Perl et les bases de données (DBI)

par djibril (Site personnel)

Date de publication : 2009-01-26

Dernière mise à jour : 2011-04-03

Ce tutoriel vous permettra d'apprendre les bases d'utilisation du module DBI afin de se connecter à une base de données et de lire ou insérer des données.
La notion des *placeholders* pour protéger et optimiser les requêtes sera abordée.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

A - Introduction

Le but de ce tutoriel est d'apprendre à interagir avec les bases de données en Perl.
Nous commencerons par définir la notion de base de données. Il vous sera ensuite expliqué et listé ce dont vous avez besoin pour travailler.

A-1 - Les bases de données

Une base de données (communément appelée *"BD"* ou *"database"* en anglais) est une entité dans laquelle on stocke des informations de façon structurée. Il est possible de localiser facilement ces données et de les mettre rapidement à jour.
Un autre avantage de la BD est qu'elle peut être interrogée par plusieurs personnes, dans un réseau sécurisé, par l'intermédiaire d'un système de gestion de base de données (communément abrégé *"SGBD"*).
Le terme entité peut vous paraître abstrait. Pour mieux comprendre, retenez qu'une base de données n'est en fait qu'un ensemble de tables de données, un peu à la manière dont vous pourriez inscrire des données dans une feuille de tableur (Excel ou autre).
Le but de ce tutoriel n'est pas de vous apprendre les notions de BD et SGBD, reportez-vous aux cours du site de developpez.com pour avoir de plus amples informations sur ce sujet.

Voici quelques liens :

1. des **cours SQL** ;
2. des **cours SGBD** ;
3. et vous en trouverez facilement d'autres sur developpez.com.

A-2 - Interaction avec les bases de données

Il existe différentes façons d'interagir avec une base de données :

- en ligne de commande via une console Dos, Unix, Linux... ;
- à partir d'une interface web (avec un logiciel comme phpMyAdmin) ;
- via un langage de programmation comme Perl.

Notez que Perl n'est pas le seul langage qui permet d'interroger une base de données. La plupart des langages de programmation le permettent (ex : PHP, Java, C#...).

B - Perl et les bases de données

Cette section du tutoriel permet de rentrer dans le vif du sujet ! Il faut dans un premier temps installer les outils nécessaires pour commencer à travailler sur les bases de données avec Perl.

B-1 - Prérequis pour travailler

Pour pouvoir créer ou interroger une base de données, vous devez dans un premier temps installer un système de gestion de base de données (**MySQL**, **PostgreSQL**, ou autres). Il en existe beaucoup. Si vous avez installé **EasyPHP**, vous disposez d'un SGBD.

B-2 - Modules Perl nécessaires

Perl a besoin du module DBI pour interagir avec les bases de données et d'un module driver pour les SGBD. Dans cet article, nous utiliserons MySQL comme SGBD. Nous aurons besoin du module driver adéquat : **DBD::mysql**.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

Sachez qu'il existe des drivers pour toutes les SGBD que vous souhaiterez utiliser. Ils sont disponibles sur le site du CPAN et sont généralement sous la forme **DBD::***.

Exemple de drivers

- **DBD::Pg** PostgreSQL database driver for the DBI module ;
- **DBD::Oracle** Oracle database driver for the DBI module ;
- **DBD::Sybase** Sybase database driver for the DBI module ;
- **DBD::Ingres** DBI driver for Ingres database systems ;
- **site officiel de Perl DBI** ;
- ...

Si vous rencontrez des difficultés dans l'installation des modules, lisez le tutoriel sur **l'installation des modules Perl** écrit par nos soins ! Sinon, le forum Perl est toujours à votre disposition !

B-3 - Débuter par l'exemple

Nous sommes maintenant prêts à coder ! La meilleure façon d'apprendre est de faire un exemple. Voici donc l'énoncé d'un exercice qui nous permettra de comprendre les commandes de base.

