UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

Devoir # 2 - Automne 2018
IFT 287
Exploitation de base de données relationnelles et OO

Devoir à remettre au plus tard le vendredi 19 octobre 2018 à 23h59.

Ce devoir a pour but de vous faire pratiquer l'analyse et la conception d'un système basé sur une base de données relationnelle. Vous devrez concevoir le diagramme entité-association, le transformer en modèle relationnel ainsi qu'implémenter une solution au problème dans le langage Java en utilisant JDBC et l'architecture trois-tiers.

Ce devoir est à faire en équipe de 2 obligatoirement.

Votre fonction `main` doit se trouver dans la classe `CentreSportif` du package `CentreSportif`.

Il est fortement recommandé de faire les différentes étapes du devoir dans l'ordre présenté.

Tous les fichiers soumis doivent être encodés au format UTF-8.

Une erreur lors de la soumission ou l'incapacité à respecter ces consignes vous fera perdre 25 points.

Pour accélérer la conception et l'implémentation de systèmes complexes utilisant des bases de données relationnelles, il est utile de réfléchir à l'interaction entre les données du système, en plus de la façon dont la base de données doit être construite. Dans ce devoir, vous avez à réaliser les tâches suivantes :

1. Écrire un diagramme entité-association.
2. Transformer le diagramme entité-association en modèle relationnel et écrire le script permettant de générer la base de données associée.
3. Spécifier les classes de votre architecture.
4. Écrire un programme en Java qui utilisera la base de données créée aux étapes précédentes.

Le détail de chacune des parties est donné dans les sections suivantes.

1 Partie 1

Le centre sportif désire embaucher des étudiants pour développer leur nouveau système de gestion des ligues intra-muros. Vous devez donc leur prouver que vous êtes la meilleure personne pour développer leur logiciel. Afin de bien planifier le développement de l'application, vous devez créer le diagramme entité-association de ce problème. Pour ce faire, vous devez vous baser sur la description du problème qui suit.

1.1 Description du problème

Les ligues intra-muros regroupent toutes les activités de compétition du centre sportif. Le centre sportif a donc besoin de pouvoir inscrire des participants à ses activités. Le centre sportif doit pouvoir créer de nouvelles ligues lorsque nécessaire. Chaque ligue contient des équipes formées de participants déjà inscrit au centre sportif. Chaque équipe a un nombre maximum de joueur pouvant être ajouté. Les équipes d'une même ligue s'affrontent dans des compétitions amicales. Il faut donc aussi un moyen de garder les résultats des compétitions entre deux équipes. Le centre sportif aura aussi besoin d'afficher la liste des équipes d'une ligue, ainsi que tous les joueurs d'une équipe.

Voici la liste complète des commandes que votre système doit pouvoir supporter, incluant les paramètres (entre <>).

— `inscrireParticipant <prenom> <nom> <motDePasse> <matricule>`
Cette méthode crée un compte pour un participant.
— `supprimerParticipant <matricule>`
Cette méthode supprime le compte du participant, seulement s'il ne joue dans aucune équipe.
— `ajouterLigue <nomLigue> <nbJoueurMaxParEquipe>`
Permet au gestionnaire du centre sportif de créer une nouvelle ligue de sport.
— `supprimerLigue <nomLigue>`
Supprime la ligue, ainsi que toutes les équipes et les résultats des compétitions.
— `ajouterEquipe <nomLigue> <nomEquipe> <capitaine>`
Permet à un capitaine de créer une équipe
— `ajouterJoueur <nomEquipe> <matricule>`
Permet à un participant de s'inscrire à une équipe.
— `accepterJoueur <nomEquipe> <matricule>`
Accepter l'inscription d'un joueur dans l'équipe.
— `refuserJoueur <nomEquipe> <matricule>`
Refuser l'inscription d'un joueur dans l'équipe.
— `supprimerJoueur <nomEquipe> <matricule>`
Supprime un joueur déjà accepté de l'équipe.
— `afficherEquipes`
Affiche l'ensemble des équipes, par ligues.
— `afficherEquipe <nomEquipe>`
Affiche l'information complète d'une équipe, incluant les résultats de ses parties.
— `afficherLigue <nomLigue>`
Affiche l'ensemble des équipes d'une ligue, avec le nombre de victoires, défaites et nulles.
— `ajouterResultat <nomEquipeA> <scoreEquipeA> <nomEquipeB> <scoreEquipeB>`
Ajoute un résultat entre deux équipes.
— `quitter`

