Conception de bases de données

Implémentation de bases de données relationnelles avec PostgreSQL

LAP1

utc Formation

Stéphane Crozat

17 février 2017

Table des matières

Cours

I

A. Introduction à PostgreSQL : présentation, installation, manipulations de base

1. Présentation

PostgreSQL est :

- un SGBDR
- libre (licence BSD)
- multi-plate-formes (Unix, Linux, Windows, MacOS, ...)
- puissant
- très respectueux du standard
- très bien documenté

Fondamental: Documentation de PostgreSQL

https://www.postgresql.org/docs/[1]

(en français : *http://docs.postgresqlfr.org/*[2])

Fonctionnement général

PostgreSQL comme la grande majorité des SGBD est un logiciel client-serveur. Il faut donc pour l'utiliser un serveur (programme *postgresql* sous Linux) et un client.

Il existe plusieurs clients, les deux plus utilisés sont le client textuel *psql* et le client graphique *PgAdmin III*.

Complément

- *http://www.postgresql.org/*[3]
- *http://www.postgresql.fr/*[4]

2. Installation

Attention

Nous présentons ici une installation *a minima* de PostgreSQL uniquement à des fins de test et d'apprentissage, et pour un usage local. Pour mettre en production une

1 - https://www.postgresql.org/docs/
2 - http://docs.postgresqlfr.org/
3 - http://www.postgresql.org/
4 - http://www.postgresql.fr/

base de données PostgreSQL sur le réseau, il faut suivre des directives supplémentaires, notamment pour assurer la sécurité du système, sa sauvegarde...

Ces thèmes ne sont pas abordés dans le cadre de ce cours.

PostgreSQL est à l'origine une base de données conçue pour Unix, son installation et son fonctionnement sont possibles aujourd'hui sur les trois OS, mais Linux reste son environnement de prédilection. C'est l'architecture PostgreSQL sur Linux qui est étudiée ici.

https://www.postgresql.org/download/[5]

Exemple : Installer PostgreSQL sur Ubuntu

Une installation sur Ubuntu est très facile :

1. Installer les paquets : `sudo apt-get install postgresql`
2. Activer l'utilisateur *postgres* : `sudo passwd postgres`
3. Se connecter en tant que *postgres* : `su postgres`
4. Lancer le client *psql* : `psql -h localhost -d postgres -U postgres` (ou `psql`)

Complément

https://doc.ubuntu-fr.org/postgresql[6]

Complément : Base de données par défaut

L'installation crée une base de données par défaut qui s'appelle *postgres* et un utilisateur *postgres* qui possède cette base (`OWNER`).

Complément : Informations réseau

- Le port standard de communication de PosgreSQL est *5432*.
- Si le client et le serveur sont sur la même machine, le client peut bien entendu se connecter au serveur *localhost* sur le port *5432*.

Complément : Installer PostgreSQL sous Windows

1. Télécharger un *installer* depuis *http://www.enterprisedb.com/products-services-training/pgdownload#windows*[7]
2. Exécuter l'installation en validant les propositions par défaut (et sans installer les éventuels programmes complémentaires proposés à l'issue de l'installation)
3. Exécuter le client *psql* (également appelé *Shell SQL*)

3. Le client textuel "psql"

Définition : psql

`psql` est le client textuel de PostgreSQL.

Syntaxe : Connexion à un serveur PostgreSQL avec le client psql

```
1  psql -h server.adress.or.ip -d database -U user
```

Complément

La commande *psql* utilisée sans paramètre se connecte au serveur *localhost* avec pour nom de *database* **et** pour nom de *user* le nom de l'utilisateur du système qui invoque la commande (utilisateur Linux typiquement).

`me@mypc:~$ psql` équivaut à : `me@mypc:~$ psql -h localhost -d me -U me`

5 - https://www.postgresql.org/download/
6 - https://doc.ubuntu-fr.org/postgresql
7 - http://www.enterprisedb.com/products-services-training/pgdownload#windows

Syntaxe : Écrire une instruction SQL

```
1 dbnf17p015=> SELECT * FROM maTable ;
```

Syntaxe : Écrire une instruction SQL sur plusieurs lignes

Une instruction SQL peut s'écrire sur une ou plusieurs lignes, le `retour chariot` n'a pas d'incidence sur la requête, c'est le `;` qui marque la fin de l'instruction SQL et provoque son exécution.

```
1 dbnf17p015=> SELECT *
2 dbnf17p015-> FROM maTable
3 dbnf17p015-> ;
```

On notera dans `psql` la différence entre les caractères => et -> selon que l'on a ou pas effectué un retour chariot.

