INFO-H-303 : Bases de données
Projet : *Villo!*

Professeur : Esteban Zimányi

http://cs.ulb.ac.be/public/teaching/infoh303

Étude de cas

On vous demande de créer une application de gestion de locations pour les vélos *Villo !*[1][2] de la région de Bruxelles Capitale.

Le système est composé d'un ensemble de plus ou moins 180 stations de vélos réparties dans la ville de Bruxelles. Chaque station peut contenir un certain nombre de vélos. Les utilisateurs peuvent prendre un vélo dans une station, l'utiliser à leur gré et le déposer dans n'importe quelle station disponible quand ils ont fini de l'utiliser.

Il y a deux types d'utilisateurs : les abonnés et les temporaires. Les utilisateurs abonnés payent une redevance annuelle. L'abonnement se fait par internet et carte de crédit et les abonnés reçoivent une carte à puce RFID à identifiant unique. Pour prendre un vélo, ces utilisateurs utilisent leur carte RFID. Ils reçoivent également lors de leur enregistrement un numéro d'utilisateur et un mot de passe. Les utilisateurs ne désirant pas s'abonner, comme les touristes, peuvent acheter un ticket valable soit 24 heures, soit 7 jours. Les tickets sont disponibles uniquement dans les stations disposant d'une borne de paiement et doivent être obligatoirement payés par carte de crédit. Ces utilisateurs reçoivent également un numéro d'utilisateur et un mot de passe temporaires. Pour les utilisateur abonnés, on désire retenir leur nom, leur adresse, leur numéro de téléphone et leurs données bancaires. Pour les utilisateurs temporaires, seules les données de la carte de crédit sont gardées.

La tarifs sont les suivants : une carte d'un an revient à 30 euros, un ticket de 7 jours à 7 euros et un ticket de 24 heures à 1,5 euros. Ensuite, la location proprement dite est gratuite pour la première demi-heure. La seconde demi-heure coute 50 centimes, la troisième 1 euro et les suivantes 2 euros. Les demi-heures ne sont pas divisibles. Pour tous les utilisateurs, un dépôt de garantie de 150 euros est demandé. Cette garantie est encaissée uniquement en cas de non respect des conditions générales d'utilisation.

Chaque station est identifiée par un numéro. Pour chaque station, on désire retenir son nom, ses coordonnées GPS, sa capacité en terme de vélos et si elle dispose d'une borne de paiement.

Un vélo est également identifié par un numéro. On enregistre la date de mise en service, le modèle et un état permettant de déterminer si le vélo est en ordre de fonctionnement.

Afin de pouvoir gérer finement ce système (facturation, statistiques, ...), votre application doit garder l'historique de chaque déplacement. Un déplacement est matérialisé par l'heure de prise du vélo, l'heure de dépôt du vélo, l'utilisateur, le vélo, la station de départ et la station d'arrivée.

[1] `http://www.villo.be/`
[2] `http://fr.wikipedia.org/wiki/Villo!`

Format des données

Vous recevrez plusieurs fichiers à importer dans votre application : un fichier concernant les stations, un autre concernant les vélos, un troisième concernant les déplacements et un dernier concernant les utilisateurs. Les trois premiers fichiers sont au format CSV[3]. Le dernier, concernant les utilisateurs est au format XML. Pour les 4 fichiers, toutes les dates respectent le format XML Schema DateTime (YYYY-MM-DDThh :mm :ss)[4]. Vous pouvez considérer que les données fournies sont valides.
Le fichier concernant les stations est au format suivant :

```
numéro;nom;borne de paiement;capacité;coordonnée X;coordonnée Y
0;JUBILE;False;25;4.338338888888889;50.865036111111102
1;SIMONIS;True;25;4.3311668564217003;50.863055654868198
...
```

Le fichier concernant les vélos est au format suivant :

```
numéro;mise en service;modèle;fonctionne
0;2010-03-21T09:00:00;JCDECAUX2010;True
1;2009-01-02T14:39:12;JCDECAUX2009;True
...
```

Le fichier concernant les déplacements est au format suivant :

```
vélo;utilisateur;depart;heure départ;arrivée;heure arrivée
102;87;1;2010-10-19T12:25:42;23;2010-10-19T12:44:21
99;54;65;2010-10-19T18:00:22;None;None
...
```

Le fichier concernant les utilisateurs est au format XML. On y trouve d'abord sous la balise `<subscribers>` la liste des utilisateurs abonnés (annuels) et ensuite dans la balise `<temporaryUsers>` les utilisateurs utilisant un ticket temporaire :

```
<?xml version="1.0" encoding="UTF-8"?>
<users>

    <subscribers>
        <user>
            <userID> ID </userID>
            <RFID> Numéro RFID </RFID>
            <lastname> Nom </lastname>
            <firstname> Prenom </firstname>
            <password> Mot de passe </password>
            <phone> Téléphone <phone>
            <address>
                <city> Ville du domicile </city>
                <cp> Code postal du domicile </cp>
                <street> Rue du domicile </street>
                <number> Numéro du domicile </number>
            </address>
            <subscribeDate> Date d'inscription <subscribeDate>
            <expiryDate> Date d'expiration de l'abonnement <expiryDate>
            <card> Numéro de la carte de banque </card>
        </user>
    ...
    </subscribers>
    <temporaryUsers>
        <user>
            <userID> ID </userID>
            <password> Mot de passe </password>
            <expiryDate>Date d'expiration du ticket<expiryDate>
            <card>numéro de la carte de banque</card>
        </user>
            ...
    </temporaryUsers>
</users>
```

On vous demande créer un script d'insertion de ces données dans votre application.