Attention de bien gérer les contraintes sur les commandes. Par exemple, il n'est pas possible de supprimer un participant s'il est dans une équipe. Il faut premièrement l'enlever de l'équipe. De plus, on ne peut supprimer une ligue s'il y a des équipes avec des joueurs actifs dans l'équipe.

2 Partie 2

Afin de répondre aux besoins du centre sportif, vous avez décidé d'utiliser une base de données relationnelle. Vous devez créer un schéma relationnel pour votre base de données, basé sur le diagramme entité-association modélisé précédemment. Vous devez donc produire le schéma relationnel, en plus de créer le script SQL qui permet de construire la base de données dans PostgreSQL. En plus de votre diagramme relationnel, vous devez produire trois fichiers de commandes SQL. Le premier sert à créer votre base de données et doit se nommer `creation.sql`. Le second fichier doit pouvoir détruire toutes vos tables afin de faire le ménage. Ce second fichier doit se nommer `destruction.sql`. Les seules commandes dans ce fichier devraient être des `DROP TABLE`. Finalement, le troisième fichier doit se nommer `affichage.sql` et doit contenir des commandes `SELECT *` pour chacune des tables de votre base de données.

3 Partie 3

Maintenant que vous avez conçu les schémas du système qui gèrera les ligues intra-muros du centre sportif, vous devez implémenter une solution en Java qui respecte l'architecture trois-tiers vue en classe. Vous devez donc créer un programme qui lit des commandes à la console et qui les exécute. Les commandes sont celles définies dans la partie 1. Les commandes doivent modifier la base de données. Vous pouvez utiliser la classe `Connexion` fourni dans les exemples sur le site web du cours. Votre programme doit recevoir en paramètre le nom du serveur (local ou dinf), le nom de la base de données, le nom de l'utilisateur de la base de données et son mot de passe (comme dans les exemples du cours) que vous pouvez demander à l'enseignant. Avant de commencer le code, vous devez écrire la spécification globale de vos classes (tuples, gestionnaires, etc) dans un diagramme de classe (style UML).

4 Soumission

Vous devez soumettre, sur `http://opus.dinf.usherbrooke.ca/` dans le projet TP2, un fichier nommé `TP2.zip` contenant vos trois schémas au format PDF (diagramme entité-association, schéma relationnel et diagramme

de classes), tous vos fichiers de code (.java) ainsi que trois fichiers SQL (*creation.sql*, *destruction.sql* et *affichage.sql*). Afin de soumettre correctement vos fichiers de code, vous devez exporter votre projet. Pour exporter votre projet vous devez cliquer droit sur votre projet et choisir l'option Exporter. Choisissez ensuite l'option Système de fichier. Sur la page suivante, vous devez sélectionner le projet que vous souhaitez exporter, ainsi que l'emplacement où faire la sauvegarde. Une fois votre projet exporté, vous pourrez copier vos fichiers de schéma et vos scripts SQL à la racine du dossier créé (le dossier contenant le dossier `src`). Vous pourrez ensuite compresser votre répertoire racine sous le nom de TP2.zip. Vous pourrez soumettre votre fichier compressé sur *turnin*.

Votre soumission devrait contenir au minimum la description qui suit. Eclipse ajoute plusieurs fichiers, il est donc possible, et normal, que vous ayez plus de fichiers dans votre soumission.

```
TP2.zip
  +--> TP2 (dossier)
       +--> src (dossier)
       |    +--> CentreSportif (dossier)
       |    |    +--> Vos fichiers de code (.java)
       +--> creation.sql
       +--> destruction.sql
       +--> affichage.sql
       +--> Diagramme Entité association, en PDF
       +--> Diagramme relationnel, en PDF
       +--> Diagramme de classe, en PDF
```

Bonne chance !