Fondamental : Commandes de base : aide

`\?` : Liste des commandes `psql`

`\h` : Liste des instructions SQL

`\h CREATE TABLE` : Description de l'instruction SQL `CREATE TABLE`

Fondamental : Commandes de base : catalogue

`\d` : Liste des relations (catalogue de données)

`\d maTable` : Description de la relation `maTable`

Fondamental : Commandes de base : quitter

`\q` : Quitter `psql`

Complément

http://www.postgresql.org/docs/current/static/app-psql.html[8]

4. Types de données

Types standards

- numériques : integer (int2, int4, int8), real (float4, float8)
- dates : date (time, timestamp)
- chaînes : char, varchar, text
- autres : boolean

Complément : Documentation

http://docs.postgresqlfr.org/8.1/datatype.html[9]

8 - http://www.postgresql.org/docs/current/static/app-psql.html
9 - http://docs.postgresqlfr.org/8.1/datatype.html

B. Éléments complémentaires indispensables à l'utilisation de PostgreSQL

1. Exécuter des instructions SQL depuis un fichier

Il est souvent intéressant d'exécuter un fichier contenant une liste de commandes SQL, plutôt que de les entrer une par une dans le terminal. Cela permet en particulier de recréer une base de données à partir du script de création des tables.

Syntaxe

Pour exécuter un fichier contenant du code SQL utiliser la commande PostgreSQL `\i chemin/fichier.sql`

- `chemin` désigne le répertoire dans lequel est le fichier `fichier.sql`
- le dossier de travail de `psql` est le dossier dans lequel il a été lancé, le script peut être lancé à partir de son dossier `home` pour en être indépendant (`~/.../fichier.sql`)
- chaque commande doit être terminée par un `;`

```
1  dbnf17p015=> \i /home/me/bdd.sql
```

2. Importer un fichier CSV

Syntaxe

```
1  \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV DELIMITER ';'
   QUOTE '"'
```

- `WITH` introduit les options de l'import
- `CSV` indique qu'il s'agit d'un fichier CSV
- `DELIMITER 'c'` indique que le caractère **c** est utilisé comme délimiteur de champ (en général **;** ou **,**)
- `QUOTE 'c'` indique que le caractère **c** est utilisé comme délimiteur de chaîne (en général **"**)

Remarque

- La table `nom_table` doit déjà exister
- Le nombre de colonnes spécifié doit correspondre au nombre de colonnes du fichier CSV
- Les types doivent être compatibles

Remarque

Ajouter l'option `HEADER` après `WITH CSV` si le fichier CSV contient une ligne s'entête.

```
1  \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV HEADER DELIMITER
   ';' QUOTE '"'
```

Méthode : Localisation du fichier CSV depuis psql

Par défaut, la commande `\copy` prendra le chemin du répertoire courant au moment où la commande `psql` a été lancée.

Sous `psql`, vous pouvez utiliser les commandes :

- `dbnf17p007=> \! pwd`
  Pour exécuter la commande *shell* `pwd` et obtenir le répertoire courant
- `dbnf17p007=> \cd directory`

Pour changer le répertoire courant

Complément

PostgreSQL sous Linux
Fichier CSV - p.25

3. Notion de schéma

Définition

A PostgreSQL database cluster contains one or more named databases. Users and groups of users are shared across the entire cluster, but no other data is shared across databases. Any given client connection to the server can access only the data in a single database, the one specified in the connection request.

A database contains one or more named schemas, which in turn contain tables. Schemas also contain other kinds of named objects, including data types, functions, and operators. The same object name can be used in different schemas without conflict; for example, both schema1 and myschema can contain tables named mytable. Unlike databases, schemas are not rigidly separated: a user can access objects in any of the schemas in the database he is connected to, if he has privileges to do so.

http://www.postgresql.org/docs/8.4/static/ddl-schemas.html[10]

Graphique 1 Organisation en cluster, base, schéma, table dans PostgreSQL

Syntaxe : Créer un schéma

```
1 CREATE SCHEMA myschema;
```

10 - http://www.postgresql.org/docs/8.4/static/ddl-schemas.html

Syntaxe : Créer une table dans un schéma

```
1 CREATE TABLE myschema.mytable (
2 ...
3 );
```

