IBD
07/10/2015

TP PostgreSQL n° 1

– Prise en main –

1. SQL (Structured Query Language)

- Langage de Définition de Données (LDD) : création et modification de la structure des tables de la base de données.
- Langage de Manipulation de Données (LMD) : insertion et modification de données dans la base de données.
- Langage de Contrôle de Données (LCD) : gestion de la sécurité, confidentialités et contraintes d'intégrité.

Comparaison entre le modèle relation et SQL :

Modèle relationnel	**SQL**
Relation	Table
Attribut	Colonne
Tuple	Ligne

2°) PostgreSQL et psql

Connectez-vous à la base de données tp_intro que vous venez de créer.

Commandes de base. Les principales commandes de psql sont les suivantes :
– \? : Liste des commandes psql,
– \h : liste des instructions SQL,
– \h <une_instruction> : description de l'instruction SQL <une_instruction>. Essayez avec l'instruction CREATE
TABLE,
– \d : liste des tables (ou relations, cf. partie 2),
– \d <nom_table> : description de la table <nom_table>,
– \i <chemin/nom_fichier_script.sql> : exécution d'un fichier de script SQL,
– \o <chemin/nom_fichier_resultat.sql> : écriture des résultats des instructions dans le fichier passé en paramètre,
– \o : retour à un affichage à l'écran,.
– \q : quitter psql.
Retenez ces commandes, elles seront indispensables pour les TP.

3. SQL LDD

a) Types syntaxiques :

VARCHAR2(n) : chaîne de caractères de longueur variable (maximum n)
CHAR(n) : chaîne de caractères de longueur fixe (n caractères)
NUMBER : Nombre entier (40 chiffres maximum)
NUMBER(n,m) : Nombre de longueur totale n avec m décimales
DATE : Date, le format par défaut : DD-MM-YY
LONG : Flot de caractères

b) Création de tables

http://www.postgresql.org/docs/9.2/static/sql-createtable.html

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT
EXISTS ] table_name ( [
  { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]
```

c) Modification de la structure d'une table

http://www.postgresql.org/docs/9.2/static/sql-altertable.html

```
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ]
    action [, ... ]
```

where action is one of:

```
    ADD [ COLUMN ] column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
    DROP [ COLUMN ] [ IF EXISTS ] column_name [ RESTRICT | CASCADE ]
    ALTER [ COLUMN ] column_name [ SET DATA ] TYPE data_type [ COLLATE collation ] [ USING expression ]
    ALTER [ COLUMN ] column_name SET DEFAULT expression
    ALTER [ COLUMN ] column_name DROP DEFAULT
    ALTER [ COLUMN ] column_name { SET | DROP } NOT NULL
    ALTER [ COLUMN ] column_name SET STATISTICS integer
    ALTER [ COLUMN ] column_name SET ( attribute_option = value [, ... ] )
    ALTER [ COLUMN ] column_name RESET ( attribute_option [, ... ] )
    ALTER [ COLUMN ] column_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
    ADD table_constraint [ NOT VALID ]
```

Etc .....

d) Destruction d'une table

http://www.postgresql.org/docs/9.2/static/sql-droptable.html

```
DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

e) Consultation de la structure d'une table (équivalent à la commande describe sous Oracle)

\d <nom de la table>

4. SQL LMD

a) Interrogation

SELECT [**DISTINCT**] <nom de colonne>[,<nom de colonne>]…
FROM <nom de table>[,<nom de table>]…
WHERE <condition>
GROUP BY <nom de colonne>[,<nom de colonne>]…
HAVING <condition avec calculs verticaux>
ORDER BY <nom de colonne>[,<nom de colonne>]…

b) Insertion de données

INSERT INTO <nom de table> [(colonne,…)] **VALUES** (valeur,…)

c) Modification de données

UPDATE <nom de table> **SET** colonne=valeur,… **WHERE** condition

d) Suppression de données

DELETE FROM <nom table> **WHERE** condition

5. Prise en main

a) Connectez-vous à la base de données

b) Créez la table correspondant à la relation suivante :

```
Pays (numpays,nom, nbhabitants, superficie)
```

(par convention, un attribut souligné est clé primaire de la relation)

c) Insérez les tuples suivants:

```
(1, France, 60, 980), (2, Belgique, 12, 78), (3, Italie, 50, 850)
```

d) Effacez la table et les données

e) Avec un éditeur (kate,vi,…), créez un fichier `tp1-1.sql` contenant les commandes vues en b) et c) puis chargez le fichier dans l'éditeur PostgreSQL.

f) Créez un fichier `tp1-2.sql` qui contiendra la relation et les tuples suivants :

```
Ville (numville, nom)

(1, Strasbourg), (2, Paris), (3, Rome), (4, Bruxelles), (5, Venise)
```

g) Chargez ce dernier fichier sous PostgreSQL. Sous l'éditeur, modifiez la table `Ville` en lui rajoutant une colonne correspondant au nombre d'habitants. Modifiez les données avec les paramètres suivants :

Ville	**Nombre d'habitants**
Strasbourg	267000
Paris	2147000
Rome	2655000
Bruxelles	997000
Venise	130000

h) Quelle commande permet d'obtenir la description de la table `Pays` ?

i) Quelle commande permet d'obtenir la liste des tuples de la table `Pays` ?
Quelle commande permet d'afficher le nom et le nombre d'habitants des pays ? D'afficher le résultat précédent avec un tri sur le nombre d'habitants par ordre croissant, puis décroissant ? D'afficher le nom des pays ayant plus de 40 millions d'habitants ?

j) Sous l'éditeur PostgreSQL, modifiez la table `Ville` pour lui rajouter une colonne `refpays` dont les valeurs seront les clés vers la table `Pays`.

k) Modifiez les données de la table `Ville` en conséquence.

l) Supprimez la table `Pays`.

m) Affichez les villes avec le pays correspondant.