[3] http://fr.wikipedia.org/wiki/Comma-separated_values
[4] http://www.w3schools.com/Schema/schema_dtypes_date.asp

Déroulement du projet

Première partie

Pour cette partie, on vous demande de modéliser le problème à l'aide du formalisme entité-association et de préciser les contraintes d'intégrité nécessaires. Ces contraintes doivent être exprimées en français et utiliser les mêmes noms d'entités, d'associations ou d'attributs que dans votre modèle conceptuel. Vous pouvez également exprimer et justifier des hypothèses sur votre modèle. Ces hypothèses peuvent résulter par exemple d'ambiguïtés dans l'énoncé du problème.

Déduisez ensuite de ce modèle conceptuel le modèle relationnel correspondant ainsi que ses contraintes. Vos choix de modélisation doivent être justifiés.

Ces modèles doivent être suffisamment riches pour pouvoir servir de support à l'application décrite ci-dessous ainsi que pour pouvoir répondre aux requêtes de la deuxième partie.

Deuxième partie

Création

On vous demande tout d'abord de déduire de votre modèle relationnel un script SQL DDL de création de la base de données et de ses différentes tables ainsi que de créer cette base de données.

Initialisation

On vous demande d'écrire un script permettant d'importer dans votre base de données les fichiers fournis. Ces données devront être présentes dans votre base de données lors de la défense.

Application

Nous vous demandons ensuite de développer une interface graphique pour votre base de données permettant au minimum de réaliser les opérations ci-dessous :

- enregistrer un utilisateur (abonnement)
- consulter les stations et les Villos disponibles
- connecter un utilisateur abonné, une fois connecté, il peut :
    - prendre un Villo dans une station
    - déposer un Villo dans une station
    - consulter les stations et les Villos disponibles
    - consulter l'historique de ses trajets
    - signaler un problème à propos d'un Villo

Votre application devra veiller à ce que la base de données reste cohérente.

Vous pouvez bien sûr ajouter des fonctionnalités à votre application comme par exemple

- gérer les utilisateurs temporaires
- afficher des statistiques de manière graphiques (sur les utilisateurs, les trajets ou les stations)
- présenter les stations sur une carte (Par exemple, à l'aide de Google Maps)
- afficher les trajets d'un utilisateur ou d'un Villo sur une carte
- gérer la facturation mensuelle
- gérer l'altitude des stations et la prendre en compte dans la facturation : si un vélo est déposé dans une station plus basse que la station de départ, le prix standard est augmenté. Dans le cas contraire, le prix est diminué.
- gérer l'internationalisation
- etc.

Ces apports personnels seront valorisés à hauteur de 4 points sur 20.

Requêtes

Nous vous demandons d'écrire en algèbre relationnelle et calcul relationnel tuple les requêtes R1, R2, R3 et R4 ainsi que toutes les requêtes en SQL. Si une requête vous semble imprécise, indiquez dans votre rapport les hypothèses que vous avez faites pour lever ces imprécisions. Attention, pour les requêtes en algèbre relationnelle et en calcul relationnel, vous ne pouvez pas utiliser de fonction d'agrégation.

- R1 : Les utilisateurs habitant Ixelles ayant utilisé un Villo de la station Flagey.
- R2 : Les utilisateurs ayant utilisé Villo au moins 2 fois.
- R3 : Les paires d'utilisateurs ayant fait un trajet identique.
- R4 : Les vélos ayant deux trajets consécutifs disjoints (station de retour du premier trajet différente de la station de départ du suivant).
- R5 : Les utilisateurs, la date d'inscription, le nombre total de trajet effectués, la distance totale parcourue et la distance moyenne parcourue par trajet, classés en fonction de la distance totale parcourue.
- R6 : Les stations avec le nombre total de vélos déposés dans cette station (un même vélo peut-être comptabilisé plusieurs fois) et le nombre d'utilisateurs différents ayant utilisé la station et ce pour toutes les stations ayant été utilisées au moins 10 fois.

Rapports

Pour la première partie, on vous demande les documents suivants :

- un diagramme entité-association modélisant le projet ainsi que ses contraintes
- une traduction relationnelle de ce diagramme et ses contraintes
- vos hypothèses et la justification de vos choix de modélisation

Le formalisme utilisé doit être un de ceux vus au cours ou aux TPs.

Pour la deuxième partie, vous rendrez un rapport contenant :

- les documents de la première partie tenant compte des remarques des assistants
- le script SQL DDL de création de la base de données
- les requêtes demandées
- les explications et justifications de vos choix et hypothèses

Informations pratiques

- Le projet se fera obligatoirement par groupe de deux.
- La première partie, en version papier, devra être rendue aux assistants pour le **24 avril** au TP et être présentée la semaine suivante suivant un horaire à déterminer.
- Le rapport de la seconde partie devra être rendu en version papier lors de votre défense. Une archive contenant tous les codes sources devra être envoyée à l'adresse **Michael.Waumans@ulb.ac.be** pour le **15 mai** à minuit. La défense du projet aura lieu la même semaine suivant un horaire à déterminer. On vous demandera de présenter votre application pendant 10 minutes et de répondre à quelques questions.
- Le projet comptera pour 25% de la note finale du cours.
- Les apports personnels au projet seront valorisés à hauteur de 4 points sur 20. En d'autres mots, un projet parfait sans apport personnel aura une valeur de 16 points sur 20.
- Sauf mention explicite, vous pouvez utiliser les langages et outils de votre choix (MySQL, PostgreSQL, Python, PHP, Java, ...).
- Vous pouvez développer sur votre propre machine et présenter vos projets sur un ordinateur portable que vous apporterez lors de la défense. Au besoin, nous mettrons à votre disposition une base de données MySQL ainsi qu'un espace web PHP sur un serveur de l'Université.

Bon travail !