Syntaxe : Requêter dans un schéma

```
1 SELECT ...
2 FROM myschema.mytable
```

Exemple

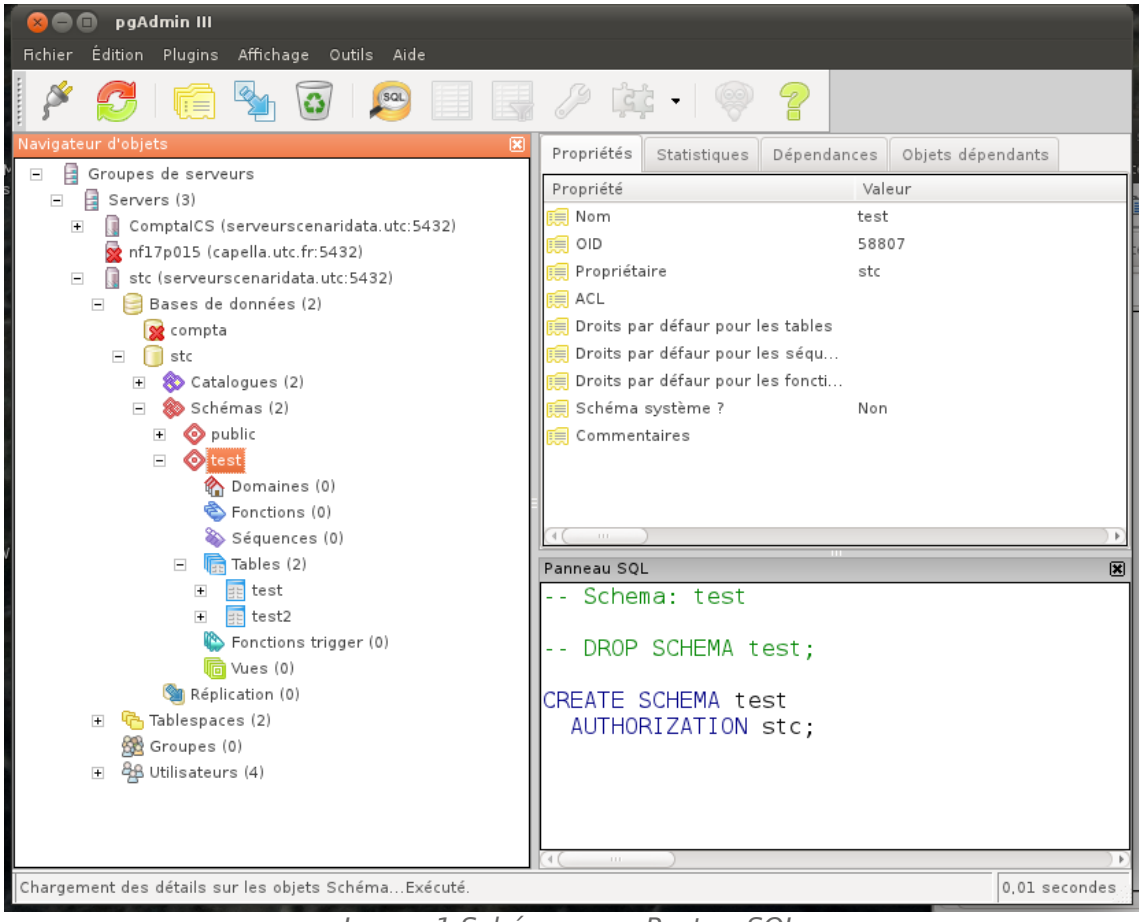

Image 1 Schéma sous PostgreSQL

Syntaxe : Catalogue : schémas

\dn : Liste des schémas de ma base de données

Complément : Schéma par défaut : search_path

Afin d'alléger la syntaxe il est possible de définir un schéma par défaut, dans lequel seront créer les tables non-préfixées et un ou plusieurs schémas par défaut dans lesquels seront requêtées les tables non-préfixées.

```
1 SET search_path TO myschema,public;
```

Cette instruction définit le schéma `myschema` comme schéma par défaut pour la création de table et le requêtage, puis `public` pour le requêtage, le premier étant prioritaire sur le second :

- `CREATE mytable` créera ma `mytable` dans le schéma `mychema`.

- `SELECT FROM mytable` cherchera la table dans la schéma `mychema`, puis dans le schéma `public` si la table n'existe pas dans le premier schéma.

Remarque : Schéma "public"

Le schéma `public` est un schéma créé par défaut à l'initialisation de la base, et qui sert de schéma par défaut en l'absence de toute autre spécification.

Remarque : Catalogue : \d

La liste des tables retournée par `\d` dépend du *search_path*. Pour que `\d` retourne les tables de `schema1`, il faut donc exécuter l'instruction :

```
SET search_path TO public, schema1
```

Complément : Pour aller plus loin

http://www.postgresql.org/docs/8.4/static/ddl-schemas.html[11]

4. Créer des bases de données et des utilisateurs

Syntaxe : Créer un utilisateur

```
1  CREATE USER user1 WITH ENCRYPTED PASSWORD 'password';
```

Syntaxe : Créer une base de données

```
1  CREATE DATABASE mydb WITH OWNER user1;
2
```

Complément : Supprimer des bases de données et des utilisateurs

```
1  DROP DATABASE mydb;
2  DROP USER user1;
```

Syntaxe : Catalogue : utilisateurs et bases de données

- `\du` : liste des utilisateurs
- `\l` : liste des bases de données

Syntaxe : psql : changer d'utilisateur

`\c db user` : se connecter à la base *db* avec le compte *user*.

