Formations en Informatique de Lille – 2016/2017

TD Technologies du Web

février 2017

Bases de données - SQL

Pour ces exercices, vous manipulerez votre base sur le serveur webtp par l'intermédiaire d'une interface web appelée **phppgadmin**. Dans un navigateur, rendez-vous sur la page principale du serveur webtp puis cliquer sur «Accès à votre base PostgreSQL» (lien situé vers le bas de la page d'accueil). Après authentification (cliquer à gauche sur «postgreSQL»), vous êtes connecté à l'interface.

Exercice 1 : Une base de données est consacrée au tour de France cycliste. Elle comporte pour l'instant deux tables :

- **coureurs** : y sont définis le numéro de **dossard** d'un coureur (entier), son **nom** (chaîne), son **equipe** (chaîne), sa **taille** en centimètre (entier). Au plus un coureur porte un numéro de dossard donné. L'homonymie est par contre possible.
- **equipes** : le **nom**, la **couleur** dominante des maillots et le nom du **directeur** sportif (chaînes). Au plus une équipe porte un nom donné.

Pour commencer, il vous faudra importer ces deux tables dans votre base Postgres. Les commandes de création et de peuplement des tables figurent dans un fichier `tables.sql` que vous sauverez dans l'un de vos répertoires. Visualisez le contenu du fichier . Vous constaterez qu'il contient les commandes permettant de créer 2 tables et de mettre quelques valeurs dans ces tables.

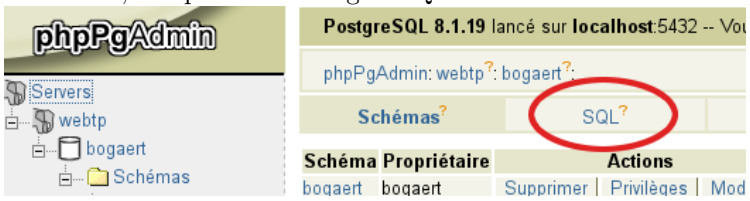
- **Méthode 1** (recommandée) :
  Ouvrez un terminal et placez-vous dans le dossier contenant le fichier `tables.sql`. Exécutez ensuite la commande :

  ```
  psql -h webtp.fil.univ-lille1.fr -U votre_login -f tables.sql
  ```

  Votre mot de passe vous sera demandé, puis vous verrez les résultats de l'exécution des commandes SQL.
- **Méthode 2** (via l'interface graphique)
  - Tout d'abord, vous allez déplier le menu arborescent de la colonne gauche, pour faire apparaître l'item « Schémas ».

  - Cliquez ensuite sur cet item « Schémas ». Dans la partie droite apparaît un menu horizontal, comportant un onglet SQL.

  - Cliquez sur **cet onglet « SQL »** (et pas sur le lien "SQL" qui se trouve en haut à droite de la fenêtre !)

— Puis « importez un script SQL » en choisissant le fichier `tables.sql`.

— Lancez l'exécution.

Quelle que soit la méthode choisie, vous pouvez constater via l'interface graphique la création de ces 2 tables en cliquant sur l'item « Tables » du schéma qui porte votre nom :

Question 1.1 :
Dans cette question, vous allez tester des commandes SQL en les saisissant dans l'interface phpPgAdmin. ***Vous conserverez dans un fichier la copie de toutes les commandes SQL qui constituent une réponse correcte***
Écrire et tester les requêtes SQL permettant d'obtenir :

a - La liste de tous les coureurs avec pour chacun son numéro de dossard et son nom

b - La liste de tous les coureurs avec pour chacun son numéro de dossard et son nom, classés par numéro de dossard (croissant).

c - La liste de tous les coureurs avec pour chacun son numéro de dossard et son nom, classés par équipe et, au sein d'une même équipe, par nom

d - La liste de tous les coureurs du plus petit au plus grand avec pour chacun son numéro de dossard son nom et sa taille

e - Les noms et dossards des coureurs de l'équipe `'LavePlusBlanc'`. Remarquez que vous devrez utiliser comme délimiteur des chaîne les simples quote (`'`). Vous pouvez vérifier par test que les doubles quotes (`"`) ne sont pas admises dans ce rôle.

