UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

Devoir # 2 - Autonme 2023
IFT 287
Exploitation de base de données relationnelles et OO

Devoir à remettre au plus tard le vendredi 6 Octobre 2023 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. Il vaut pour 10% de la note finale.

Votre fonction `main` doit se trouver dans la classe `AubergeInn`.

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

Une erreur lors de la soumission vous fera perdre 25 points.

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

Le package contenant votre code devrait se nommer `AubergeInn`.

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. 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 propriétaire du *Bed and Breakfest* **L'auberge-Inn** a lancé un appel d'offre pour développer sa nouvelle application de gestion d'entreprise et c'est vous qui avez obtenu le contrat. 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

L'auberge-Inn est un *Bed and Breakfest* qui offre à ses clients des chambres comportant différentes commodités. L'entreprise est en pleine expansion et il est fort probable que des agrandissements aient lieu. Votre système doit donc pouvoir ajouter de nouvelles chambres et prévoir l'ajout de nouveaux services. Le système de gestion doit aussi pouvoir inscrire de nouveaux clients. Les clients louent des chambres et sont facturés à la nuit. Le prix total dépend des commodités offertes par la chambre. Le propriétaire désire pouvoir obtenir rapidement de l'information claire sur l'état de location de ses chambres, ainsi que sur ses clients.

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

— `ajouterClient <idClient> <prenom> <nom> <age>`
Cette commande ajoute un nouveau client au système.

— `supprimerClient <idClient>`
Cette commande supprime un client s'il n'a pas de réservation en cours.
— `ajouterChambre <idChambre> <nom de la chambre> <type de lit> <prix de base>`
Cette commande ajoute une nouvelle chambre au système.
— `supprimerChambre <idChambre>`
Cette commande supprime une chambre si elle n'est pas réservée et n'a pas de réservation future.
— `ajouterCommodite <idCommodite> <description> <surplus prix>`
Cette commande ajoute un nouveau service offert par l'entreprise.
— `inclureCommodite <idChambre> <idCommodite>`
Cette commande ajoute une commodité à une chambre.
— `enleverCommodite <idChambre> <idCommodite>`
Cette commande enlève une commodité d'une chambre.
— `afficherChambresLibres <dateDebut> <dateFin>`
Cette commande affiche toutes les chambres qui sont disponibles entre ces 2 dates. La date de début est celle d'arrivée, et la date de fin est celle de départ (ex. une chambre libre du 29 au 30 mars est libre la nuit entre le 29 et le 30 mars, et ne l'est pas nécessairement le 30 au soir). L'affichage doit inclure le prix de location de la chambre (prix de base, plus les commodités).
— `afficherClient <idClient>`
Cette commande affiche toutes les informations sur un client, incluant les réservations présentes et passées. Les réservations contiennent le prix total de la réservation, sans les taxes.
— `afficherChambre <idChambre>`
Cette commande affiche les informations sur une chambre, incluant les commodités offertes.
— `reserver <idClient> <idChambre> <dateDebut> <dateFin>`
Cette commande réserve une chambre pour un client de dateDebut (son arrivée) à dateFin (son départ). Par exemple, s'il réserve une chambre du 29 au 30 mars, il la réserve pour 1 nuit, celle entre le 29 et le 30 mars (et non 2 nuits).
— `quitter`
Cette commande quitte l'application.

Vos affichages doivent être lisibles et clairs.

2 Partie 2

Afin de répondre aux besoins de l'entreprise **L'auberge-Inn**, 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 vos tables dans 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 l'entreprise, 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 PostgreSQL. Vous devez vous baser sur le code fourni sur le site web du devoir. Votre programme doit donc 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 et le code fourni).

4 Soumission

Vous devez soumettre, sur http://turnin.dinf.usherbrooke.ca/ dans le projet TP2, un fichier nommé TP2.zip contenant vos deux schémas au format PDF (diagramme entité-association et schéma relationnel). Vous devez de plus inclure 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 dans Eclipse, 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. Sous IntelliJ (ou autre), compressez tout simplement le dossier du projet (et nommez le TP2.zip). Vous pourrez soumettre votre fichier compressé sur *turnin*.

Votre soumission devrait contenir au minimum la description qui suit. Eclipse et IntelliJ (ou autre) ajoutent plusieurs fichiers, il est donc possible, et normal, que vous ayez plus de fichiers dans votre soumission. **Ne soumettez pas le dossier bin (ou out, selon), car il ne contient que des fichiers compilés qui prennent de la place sur le serveur et qui seront régénérés, de toute façon, par le correcteur.**

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

Bonne chance !