5. PostgreSQL sous Linux

Syntaxe : Exécuter une instruction PostgreSQL depuis Linux

```
1  psql -c "instruction psql"
```

Exemple : Exécuter un script PostgreSQL depuis Linux

```
1  #!/bin/sh
2  psql -c "DROP DATABASE mydb"
3  psql -c "CREATE DATABASE mydb"
4  psql -c "GRANT ALL PRIVILEGES ON DATABASE mydb TO user1"
```

11 - http://www.postgresql.org/docs/8.4/static/ddl-schemas.html

Complément : psql : exécuter une commande Linux

- `\!` : permet d'exécuter certaines commandes du *shell* Linux depuis le client *psql*.

6. Les clients graphiques pgAdminIII et phpPgAdmin

pgAdminIII

Un client graphique une interface graphique permettant d'effectuer les mêmes opérations qu'avec le client `psql`.

- Le client graphique pgAdminIII est un client lourd qui fonctionne très bien sous Linux et sous Windows.
- Le client graphique phpPgAdmin est un client léger (qui tourne dans un navigateur Web donc).

Déclarer une connexion dans pgAdminIII

1. Sélectionner `Fichier > Ajouter un serveur`
2. Saisissez les informations de connexion à un serveur PostgreSQL existant :
   - Hôte : tuxa.sme.utc
   - Port : 5432 (port standard de PostgreSQL)
   - Base : dbnf17p...
   - Nom : nf17p...
   - Mot de passe : ...

Image 2 Fenêtre de connexion pgAdmin III

Ouvrir un terminal SQL dans pgAdminIII

1. Sélectionner sa base de données dans la liste `Bases de données`
2. Sélectionner `Outils > Éditeur de requêtes` (ou `CTRL+E`)