f - Les doubles quotes peuvent servir à délimiter les noms d'attributs. Reprenez la même requête en encadrant les noms d'attributs par des double quotes. Vérifiez également que l'on peut toujours préfixer le nom d'un attribut par celui de la table (ex `coureurs.dossard` ou `coureurs."dossard"`)

g - Le nom, la taille et l'équipe des coureurs de moins de 1,80m

h - Assurez-vous d'obtenir le même résultat mais avec les coureurs classés par taille croissante.

i - La liste des couleurs des équipes.

Question 1.2 : Cette page de documentation Postgresql référence l'ensemble des fonctions disponibles avec Postgres. Nous allons tout d'abord nous intéresser aux fonctions sur les chaînes (ouvrez dans un navigateur la documentation correspondante).

a - En utilisant l'opérateur de concaténation, obtenez une liste contenant pour chaque coureur une chaîne sous la forme :

```
alain appartient à l'équipe LavePlusBlanc
alphonse appartient à l'équipe PicsouBank
...
```

b - Remarquez que le nom de la colonne (attribut) obtenu n'est pas très satisfaisant. Modifiez la requête SQL de manière à ce que la colonne s'appelle `"appartenance"`

c - Obtenez pour chaque coureur son nom écrit en majuscules ainsi que la longueur de son nom. La table obtenue devra avoir 2 attributs : l'un nommé `"nom maj"` et l'autre `"lg"`

d - Faites de même en assurant un classement par longueurs croissantes des noms (faites une version sans utiliser le nommage de la colonne et une autre en l'utilisant)

e - Obtenez pour chaque coureur son dossard, son nom avec initiale majuscule et les 3 premières lettres de son équipe en majuscules

Question 1.3 : Ouvrez la documentation Postgres concernant le **Pattern Matching**. La fonction de pattern matching la plus utilisée en bases de données est `LIKE` (et sa variante `ILIKE`). Elle repose sur 2 caractères «joker» : `_` et `%`.

a - En utilisant like, obtenez les noms des coureurs commençant par la lettre 'a'

b - En utilisant like, obtenez les noms des coureurs contenant la chaîne 'er'

c - En utilisant like, obtenez les noms des coureurs comportant 5 lettres exactement.

d - En utilisant like, obtenez les noms des coureurs comportant un a suivi de deux lettres exactement.

e - En utilisant like, obtenez les noms des coureurs comportant un a suivi de deux lettres au moins.

Question 1.4 : Dans la table coureurs, les tailles sont données en centimètres. Nous souhaitons les obtenir en mètres, avec 2 chiffres après la virgule. Cet exemple permettra d'utiliser des opérateurs numériques, ainsi que le «cast»

a - Affichez les tailles des coureurs divisées par `100`. Expliquez le résultat (qui ne correspond pas à notre attente)

b - Recommencez en divisant par `100.0`. Ce n'est pas encore le résultat souhaité : pourquoi ?

c - Pour convertir une valeur numérique dans un autre type numérique compatible, on peut utiliser la commande `cast` dont la syntaxe est `cast (valeur as type)` Mettez cela en pratique pour obtenir la taille sous la forme souhaitée en début de question (consultez les types disponibles sur vos notes de cours)

Question 1.5 : Produit cartésien et jointure.

a - Pour obtenir en SQL le produit cartésien de plusieurs tables, il faut citer leurs noms en les séparant par une virgule. Par exemple :

```
select * from table1,table2
```

donnera le produit de table1 par table2.

Obtenez le produit cartésien des tables `coureurs` et `equipes`. Constatez que le résultat contient deux attributs `nom`, issus des 2 tables d'origine. Pour les distinguer il faudra utiliser les noms qualifiés `coureurs.nom` et `equipes.nom`

b - Dans le résultat obtenu, certaines lignes ne présentent pas d'intérêt : ce sont les lignes qui combinent les informations sur un coureur et des informations sur une autre équipe que la sienne. En d'autres termes les lignes pour lesquelles la valeur de l'attribut `coureurs.equipe` est différente de celle de l'attribut `equipes.nom` ne nous intéressent pas. Vous allez utiliser la clause `where` pour ne conserver que les lignes où ces deux valeurs sont égales.