3-1 - Enoncé de l'exercice

Afin de jouer avec le module DBI, nous allons créer un programme dont le but sera d'effectuer les tâches suivantes :

- lister tous les départements d'une région ;
- afficher le nom du département, de la région et le nombre de communes en fonction d'un numéro de département ;
- donner le nom du département et de la région d'une commune.

Pour résoudre cet exercice, il faut que l'on puisse accéder à ces informations. Pour cela, nous téléchargeons trois fichiers contenant toutes les communes, départements et régions de France connus au 1er janvier 2011 sur le site de l'**INSEE**. Grâce à ces fichiers, on crée une base de données et trois tables SQL.
Voici un affichage de quelques lignes de ces fichiers :

REGION	CHEFLIEU	TNCC	NCC	NCCENR
1	971053	3	GUADELOUPE	Guadeloupe
11	75056	1	ILE-DE-FRANCE	Île-de-France

CDC	CHEFLIEU	REG	DEP	COM	AR	CT	TNCC	ARTMAJ	NCC	ARTMIN	NCCENR
1	4	11	75	56	1	99	0		PARIS		Paris
0	0	82	1	42	2	27	0		BEY		Bey
0	0	24	41	023	1	13	0		BOURRE		Bourré
0	0	82	26	142	1	06	0		GLANDA		Glandage
0	0	74	87	003	1	28	1		ARNAC-LA-POSTE		Arnac-la-Poste

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

REGION	DEP	CHEFLIEU	TNCC	NCC	NCCENR
82	1	1053	5	AIN	Ain
22	2	2408	5	AISNE	Aisne

3-2 - Création de la base de données, des tables SQL et insertion des données

Créons une base de données que nous nommerons **France2011** via une console (ou une interface Web).

Commande SQL
```
/*  Création de la base de données France2011 */
CREATE DATABASE `France2011`;
```

```
C:\>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 5.1.31-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE `France2011`;
Query OK, 1 row affected (0.04 sec)
```

La suite sera totalement intégrée dans un script Perl.
Créons un script **CreationTableSQLFrance.pl** qui se charge de créer les tables SQL et de les remplir avec les données du fichier.
Voici notre programme :

CreationTableSQLFrance
```
1. #!/usr/bin/perl
2. #===============================================================================
3. # Auteur : djibril
4. # But    : Création des tables SQL et insertion de données
5. #===============================================================================
6. use warnings;
7. use strict;
8.
9. use DBI;                    # Charger le module DBI
10. use vars qw/ $VERSION /;    # Version du script
11. $VERSION = '1.0';
12.
13. # Paramètres de connexion à la base de données
14. my $bd          = 'France2011';
15. my $serveur     = 'localhost';    # Il est possible de mettre une adresse IP
16. my $identifiant = 'root';         # Identifiant
17. my $motdepasse  = 'admin';        # Nous n'avons pas de mot de passe
18.
19. # Connexion à la base de données MySQL
20. my $dbh = DBI->connect( "dbi:mysql:dbname=$bd;host=$serveur;", $identifiant, $motdepasse )
21.   or die "Connexion impossible à la base de données $bd !";
22.
23. # Création des tables
24. print "Création de la table Regions\n";
25. my $sql_creation_table_regions = <<"SQL";
26. CREATE TABLE Regions (
27.   id_region INT             NOT NULL ,
28.   cheflieu  VARCHAR( 6 )    NOT NULL ,
29.   tncc      INT             NOT NULL ,
30.   ncc       VARCHAR( 100 )  NOT NULL ,
31.   nccenr    VARCHAR( 100 )  NOT NULL ,
32.   PRIMARY KEY ( id_region )
33. ) COMMENT = 'Les régions 2011 en France';
34. SQL
35.
36. $dbh->do('DROP TABLE IF EXISTS Regions;') or die "Impossible de supprimer la table Regions\n\n";
37. $dbh->do($sql_creation_table_regions) or die "Impossible de créer la table Regions\n\n";
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

CreationTableSQLFrance