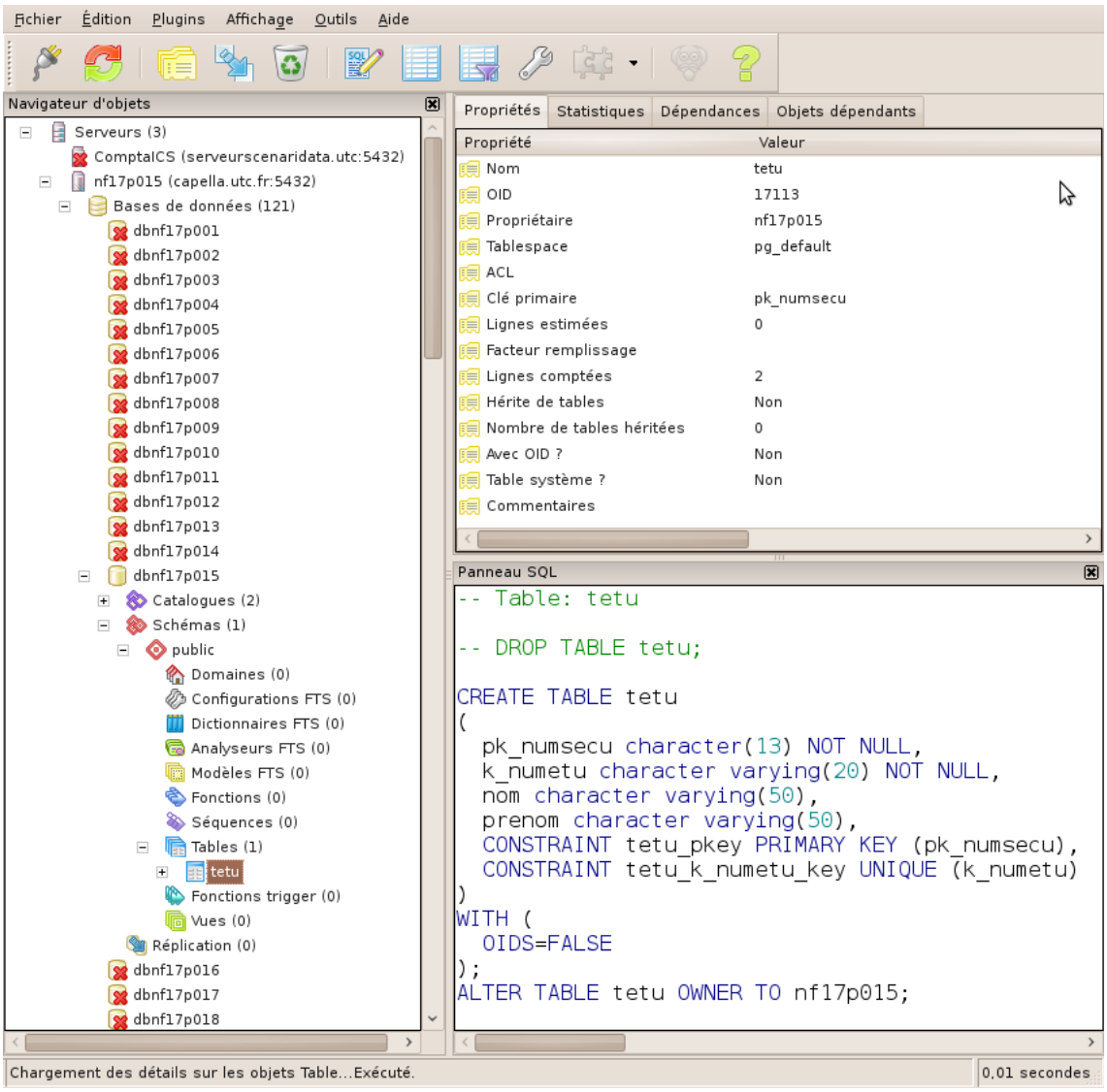

Image 3 pgAdminIII

Complément : phpPgAdmin

http://phppgadmin.sourceforge.net[12]

7. Compléments

Complément : Héritage (clause INHERITS)

http://www.postgresql.org/docs/current/static/sql-createtable.html[13]

Complément : PL/pgSQL

http://www.postgresql.org/docs/current/static/plpgsql.html[14]

Complément : Autres langages procéduraux (PL)

- PL/Tcl
- PL/Perl

12 - http://phppgadmin.sourceforge.net
13 - http://www.postgresql.org/docs/current/static/sql-createtable.html
14 - http://www.postgresql.org/docs/current/static/plpgsql.html

- PL/Python
- ...

http://www.postgresql.org/docs/current/static/xplang.html[15]
http://www.postgresql.org/docs/current/static/server-programming.html[16]

Complément : Triggers

http://www.postgresql.org/docs/current/static/sql-createtrigger.html[17]
(*http://www.postgresql.org/docs/current/static/triggers.html*[18])

15 - http://www.postgresql.org/docs/current/static/xplang.html
16 - http://www.postgresql.org/docs/current/static/server-programming.html
17 - http://www.postgresql.org/docs/current/static/sql-createtrigger.html
18 - http://www.postgresql.org/docs/current/static/triggers.html

Exercices

II

A. Découverte d'un SGBDR avec PostgreSQL

1. Configuration technique pour réaliser les exercices

Fondamental: Serveur Linux + Client Linux

La meilleure configuration pour réaliser ces exercices est de disposer :

1. **d'un serveur Linux hébergeant un serveur PostgreSQL ;**
2. **d'une base de données déjà créée (cela peut être la base par défaut *postgres*) ;**
3. **d'un utilisateur ayant les pleins droits sur cette base de données (cela peut être l'utilisateur par défaut *postgres*) ;**
4. **d'un client *psql* sous Linux connecté à la base de données.**

Rappel

Présentation

Le client textuel "psql"

Installation (du serveur PostgreSQL et du client psql)

Serveur Linux + Client Windows

Si un serveur est disponible sous Linux mais que le client est sous Windows, il y a deux solutions :

- **Se connecter en SSH au serveur Linux avec Putty[Putty] (on est ramené au cas précédent).**
- Installer le client *psql* sous Windows, mais cette solution est déconseillée :
  - le terminal est bien moins confortable que sous Linux,
  - une partie des questions sont relatives aux systèmes Linux et ne pourra être réalisée.

Complément : Serveur Windows + Client Windows

Cette configuration permettra de faire une partie des exercices, avec des adaptations.

2. Connexion à une base de données PostgreSQL

Cet exercice commence alors que vous êtes connecté à une base Oracle avec le client *psql*.

```
1  psql (9.x.x)
2  Saisissez « help » pour l'aide.
3
4  mydb=>
```

Demander de l'aide

Demander de l'aide en testant :

- help
- \?
- \h
- \h CREATE TABLE
- \h SELECT

3. Créer une base de données avec PostgreSQL

Créer une table

Exécuter les instructions suivantes.

```
1  CREATE TABLE etu (
2  pknumsecu CHAR(13) PRIMARY KEY,
3  knumetu VARCHAR(20) UNIQUE NOT NULL,
4  nom VARCHAR(50),
5  prenom VARCHAR(50));
```

```
1  INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
2  VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3  INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
4  VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

```
1  CREATE TABLE uv (
2  pkcode CHAR(4) NOT NULL,
3  fketu CHAR(13) NOT NULL,
4  PRIMARY KEY (pkcode, fketu),
5  FOREIGN KEY (fketu) REFERENCES etu(pknumsecu));
```