L'opération que vous venez de réaliser s'appelle une **jointure**

c - Obtenez la liste des coureurs avec pour chacun son dossard, son nom, celui de son équipe et la couleur de son équipe.

d - Obtenez la liste des coureurs avec pour chacun son nom et celui de son directeur sportif.

e - Obtenez les noms et dossards des coureurs dont le directeur sportif est Ralph

f - Le nom du directeur sportif du coureur 'alphonse'

Question 1.6 : Ajout de nouveaux tuples dans une table
Choisissez tout d'abord des valeurs d'attributs pour une nouvelle équipe (nom, couleur, directeur). Une première façon d'ajouter un tuple dans la table des équipes consisterait à passer par les formulaires que propose phpPgAdmin (vous pouvez faire l'essai, à condition d'effacer ensuite le tuple que vous venez d'ajouter).
Nous allons maintenant réaliser cette opération d'ajout via une requête SQL. Pour ajouter un tuple dans une table, la commande SQL s'appelle insert et sa forme la plus simple est

insert into ma_table values (v_1, v_2, v_3...)

les valeurs v_i sont les valeurs associées aux différents attributs, **dans l'ordre dans lequel ils ont été définis**. Par exemple :
`insert into equipes values ('Nouvelle Équipe','orange','Archibald')`
On peut aussi choisir l'ordre des attributs, par exemple :
`insert into equipes (couleur,directeur,nom) values ('orange','Archibald','Nouvelle Équipe')`

a - Insérez votre nouvelle équipe dans la table des équipes par une requête SQL.

b - Ajoutez (toujours en SQL) au moins 2 nouveaux coureurs appartenant à cette équipe dans la table des coureurs

Question 1.7 : Absence de valeur
L'absence de valeur dans une colonne est, dans le cas général, possible. Dans ce cas le marqueur `NULL` figure dans la colonne concernée.
Exécutez la requête : `insert into equipes (couleur,nom) values ('orange','Nouvelle Équipe')`
Puis consultez le contenu de la table.
Pour tester si une valeur est absente, il existe un prédicat spécial appelé `is null` (ne pas utiliser le test d'égalité =NULL, cela ne fonctionne pas). Il existe aussi `is not null`.

a - Obtenez par une requête SQL toutes les équipes dont le directeur sportif n'a pas été défini

b - Obtenez par une requête SQL toutes les équipes dont le directeur sportif a été défini

NB : nous verrons plus tard qu'il est possible, dans la définition de la table, d'interdire l'absence de valeur.

Question 1.8 : Modification de valeur La modification d'une valeur d'un tuple passe par la commande `update`. Sa forme la plus simple est

update ma_table set attr = nouvelle_valeur where qualification

La partie where est similaire à celle de la commande select et elle permet de sélectionner les lignes qui vont être modifiées (en l'absence de where, toutes les lignes sont modifiées). La nouvelle valeur est une expression dans laquelle peut intervenir l'ancienne valeur de l'attribut.
Quelques exemples (ne pas les exécuter ! !)

```
-- attribuer le dossard 5 à tous les coureurs (gag)
update coureurs
        set dossard = 5
;
-- attribuer le dossard 1000 à tous les coureurs de l'équipe 'PicsouBank'
update coureurs
        set dossard = 1000
        where equipe = 'PicsouBank'
;
-- ajouter «L'immense » devant le nom de chaque coureur
update coureurs
```

```
        set nom = 'L''immense ' || nom
;
```

a - Erreur à la visite médicale : les tailles des coureurs de l'équipe PicsouBank ont été majorées de 1cm. Rectifiez la base de données pour enlever ce centimètre en trop.

b - Définir un nom de directeur sportif pour l'équipe qui n'en a pas.

