓ Les niveaux autorisés sont : Étudiant, Interne, ou Docteur et les types autorisés sont : Chirurgie ou Médicaments.
- ✓ Il ne peut pas y avoir deux chirurgies pour une même ordonnance avec le même rang.
- ✓ La suppression d'un docteur doit entraîner la suppression de ses consultations. La suppression d'un patient doit entraîner la suppression de ses consultations. La suppression ou la modification d'une ordonnance ou d'un médicament, référencés respectivement dans CONSULTATION ou ORDONNANCE, ne sont pas autorisées. La suppression d'un docteur doit entraîner la modification de ses patients en donnant la valeur nulle à la matricule.

Des demandes typiques auxquelles la base de données devra pouvoir répondre sont (**exemples de cas d'utilisation – Choix d'un cas par point**) :

- ✓ Afficher le nombre de chirurgies par docteur, afficher le nombre de chirurgies par salle, ou afficher le nombre de chirurgies par type.
- ✓ Afficher le nombre de consultations par docteurs ou afficher le nombre de consultations par spécialité.
- ✓ Afficher le nombre moyen de consultations par mois, afficher le nombre de consultations par année.
- ✓ Afficher le nombre de médicaments prescrits par docteur, afficher le nombre moyen de médicaments prescrits par mois, afficher le nombre de médicaments prescrits par docteur par année.
- ✓ Ratio moyen des chirurgies versus traitements médicamenteux par mois ou ratio des chirurgies versus traitements médicamenteux par années.

Travail demandé :

1. Code du schéma SQL (20pts)

Dans cette partie, vous devez utiliser le schéma relationnel, les exigences décrites dans la description et les contraintes d'intégrité pour produire le schéma en SQL correspondant. Vos tables doivent respecter toutes les contraintes définies dans l'énoncé pour la création de tables.

- Écrire le script SQL*plus pour la création des tables avec les contraintes d'intégrité CHECK, clé primaire et référentielle. Ces contraintes d'intégrité statiques doivent être déclarées au niveau de la table (pas par des *triggers*). Comme indiqué dans l'exemple ci-dessous, toutes ces contraintes doivent être identifiées par un nom.
  Exemple :

```
CREATE TABLE NomTable
(attr1 INTEGER NOT NULL,
CONSTRAINT nomTab_pk PRIMARY KEY(attr1),
CONSTRAINT nomTab_fk FOREIGN KEY(attr1) REFERENCES Table2(attr1),
CONSTRAINT nomTab_chk CHECK( attr1 >= 10)
)
```

- Écrivez un script SQL*plus pour l'insertion de données de test. Ce dernier (jeu de données) doit être assez varié pour pouvoir vérifier toutes les contraintes implémentées. Vous devriez insérer ces données dans la base et puis les utiliser pour vérifier vos contraintes d'intégrité et les requetés des sections 2 et 3.

- Écrivez un script pour la suppression complète de votre schéma. Ce fichier doit inclure les commandes DROP pour la suppression des :
  - ✓ Tables,
  - ✓ Séquences,
  - ✓ etc.

2. Construction de l'application Java avec JDBC (35 pts)

Vous devez réaliser l'interfaçage entre le langage de programmation *Java* et le schéma de votre base de données en utilisant *JDBC***.** À faire :

a) Générez la ou les classes Java correspondantes.
b) Créez le code permettant, sans interface graphique, une insertion de données par table.
c) Implémentez le code affichant le résultat, sans interface graphique, pour réaliser trois (3) cas d'utilisation de la section 4 de votre choix. Vous serez évalué sur la consistance de votre travail et l'optimalité de votre code.

Le code source de l'interface graphique est secondaire et ne sera pas évalué. Vous allez uniquement insérer des lignes du code permettant de faire le cas d'utilisation en passant par JDBC. En conséquence, votre solution utilisera par exemple :

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM patients");
```

Le programme devra prendre les paramètres de connexion comme arguments lors de l'exécution du logiciel dans une console (paramètres à votre main). La sortie devra également être affichée à la console. Exemple :

java -jar baseJDBC.jar url user password

3. Construction de l'application Java avec Hibernate (45 pts)

Vous devez réaliser les correspondances entre la hiérarchie à objets *Java* et le schéma de votre base de données en utilisant *Hibernate***.** À faire :

a) Générer les fichiers Hibernate nécessaires et les classes Java correspondantes pour toutes les tables (fichiers de mapping).
b) Créez le code permettant, sans interface graphique, une insertion de données par table.
c) Implémentez le code du code affichant le résultat, sans interface graphique, pour réaliser trois (3) cas d'utilisation de la section 4 de votre choix. Vous serez évalué sur la consistance de votre travail et l'optimalité de votre code.

Note : Vous pouvez utiliser les mêmes choix que pour la section 2.

Le code source de l'interface graphique est secondaire et ne sera pas évalué. Vous allez uniquement insérer des lignes du code permettant de faire le cas d'utilisation en passant par Hibernate. En conséquence, si votre solution utilise par exemple :

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM patients");
```

Vous allez perdre tous vos points sur cette partie du travail parce que ce n'est pas Hibernate.

Le programme devra prendre les paramètres de connexion comme arguments lors de l'exécution du logiciel dans une console (paramètres à votre main). La sortie devra également être affichée à la console. Exemple :

java -jar baseHibernate.jar url user password

Note : Vous pouvez utiliser deux projets JAVA différents pour les sections 2 et 3.