Formations en Informatique de Lille – 2014/2015

TD Technologies du Web

mars 2015

Bases de données - SQL - PHP - semaine 9

Exercice 1 : Chaque commune française possède un identifiant unique : son code Insee sous la forme de 5 caractères, uniquement des chiffres ou des lettres majuscules (ex : 59009 pour Villeneuve d'Ascq).
L'administration française dispose d'un répertoire général des voies (« FANTOIR ») et associe à chacune un identifiant de 10 caractères composé du code commune (5 car.) suivi d'un numéro de 4 chiffres pour la voie au sein de la commune + 1 code de contrôle d'1 caractère (nous n'évoquerons plus ce dernier qui n'a aucun intérêt dans notre exercice). Ex : `590095950` pour l'avenue Paul Langevin à Villeneuve d'Ascq, ce qui se décompose en `59009 5950 0`
Dans la suite on parlera donc de « code Commune » pour désigner une suite de 5 chiffres ou majuscules et de de « code Voie » (ou code Rue) pour une suite de 4 chiffres.
Il faut donc retenir que le couple code Commune / code Voie constitue un identifiant unique pour les voies.
Nous supposerons disposer d'une base de données consacrée à des informations sur des voies publiques. Cette base se compose des tables suivantes

- table **communes**
  - `codeCommune`
  - `nomCommune`
- table **voies**
  - `codeCommune` (chaîne de longueur 5)
  - `codeVoie` (chaîne de longueur 4)
  - `nomVoie` (chaîne)
- table **adresses** : contient une entrée pour chaque adresse de la voie. Le « numéro » est en fait une chaîne et peut comporter des lettres (ex : 27B). Un même numéro n'apparaît jamais plusieurs fois dans la même voie. Suivent les coordonnées géographiques et la distance en mètres séparant l'adresse et l'origine de la voie :
  - `codeCommune` (chaîne de longueur 5)
  - `codeVoie` (chaîne de longueur 4)
  - `numero` (chaîne)
  - `latitude` (nombre flottant)
  - `longitude` (nombre flottant)

Des tables respectant ce schéma vous sont fournies. Importez-les dans votre base Postgresql.
Note : l'importation de certaines tables un peu volumineuses pose problème via l'interface phppgadmin. La meilleure méthode pour importer est d'ouvrir un terminal et d'y utiliser la commande psql comme ceci (exemple pour le fichier d'import adresses.sql) :

```
psql -h webtp -U votre_login -q -f adresses.sql
```

Votre mot de passe pour le serveur de base de données vous sera demandé

Question 1.1 : Pour chaque table, proposez un attribut ou un ensemble d'attributs pouvant constituer une clé primaire. Créez ces clés primaires dans votre base (vous pouvez utiliser l'interface

phppgadmin, onglet «contraintes»)

Question 1.2 : Proposez (et créez dans la base) des «clés étrangères» pour certaines de ces tables de manière à assurer l'intégrité référentielle de la base. Pour ajouter les contraintes «clés étrangères» vous pouvez, là aussi, utiliser l'interface `phppgadmin`.

Question 1.3 : Indiquez les commandes SQL permettant de recueillir les résultats suivants. Entre crochets, sont précisées les informations qui doivent apparaître dans la réponse à la requête SQL. Vous essaierez chaque requête via l'interface `phppgadmin` (conservez dans un fichier une copie des requêtes).

1. La liste des voies de la commune '59009' [codeVoie, nomVoie]
2. La liste des adresses de la voie de code '2120' dans la commune '59009' [codeVoie, numero]
3. La liste des adresses de la voie de code '2120' dans la commune '59009' [nomVoie, numero]
4. Nombre d'adresses de la voie de code '2120' dans la commune '59009' [nombre]
5. Liste des voies de la commune '59009' avec pour chacune son nombre d'adresses [codeVoie, nombre]
6. La liste des voies de Lille [codeVoie, nomVoie] (NB : vous n'êtes pas censé connaître le code Insee de Lille, mais il est supposé être dans la base de données !)
7. La liste des communes où existe une «Avenue Paul Langevin» [nom de commune]
8. La liste des communes où existe un numéro 12 dans l'«Avenue Paul Langevin» [nom de commune]

Exercice 2 :

Question 2.1 : On souhaite construire un site permettant de connaître la liste des adresses d'une commune, par rue. Pour cela on veut présenter un formulaire permettant de choisir une rue, parmi l'ensemble des **noms** de rues existantes dans la commune ; il contiendra bien évidemment un bouton de validation. La validation du formulaire par l'utilisateur a pour effet de charger une page en lui envoyant en paramètre (mode GET)

— le **code** de la rue sélectionnée dans une variable nommée `cVoie`

— le **code** de la commune concernée dans une variable nommée `cCommune`. Cette variable est envoyée par le formulaire mais elle ne doit pas être visible de l'utilisateur puisqu'il n'a pas à la choisir.

Écrire une fonction PHP `formVoies($cc, $url)` . Son **résultat** est une chaîne contenant le code HTML d'un formulaire conforme à la description ci-dessus. Le premier paramètre est celui de la commune concernée et le deuxième est l'URL de la page à charger lors de la validation du formulaire. Les deux paramètres sont supposés avoir une forme valide.

Question 2.2 : Écrire une fonction PHP `listeAdresses($cc, $cv)` dont les paramètres sont un code de commune et un code de voie. Le résultat (chaine) est le code HTML de la liste de toutes les adresses disponibles dans cette voie.

Question 2.3 : Écrire un script `reponseAdresses.php` qui reçoit en paramètres HTTP (mode GET) des variables `cCommune` et `cVoie` (comme celles qui pourraient lui être envoyées par le formulaire de la question 2). Ce script produit une page HTML valide contenant une liste `ul` des adresses de cette voie.
Les paramètres HTTP reçus ne peuvent pas être considérés comme « sûrs ». Afin de sécuriser le site, il convient donc d'en contrôler la forme (voir introduction de l'exercice 1) et de produire une erreur si cette forme n'est pas correcte.