Bases de données relationnelles : Mise en pratique

Pierre Senellart (pierre.senellart@telecom-paristech.fr)

19 mai 2016

Le but de ce TP est de se familiariser avec l'utilisation d'un système de gestion de base de données (SGBD, en l'occurrence PostgreSQL) pour la réalisation d'une petite application pratique (ici, une application Web simple permettant de suivre les notes des élèves d'une classe).

1 Modélisation

On veut représenter les données suivantes. On dispose d'une liste d'étudiants (avec leurs noms et prénoms) et d'une liste d'examens avec pour chacun leur type (p. ex., écrit, oral, devoir à la maison, projet), leur date, un titre éventuel, un coefficient, et une note maximale (par exemple, sur 20 ou sur 10). Chaque étudiant a, pour chaque examen, une note (éventuellement manquante si l'examen n'a pas encore eu lieu ou si l'étudiant est absent).

1. Dessiner un schéma entité – relation permettant de modéliser ces données.
2. En déduire un schéma relationnel (avec tables, attributs, clefs primaires et étrangères) pour ces données.

2 Construction de la base

On utilisera pour ce TP le serveur de bases de données PostgreSQL accessible avec les paramètres suivants :

Serveur : tiresias.enst.fr

Identifiant : de la forme tp$n$ (tp1, tp2, ..., tp30), tel qu'attribué pendant le TP

Mot de passe : identique à l'identifiant

Base : identique à l'identifiant

Ces paramètres peuvent être utilisés pour se connecter au serveur de bases de données :

en ligne de commande avec le programme de ligne de commande `psql` qui s'utilise comme suit :

```
psql base identifiant -h serveur
```

avec l'outil phpPgAdmin depuis `http://tiresias.enst.fr/phppgadmin`

depuis Python en arguments de la fonction connect du module psycopg2 (voir plus loin)

1. Tester la connexion à la base par les deux premières méthodes. La base est vide pour l'instant.
2. Créer à l'aide de l'interface de votre choix les tables du modèle relationnel.
3. Remplir ces tables en y ajoutant quelques n-uplets d'exemple d'étudiants et d'examens.
4. À l'aide d'un tableur (par exemple Libre Office Calc), créer un ensemble de n-uplets indiquant la note obtenue par chaque étudiant à chaque examen, dans un format approprié pour l'import à la base. Sauvegarder le résultat comme un fichier CSV, puis l'importer dans la base via phpPgAdmin.

3 Requêtes SQL

1. Imaginer et tester (avec phpPgAdmin ou dans l'interface en ligne de commande) des requêtes SQL pour récupérer les informations suivantes :
   a) L'ensemble des informations (titre, date, coefficient, note maximale) sur tous les examens.
   b) L'ensemble des informations sur un examen d'un identifiant donné.
   c) L'ensemble des notes de tous les étudiants à un examen donné (avec, pour chaque étudiant, son prénom et son nom).
   d) La moyenne pondérée obtenu par tous les étudiants à l'ensemble des examens, normalisée en une note sur 20.
2. Pour chacune de ces requêtes, préfixer la commande `SELECT` du mot-clef `EXPLAIN` et analyser le résultat : cela affiche le plan d'exécution correspondant.

4 Application Web

Nous allons maintenant développer une application Web écrite dans le langage de programmation Python, le framework Web Flask et le moteur de modèles Jinja.

Afin de disposer des bibliothèques nécessaires, lancer la commande :

```
source /infres/ic2/senellar/dsi/virtualenv/bin/activate
```

dans les terminaux qui serviront à exécuter du code Python.

1. Télécharger depuis `http://pierre.senellart.com/enseignement/2015-2016/liesse/tp.zip` une archive avec le squelette de l'application à développer.
2. Décompresser cette archive dans un répertoire de votre compte et ouvrir un terminal dans ce même répertoire.
3. Éditer le fichier model.py pour remplacer toutes les occurrences de *tp1* par l'identifiant qui vous a été attribué.
4. Lancer l'application Web en tapant `python run.py` en ligne de commande. La tester en accédant dans un navigateur à l'URL `http://127.0.0.1:5000/`
5. Prendre le temps de vous familiariser avec l'application en étudiant les fichiers Python et les modèles Jinja utilisés ; inutile de modifier quoi que ce soit à cette étape.
6. L'application actuelle ne fait rien d'utile ; il faut modifier le fichier `model.py` en y mettant à la place des ordres `select 1 where false` (qui renvoient toujours faux) les ordres SQL que vous avez construits à l'étape précédente, à l'endroit approprié. Modifier également les fichiers `templates/exams.html` et `templates/grades.html` pour adapter les noms des attributs à ceux que vous avez utilisés dans vos tables et requêtes. Tester au fur et à mesure que l'application fonctionne.

5 Pour aller plus loin

Au choix :

— Ajouter des fonctionnalités supplémentaires à l'application : ajout d'étudiants, modification d'une note, suppression d'un examen, etc.
— Rendre l'application plus agréable à utiliser grâce à des règles de style CSS.
— Ajouter au modèle de données la possibilité de gérer plusieurs classes, plusieurs trimestres, plusieurs années, etc.
— Lire le tutoriel de l'ORM Python Storm `https://storm.canonical.com/Tutorial` et refaire tout le TP en utilisant cet ORM.