```
38.
39. print "Création de la table Communes\n";
40. my $sql_creation_table_communes = <<"SQL";
41. CREATE TABLE Communes (
42.   id_communes     INT             NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT 'Elle sera générée automatiquem
43.   cdc             INT             NOT NULL ,
44.   cheflieu        VARCHAR( 6 )    NOT NULL ,
45.   id_region       INT             NOT NULL ,
46.   id_departement  VARCHAR( 10 )   NOT NULL ,
47.   com             INT             NOT NULL ,
48.   ar              INT             NOT NULL ,
49.   ct              INT             NOT NULL ,
50.   tncc            INT             NOT NULL ,
51.   artmaj          VARCHAR( 20 )   ,
52.   ncc             VARCHAR( 100 )  NOT NULL ,
53.   artmin          VARCHAR( 20 )   ,
54.   nccenr          VARCHAR( 100 )  NOT NULL
55. ) COMMENT = 'Les communes 2011 en France';
56. SQL
57.
58. $dbh->do('DROP TABLE IF EXISTS Communes;') or die "Impossible de supprimer la table Communes\n\n";
59. $dbh->do($sql_creation_table_communes) or die "Impossible de créer la table Communes\n\n";
60.
61. print "Création de la table Departements\n";
62. my $sql_creation_table_departements = <<"SQL";
63. CREATE TABLE Departements  (
64.   id_departement  VARCHAR( 10 )   NOT NULL COMMENT 'les valeurs sont uniques',
65.   id_region       INT             NOT NULL ,
66.   cheflieu        VARCHAR( 6 )    NOT NULL ,
67.   tncc            INT             NOT NULL ,
68.   ncc             VARCHAR( 100 )  NOT NULL ,
69.   nccenr          VARCHAR( 100 )  NOT NULL ,
70.   PRIMARY KEY ( id_departement )
71. ) COMMENT = 'Les départements 2011 en France';
72. SQL
73.
74. $dbh->do('DROP TABLE IF EXISTS Departements;') or die "Impossible de supprimer la table Departements
\n\n";
75. $dbh->do($sql_creation_table_departements) or die "Impossible de créer la table Departements\n\n";
76.
77. # Lecture des fichiers et insertion des données
78. my $fichier_regions      = 'reg2011.txt';
79. my $fichier_communes     = 'comsimp2011.txt';
80. my $fichier_departements = 'depts2011.txt';
81.
82. # Fichier Region
83. print "Insertion des données dans la table regions\n";
84. open my $fh_regions, '<', $fichier_regions or die "Impossible de lire le fichier $fichier_regions\n";
85. my $entete_fichier_region = <$fh_regions>;
86.
87. # Insertion des données
88. my $requete_sql_region = <<"SQL";
89.   INSERT INTO regions ( id_region, cheflieu, tncc, ncc, nccenr )
90.   VALUES ( ?, ?, ?, ?, ? );
91. SQL
92.
93. my $sth_regions = $dbh->prepare($requete_sql_region) or die $dbh->errstr;
94.
95. while ( my $ligne = <$fh_regions> ) {
96.   chomp $ligne;
97.   my ( $REGION, $CHEFLIEU, $TNCC, $NCC, $NCCENR ) = split /\t/, $ligne;
98.   $sth_regions->execute( $REGION, $CHEFLIEU, $TNCC, $NCC, $NCCENR )
99.     or die "Echec Requête $requete_sql_region : $DBI::errstr";
100. }
101. close $fh_regions;
102.
103. # Fichier Communes
104. print "Insertion des données dans la table Communes\n";
105. open my $fh_communes, '<', $fichier_communes or die "Impossible de lire le fichier $fichier_communes\n";
106. my $entete_fichier_communes = <$fh_communes>;
107.
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