```
1  INSERT INTO uv (pkcode, fketu)
2  VALUES ('NF17', '1800675001066');
3  INSERT INTO uv (pkcode, fketu)
4  VALUES ('NF26', '1800675001066');
5  INSERT INTO uv (pkcode, fketu)
6  VALUES ('NF29', '1800675001066');
```

Question 1

[*Solution n°1 p 20*]

Utiliser le catalogue pour vérifier la création de la table.

Question 2

[*Solution n°2 p 20*]

Utiliser deux instructions SELECT pour vérifier le contenu de la table.

4. Import de données depuis un fichier CSV

Nous allons à présent réinitialiser la base avec des données contenues dans un fichier.

Question 1

[*Solution n°3 p 20*]

Exécuter les instructions nécessaires afin de **supprimer** les données existantes dans les tables (instruction DELETE du SQL LMD). Vérifier que les tables sont vides.

Question 2

Créer les deux fichiers de données suivants, respectivement *etus.csv* et *uvs.csv*. Regarder le contenu des fichiers. Pourquoi les appelle-t-on des fichiers CSV ?

```
1;A;Dupont;Pierre
2;B;Durand;Georges
3;C;Duchemin;Paul
4;D;Dugenou;Alain
5;E;Dupied;Albert
```

```
1;NF17
1;NF18
1;NF19
1;NF20
1;LA13
1;PH01
2;NF17
2;NF18
2;NF19
2;TN01
2;LA14
2;PH01
3;NF17
3;NF18
3;NF19
3;NF21
3;LA14
3;PH01
4;NF17
4;NF20
4;NF21
4;GE10
4;LA14
4;PH01
5;NF17
5;NF18
5;NF20
5;GE10
5;PH01
5;PH02
```

Indice :

Fichier CSV - p.25

Question 3

[*Solution n°4 p 20*]

Insérer les données en complétant les instructions suivantes.

```
\copy ... (...) FROM '...' WITH CSV DELIMITER '...'
\copy ... (...) FROM '...' WITH CSV DELIMITER '...'
```

Indice :

Importer un fichier CSV

Question 4

[*Solution n°5 p 20*]

Écrivez les requêtes permettant d'obtenir le nombre d'UV suivies par un étudiant et le nombre d'étudiants inscrits par UV.

5. Manipulation de schémas

Question 1

Visualiser la liste des schémas existants dans votre base de données.

Indice :

Notion de schéma

Question 2

[*Solution n°6 p 20*]

Créer un second schéma *loisir*.

Question 3

[*Solution n°7 p 21*]

Créer une table *sport* dans le schéma *loisir* ; l'objectif est d'indiquer pour chaque étudiant, la liste des sports qu'il pratique, par exemple : Tennis, Karaté, Aviron...
Vérifier votre création dans le dictionnaire.

6. Exécution de fichiers SQL et de scripts Linux

Reporter dans un fichier SQL l'ensemble des instruction permettant de supprimer, de créer et d'instancier les tables de la base de données.

Question 1

Exécuter ce fichier depuis *psql*.

Indice :

Exécuter des instructions SQL depuis un fichier

Question 2

Exécuter ce fichier depuis un script Linux.

Indice :

PostgreSQL sous Linux

7. Test de pgAdminIII

Connexion depuis un PC avec pgAdmin III

Installer si nécessaire, puis lancer et tester le programme `pgAdminIII`.
Exécuter une ou deux requêtes permettant de se familiariser avec son fonctionnement.

Devoirs

III

A. Tourisme

[30 min]

Une commune veut mieux tenir à jour et représenter les offres disponibles en terme de tourisme, et cherche ainsi à en établir une base de données.

Soit les modèles conceptuel et logique suivants établis à partir de la note de clarification.

Une architecture LAPP a été choisie.

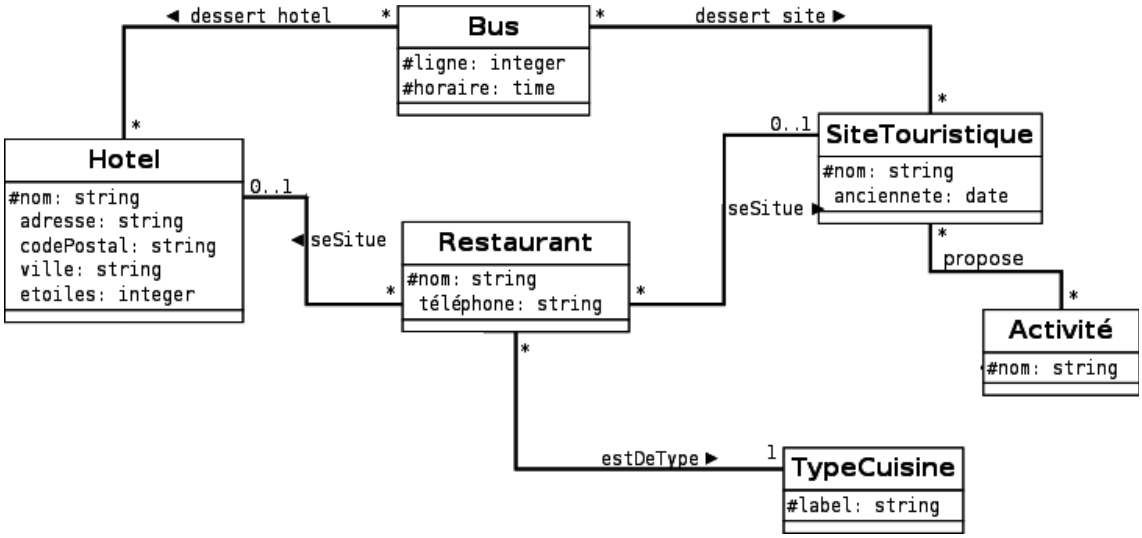

Image 4 Tourisme

