Université de Lille — FACULTÉ DES SCIENCES ET TECHNOLOGIES — Département Informatique

Formations en Informatique de Lille – 2021/2022

Bases de Données (BDD1)

septembre 2021

sujet n°1 (semaines 1 et 2)

Pour ces exercices, vous manipulerez votre base sur le serveur webtp par l'intermédiaire d'une interface web appelée **Phppgadmin**. Dans un navigateur, ouvrez le lien

`http://webtp.fil.univ-lille1.fr/phppgadmin/`

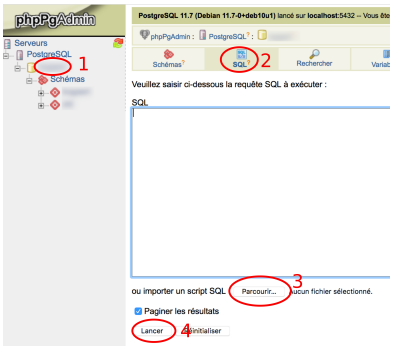
puis cliquez à gauche sur « postgreSQL » pour vous connecter.
Phppgadmin est un client du SGBD qui permet de visualiser les tables et leurs contenus et de réaliser de nombreuses opérations de façon « graphique ». Il permet aussi d'exécuter les requêtes SQL de son choix.

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 le fichier `tables.sql`

- enregistrez `tables.sql` sur votre ordinateur.
- visualisez le contenu du fichier pour constater qu'il contient les commandes permettant de créer 2 tables et de mettre quelques valeurs dans ces tables.
- dans phppgadmin :
  1. cliquer sur votre nom dans la colonne de gauche (voir image ci-dessous)
  2. cliquer sur "SQL"
  3. sélectionnez le fichier tables.sql
  4. lancez l'exécution



- vous pourrez constater la création de ces 2 tables en cliquant sur l'item « Tables » du schéma qui porte votre nom :

Q 1 .
Vous allez tester des commandes SQL en les saisissant dans l'interface phpPgAdmin (le lien « SQL » en haut à droite ouvre une fenêtre de saisie). ***Vous conserverez dans un fichier la copie de toutes vos commandes SQL qui constituent une réponse correcte aux différentes questions.***
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 équipe, 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 qu'il faut utiliser les simples quotes (`'`) comme délimiteurs de chaîne. Les doubles quotes (`"`) ne sont pas admises car elles ont une autre rôle (voir ci-dessous).

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 - Obtenez les noms, tailles et équipes 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.

Q 2 . Cette page de documentation Postgresql (lien cliquable) 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

Q 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.

Q 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)

d - Le même résultat aurait aussi pu être obtenu sans `cast` mais en utilisant la fonction mathématique `trunc`. Lisez sa spécification dans la documentation et écrivez une nouvelle version de la requête.

Q 5 . Jointures.
L'objet de cette question est obtenir des informations en combinant par jointure les tables `coureurs` et `equipes`.

a - Quelle condition de jointure est-elle pertinente pour obtenir des informations cohérentes provenant des 2 tables ?

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

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

d - Obtenez les noms et dossards des coureurs dont le directeur sportif est `Ralph`. Attention, pour cette question et pour la suivante vous distinguerez bien ce qui relève de la qualification (à exprimer par un « where ») et non de la condition de jointure.

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

Q 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 (ce sera donc la première requête SQL modifiant les données (tuples) que nous rencontrons) .
Pour ajouter un tuple dans une table, la commande SQL s'appelle **insert into** 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 spécifier un ordre différent pour les attributs et leurs valeurs, comme ceci :
```
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

Q 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.

Q 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
;
```

Questions :

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.