CreationTableSQLFrance

```
108. # Insertion des données
109. my $requete_sql_communes = <<"SQL";
110.   INSERT INTO communes (
111.     id_communes, cdc,            cheflieu,
112.     id_region,   id_departement, com,
113.     ar,          ct,             tncc,
114.     artmaj,      ncc,            artmin,
115.     nccenr )
116.   VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? );
117. SQL
118.
119. my $sth_communes = $dbh->prepare($requete_sql_communes) or die $dbh->errstr;
120. while ( my $ligne = <$fh_communes> ) {
121.   chomp $ligne;
122.   my ( $CDC, $CHEFLIEU, $REG, $DEP, $COM, $AR, $CT, $TNCC, $ARTMAJ, $NCC, $ARTMIN, $NCCENR
 ) = split /\t/,
123.     $ligne;
124.
125.
  $sth_communes->execute( undef, $CDC, $CHEFLIEU, $REG, $DEP, $COM, $AR, $CT, $TNCC, $ARTMAJ, $NCC, $ARTMIN,
126.     $NCCENR )
127.     or die "Echec Requête $requete_sql_communes : $DBI::errstr";
128. }
129. close $fh_communes;
130.
131. # Fichier Departements
132. print "Insertion des données dans la table Departements\n";
133. open my $fh_departements, '<', $fichier_departements
134.   or die "Impossible de lire le fichier $fichier_departements\n";
135. my $entete_fichier_departements = <$fh_departements>;
136.
137. # Insertion des données
138. my $requete_sql_departement = <<"SQL";
139.   INSERT INTO Departements (
140.     id_departement, id_region, cheflieu,
141.     tncc,           ncc,       nccenr )
142.   VALUES ( ?, ?, ?, ?, ?, ? );
143. SQL
144.
145. my $sth_departements = $dbh->prepare($requete_sql_departement) or die $dbh->errstr;
146. while ( my $ligne = <$fh_departements> ) {
147.   chomp $ligne;
148.   my ( $REGION, $DEP, $CHEFLIEU, $TNCC, $NCC, $NCCENR ) = split /\t/, $ligne;
149.   $sth_departements->execute( $DEP, $REGION, $CHEFLIEU, $TNCC, $NCC, $NCCENR )
150.     or die "Echec Requête $requete_sql_departement : $DBI::errstr";
151. }
152. close $fh_departements;
153.
154. # Déconnexion de la base de données
155. $dbh->disconnect();
```

- **Explication :**

Ligne 9 : nous chargeons le module **DBI**.
Ligne 10 - 11 : version de notre programme.
Ligne 13 - 17 : déclaration des variables permettant de se connecter à notre base de données.
Ligne 19 - 21 : pour se connecter à la base de données, on utilise la méthode **connect**. Si nous avions utilisé un SGBD tel PostgreSQL, nous aurions écrit ceci :

Connexion à la base de données PostgreSQL

```
# Connexion à la base de données PostgreSQL
my $dbh = DBI->connect( "dbi:Pg:dbname=$bd;host=$serveur;", $identifiant, $motdepasse )
  or die "Connexion impossible à la base de données $bd !";
```

C'est donc à ce niveau que le module DBI distingue le driver à utiliser et effectue la connexion à notre base. Créons maintenant les tables SQL.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

Ligne 24 - 75 : Les tables SQL sont créées (et au préalable supprimées si elles existaient). Nous avons utilisé la méthode **do** qui peut être utilisée pour toutes les commandes SQL non répétées (c'est-à-dire requêtes exécutées une seule fois) qui ne sont pas de type **SELECT**.
Ligne 77 - à la fin : insertion des données dans les tables SQL.
Nous déclarons les variables contenant les noms des fichiers à analyser pour remplir nos tables SQL (78-80).
Ensuite, nous lisons les fichiers. Prenons pour exemple le fichier **Region**.
En ligne 85, nous lisons l'entête du fichier (ligne 1 du fichier) et en lignes 88-91, nous construisons notre requête d'insertion. Vous remarquez à ce niveau que dans notre requête, nous avons des points d'interrogation. En fait, nous allons utiliser la notion de ***placeholders***.

- **Qu'est-ce que les placeholders ?**

Il existe ce qu'on appelle des *placeholders* dans le module DBI qui permettent d'exécuter les requêtes SQL lancées par Perl plus rapidement et de les protéger. Ils protègent efficacement contre l'injection SQL (fléau des sites web dynamiques), comme DBI est souvent utilisé dans un contexte web.
Ils sont supportés par la plupart des drivers. Je suis incapable de vous dire les drivers qui ne les supportent pas car ce n'est pas précisé dans la documentation.
Si un sympathique lecteur en a la liste, qu'il me fasse signe, je suis preneur !! Ayant utilisé DBD::Mysql et DBD::Pg, je n'ai jamais eu de souci.