```
1  Bus(#ligne:integer, #horaire:time)
2  SiteTouristique(#nom:string, ancienneté:integer)
3  Hotel(#nom:string, adresse:string, codePostal:string, ville:string,
   etoiles:integer)
4  Restaurant(#nom:string, téléphone:string, type_cuisine=>TypeCuisine(label),
   hotel=>Hotel(nom), site=>SiteTouristique(nom))
5  TypeCuisine(#label:string)
6  Activité(#nom:string)
7  ActivitéParSite(#site=>SiteTouristique(nom), #activité=>Activité(nom))
8  Dessert_hotel(#bus_ligne=>Bus(ligne), #bus_horaire=>Bus(horaire),
   #hotel=>Hotel(nom))
9  Dessert_site(#bus_ligne=>Bus(ligne), #bus_horaire=>Bus(horaire),
   #site=>SiteTouristique(nom))
```

Question 1

Implémenter le schéma relationnel avec PostgreSQL (instructions CREATE TABLE).

Question 2

Instancier la base de données avec des données représentatives (instructions INSERT).

Question 3

Proposer des exemples caractéristiques de données qui seront refusées grâces aux contraintes posées par le modèle (instructions INSERT donnant lieu à des erreurs).

Solution des exercices

> Solution n°1 *(exercice p. 15)*

```
\d
\d etu
\d uv
```

> Solution n°2 *(exercice p. 15)*

```
SELECT *
FROM etu;
```

```
SELECT *
FROM uv;
```

> Solution n°3 *(exercice p. 15)*

```
DELETE FROM uv;
SELECT * FROM uv;
DELETE FROM etu;
SELECT * FROM etu;
```

> Solution n°4 *(exercice p. 16)*

```
\copy etu (pknumsecu, knumetu, nom, prenom) FROM 'etus.csv' WITH CSV DELIMITER
';' QUOTE '"'
\copy uv (fketu, pkcode) FROM 'uvs.csv' WITH CSV DELIMITER ';' QUOTE '"'
```

> Solution n°5 *(exercice p. 16)*

```
SELECT pkcode, COUNT(fketu)
FROM uv
GROUP BY pkcode;
```

```
SELECT fketu, COUNT(pkcode)
FROM uv
GROUP BY fketu;
```

> Solution n°6 *(exercice p. 17)*

```
CREATE SCHEMA loisir;
```

> Solution n°7 *(exercice p. 17)*

```
1  CREATE TABLE loisir.sport (
2  pknom VARCHAR(25) NOT NULL,
3  fketu CHAR(13) NOT NULL,
4  PRIMARY KEY (pknom, fketu),
5  FOREIGN KEY (fketu) REFERENCES etu(pknumsecu));
```

```
1  INSERT INTO loisir.sport (pknom, fketu)
2  VALUES ('Tennis', '1800675001066');
3  INSERT INTO loisir.sport (pknom, fketu)
4  VALUES ('Karaté', '1800675001066');
5
```