Question 1.9 : Création de table
En utilisant la commande `create table` (voir notes de cours), créez une nouvelle table nommée `"etape"` qui contiendra les heures d'arrivée des coureurs à une étape de la course. Cette table possèdera 2 attributs :

— `dossard` de type `integer`

— `arrivee` de type `time`

Le type `time` représente une heure. Elle peut être représentée sous forme de chaîne, par exemple `'17:32:30'` pour 17h 32mn 30s. Pour assurer sa conversion en type time, cette chaîne peut être précédée du mot `time`, mais la conversion est automatique quand le type de la colonne le nécessite. Par exemple on pourra écrire
`update etape set arrivee = time '17:32:30' where dossard=13 --typage explicite`
ou
`update etape set arrivee = '17:32:30' where dossard=13 -- conversion auto`
Garnissez la table `etape` avec des heures d'arrivée pour au moins une partie des coureurs

Question 1.10 : Nouvelles jointures
Vous allez maintenant combiner des informations issues de 2 ou 3 tables.

a - Obtenir la liste des coureurs classée par heure d'arrivée, avec les informations suivantes : dossard, nom du coureur, heure d'arrivée. Vous aurez donc besoin de croiser 2 tables et de trouver la bonnne condition «de jointure» (voir question plus haut) pour ne conserver que les lignes intéressantes.

b - Idem mais en limitant la liste aux courreurs arrivés avant une certaine heure que vous choisirez (en fonction des données que vous avez insérées dans la table `etape`).

c - Obtenir la liste des coureurs classée par heure d'arrivée, avec les informations suivantes : dossard, nom du coureur, nom de l'équipe, couleur de l'équipe, heure d'arrivée. Vous aurez besoin cette fois de croiser 3 tables, avec 2 conditions de jointure.

Question 1.11 : Réaliser un script PHP dont le résultat est une page HTML faisant apparaître la liste des coureurs avec, pour chacun, ses nom, prénom, nom d'équipe et nom du directeur sportif.

Question 1.12 : Réaliser une interface web qui permette à un utilisateur de saisir le nom d'une équipe puis affiche d'une part le nom du directeur sportif et la couleur du maillot et d'autre part la liste des coureurs de l'équipe.
Vous prévoirez l'affichage d'un message d'erreur si le nom fourni n'est pas celui d'une équipe.

Question 1.13 : Avant de réaliser la question, vous allez d'abord améliorer la définition de la table etape. Celle-ci a été définie sans clé primaire et sans contrainte d'intégrité référentielle.

1. Chaque dossard ne doit apparaître qu'au plus une fois dans la table. Cet attribut peut donc constituer une clé primaire, vous allez ajouter cette caractéristique grâce à la commande SQL

```
alter table etape
 add primary key (dossard);
```

2. Pour que notre base soit cohérente, chaque `dossard` figurant dans la table `etape` devrait aussi figurer dans la table `coureurs`. C'est une contrainte dite d'« intégrité référentielle » que le SGBD peut contrôler. On l'ajoute à la table par :

   ```
   alter table etape
    add foreign key (dossard) references coureurs(dossard);
   ```

   À partir de ce moment, à chaque tentative d'insertion d'un numéro de dossard dans la table `etape`, le SGBD verifiera qu'il existe bien dans la table `coureurs` et produira une erreur dans le cas contraire.
   Note 1 : si votre table `etape` contient déjà des données incompatibles avec cette contrainte, il vous faut d'abord les effacer.
   Note 2 : une telle contrainte est définie depuis le départ dans les tables qui vous étaient fournies : l'attribut `equipe` de la table `coureurs` doit faire référence à un `nom` de la table `equipes`. Consultez le source du fichier `tables.sql` pour le vérifier.

Réaliser une interface web permettant la saisie des temps d'arrivée. Cette interface proposera de saisir un numéro de dossard et un temps. À la validation, ces données seront enregistrées dans la base. Vous prévoirez un affichage dans les cas cette opération est incorrecte.