- **Revenons à notre exemple**

Nous allons lire notre fichier ligne à ligne et exécuter plusieurs fois le même type de requête. Seules les données changent, mais l'insertion se fera toujours dans les mêmes champs. Nous avons donc **préparé** notre requête (grâce à la méthode **prepare**) en dehors de la boucle while (ligne 93). Cela évite à DBI de devoir créer la requête à chaque ligne du fichier (perte de performance).
Une fois la requête conçue, nous lisons notre fichier ligne à ligne, récupérons les données (tabulées) et exécutons la requête grâce à la méthode **execute** (lignes 98,99).
Voilà, c'est le même principe pour les trois fichiers.
N.B. En ligne 125, nous avons mis le terme **undef**. En fait, pour le champ **id_communes**, nous avons mis un attribut **AUTO_INCREMENT** qui permet de générer un identifiant unique pour toutes les nouvelles lignes.

```
id_communes  INT   NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT 'Elle sera générée automatiquement.',
```

Notre requête SQL ressemble à :

```
INSERT INTO communes (
  id_communes, cdc,           cheflieu,
  id_region,   id_departement, com,
  ar,          ct,             tncc,
  artmaj,      ncc,            artmin,
  nccenr )
VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? );
```

Nous avons exécuté cette requête de la sorte :

```
$sth_communes->execute( undef, $CDC, $CHEFLIEU, $REG, $DEP, $COM, $AR, $CT, $TNCC, $ARTMAJ, $NCC, $ARTMIN,
  $NCCENR )
  or die "Echec Requête $requete_sql_communes : $DBI::errstr";
```

Mettre **undef** permet à Perl de dire à SQL qu'il s'agit de la valeur *NULL*. Comme ce champ contient l'attribut **AUTO_INCREMENT**, SQL incrémentera tout seul la valeur du champ.
Petit Résumé :
Les méthodes **prepare** et **execute** sont généralement utilisées pour toutes les requêtes différentes du type **SELECT** (exemple : **INSERT**, **UPDATE**...).
Vous me direz, *mais **do** aussi* ! Alors laquelle choisir ?
Si vous avez une requête à exécuter **une seule fois**, comme une création de table (comme dans notre script), la mise à jour d'une seule ligne dans votre table SQL, la méthode **do** est recommandée. Par contre, si vous avez plusieurs

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

requêtes à lancer, comme c'est le cas pour l'insertion des données, il est conseillé d'utiliser les méthodes **prepare** et **execute** car il sera plus simple de les optimiser via l'utilisation des placeholders.
Voilà, notre premier script nous permet de créer nos tables et d'y insérer les données.

3-3 - Interrogation de la base de données

Maintenant que notre base de données et les tables sont prêtes, créons notre deuxième script **"GestionRegionFrance.pl"** permettant de résoudre notre exercice.

Son but est de :

- lister tous les départements d'une région choisie par l'utilisateur ;
- afficher le nom du département, de la région et le nombre de communes en fonction d'un numéro de département voulu ;
- afficher le nom du département et de la région d'une commune voulue.

Veuillez installer les modules **Term::UI** et **Term::ReadLine** qui nous permettrons d'interagir avec la console DOS, Unix, Linux...
Les méthodes **fetchrow_array**, **fetchrow_hashref**, **selectrow_hashref**, **selectall_hashref** et **selectrow_array** sont utilisées dans le script. Elles vous seront expliquées ci-dessous.
Examinons progressivement le script **"GestionRegionFrance.pl"**.
Dans un premier temps, nous avons besoin de charger les modules nécessaires, puis de nous connecter à notre base de données.

```
#!/usr/bin/perl
#===============================================================================
# Auteur : djibril
# But    : Interrogation de ma base de données
#===============================================================================

use warnings;
use strict;

use DBI;
use Term::UI;
use Term::ReadLine;

use vars qw/ $VERSION /;    # Version du script
$VERSION = '1.0';

# Paramètres de connexion à la base de données
my $bd          = 'France2011';
my $serveur     = 'localhost';    # Il est possible de mettre une adresse IP
my $identifiant = 'root';         # identifiant
my $motdepassee = 'admin';

# Connexion à la base de données mysql
my $dbh = DBI->connect( "dbi:mysql:dbname=$bd;host=$serveur;", $identifiant, $motdepassee )
  or die "Connexion impossible à la base de données $bd !";
```