```
1  SET search_path TO public,loisir;
2  \d
3  SELECT * FROM loisir.sport;
```

Glossaire

Client

Un client est un programme informatique qui a pour fonction d'envoyer des requêtes à un autre programme informatique, appelé serveur, d'attendre le résultat de cette requête et de traiter le résultat de la requête. Notons qu'un programme peut-être client vis à vis d'un programme et serveur vis à vis d'un autre. On ne prend pas ici le terme client dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes clients.

Serveur

Un serveur est un programme informatique qui a pour fonction de recevoir des requêtes d'un autre programme, appelé client, de traiter ces requêtes et de renvoyer en retour une réponse. Notons qu'un programme peut-être serveur vis à vis d'un programme et client vis à vis d'un autre. On ne prend pas ici le terme serveur dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes serveurs.

Signification des abréviations

- BD — Base de Données
- CSV — Comma Separated Values
- OS — Operating Système (Système d'Exploitation)
- SGBD — Système de Gestion de Bases de Données
- XML — eXtensible Markup Language

Références

[*Putty*] Putty est un logiciel qui propose un terminal sous Windows plus confortable que le terminal par défaut. De plus c'est un client *ssh*, plus sécurisé que *telnet*. C'est un petit exécutable qui ne nécessite pas d'installation, il peut donc être téléchargé sur le disque Z et lancé directement.

http://www.chiark.greenend.org.uk/~sgtatham/putty/[19]

19 - http://www.chiark.greenend.org.uk/~sgtatham/putty/

Contenus annexes

- Fichier CSV

Définition : Fichier CSV

CSV est un format informatique permettant de stocker des données tabulaires dans un fichier texte.

Chaque ligne du fichier correspond à une ligne du tableau. Les valeurs de chaque colonne du tableau sont séparées par un caractère de séparation, en général une **virgule** ou un **point-virgule**. Chaque ligne est terminée par un **caractère de fin de ligne** (*line break*).

Toutes les lignes contiennent **obligatoirement** le même nombre de valeurs (donc le même nombre de caractères de séparation). Les valeurs vides doivent être exprimées par deux caractères de séparation contigus.

La taille du tableau est le nombre de lignes multiplié par le nombre de valeurs dans une ligne.

La première ligne du fichier peut être utilisée pour exprimer le nom des colonnes.

Syntaxe

```
1 [NomColonne1;NomColonne2;...;NomColonneN]
2 ValeurColonne1;ValeurColonne2;...;ValeurColonneN
3 ValeurColonne1;ValeurColonne2;...;ValeurColonneN
4 ...
```

Exemple : Fichier CSV sans entête

```
1 Pierre;Dupont;20;UTC;NF17
2 Pierre;Dupont;20;UTC;NF26
3 Paul;Durand;21;UTC;NF17
4 Jacques;Dumoulin;21;UTC;NF29
```

Exemple : Fichier CSV avec entête

```
1 Prenom;Nom;Age;Ecole;UV
2 Pierre;Dupont;20;UTC;NF17
3 Pierre;Dupont;20;UTC;NF26
4 Paul;Durand;21;UTC;NF17
5 Jacques;Dumoulin;21;UTC;NF29
```

Exemple : Valeur nulle

```
1 Jacques;Dumoulin;;UTC;NF29
```

L'âge est inconnu (NULL).

Attention : Variations...

La syntaxe des fichiers CSV n'est pas complètement standardisée, aussi des variations peuvent exister :

- **Les chaînes de caractères peuvent être protégées par des guillemets (les guillemets s'expriment alors avec un double guillemet).**
- **Le caractère de séparation des nombres décimaux peut être le point ou la virgule (si c'est la virgule, le caractère de séparation doit être différent)**
- **...**

Un des problème les plus importants reste l'encodage des caractères qui n'est pas spécifié dans le fichier et peut donc être source de problèmes, lors de changement d'OS typiquement.

Méthode : Usage en base de données

Les fichiers CSV sont très utilisés en BD pour échanger les données d'une table (export/import).

Les SGBD contiennent généralement des utilitaires permettant d'exporter une table ou un résultat de requête sous la forme d'un fichier CSV, en spécifiant un certain nombre de paramètres (caractère de séparation de valeur, caractère de fin de ligne, présence ou non d'une ligne de définition des noms des colonnes, etc.). De même ils proposent des utilitaires permettant d'importer un fichier CSV dans une table (en spécifiant les mêmes paramètres), voire de créer directement une table à partir du fichier CSV (quand les noms des colonnes sont présents).

Complément : Fichiers à largeur de colonne fixe

Les fichiers à largeur de colonne fixe n'utilisent pas de séparateur de colonne, mais imposent **le même nombre de caractères** pour chaque cellule. L'avantage est de ne pas avoir à spécifier le caractère de séparation, l'inconvénient est la taille de fichier supérieure si les valeurs ne font pas toutes la même largeur.

Complément : XML

Les fichiers XML tendent de plus en plus à remplacer les fichiers CSV car ils permettent d'être beaucoup plus expressifs sur le schéma d'origine. Ils sont également plus standards (encodage spécifié, principe de séparation des données par les *tags*, etc.). Leur seul inconvénient est d'être plus verbeux et donc plus volumineux.

Complément : Tables externes

Certains SGBD, comme Oracle, permettent de créer des tables dites **externes**, qui autorisent de créer un schéma de table **directement sur un fichier CSV**, permettant ainsi un accès SQL standard à un fichier CSV, sans nécessité de l'importer d'abord dans une table.