Le code ci-dessus est semblable au code du premier script. Il nous permet de nous connecter à la base de données.
Notez que nous avons chargé les modules **Term::UI** et **Term::ReadLine** qui nous permettront de créer une interaction entre le programme et nous à travers la console.

a - Lister tous les départements d'une région

Nous souhaitons afficher tous les départements d'une région de notre choix. Afin de rendre notre programme plus interactif, ce dernier nous proposera une liste de différentes régions. On pourra ainsi effectuer notre choix.
Dans un premier temps, créons une procédure afin de récupérer dans notre base de données toutes les régions de France.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

```
1. # Procédure pour obtenir toutes les régions de France - méthode fetchrow_array
2. sub obtenir_region_france {
3.   my $dbh = shift;
4.
5.   my @les_regions_france;
6.   my $prep = $dbh->prepare('SELECT ncc FROM Regions ORDER BY ncc') or die $dbh->errstr;
7.   $prep->execute() or die "Echec requête\n";
8.   while ( my ($region) = $prep->fetchrow_array ) {
9.     push( @les_regions_france, $region );
10.  }
11.  $prep->finish();
12.  print "\n";
13.
14.  return @les_regions_france;
15. }
```

Cette procédure récupère en argument l'objet DBI et utilise les méthodes classiques pour faire un **SELECT**. Il s'agit des méthodes **prepare**, **execute**, **fetchXXX_YYY** et **finish**.
Notez qu'il est possible d'utiliser d'autres méthodes qui permettent d'écrire moins de code, nous en parlerons plus tard.

- la méthode **prepare** permet à DBI de préparer la requête SQL pour une exécution ultérieure et nous retourne un descripteur d'instruction que nous utiliserons par la suite (ligne 6) ;
- la méthode **execute** effectue le traitement nécessaire pour exécuter l'instruction préparée ;
- la méthode **fetchrow_array** récupère une ligne de données et retourne une liste contenant les valeurs de champ (ligne 8).

Pense-bête de **fetchrow_array** :

- **fetch** => récupérer ;
- **row** => ligne ;
- **array** => tableau.

Dans notre cas, nous nous attendons à plusieurs lignes de résultats (plusieurs régions), d'où l'utilisation de la boucle **while**. Tant qu'il y aura une ligne de résultat, Perl restera dans le while.
Notre requête SQL ne sélectionne qu'un seul champ (ncc), donc notre tableau n'aura qu'une seule case.

Ligne 8
```
while ( my ($region) = $prep->fetchrow_array ) {
```

Les régions sont stockées dans la liste **@les_regions_france**, ensuite, nous fermons la requête grâce à la méthode **finish**.
Maintenant que nous avons la liste des régions de France, créons une interaction entre le programme et nous par l'intermédiaire de l'utilisation du module **Term::UI**. Nous aurions pu utiliser **<STDIN>** mais le module nous facilite la vie. Voici le code :

```
#=========================================
# Lister tous les départements d'une région
#=========================================
# 1 - Récupération de toutes les régions de France
my @regions_france = obtenir_region_france($dbh);

# 2 □ Choix de l'utilisateur
my $term               = Term::ReadLine->new('brand');
my $nom_region_choisie = $term->get_reply(
  print_me => "1- Listons tous les départements d'une région",
  prompt   => 'Choisissez une région de France : ',
  choices  => \@regions_france,
  default  => 'PICARDIE',
);

# 3 - Récupération de tous les départements - méthode fetchrow_hashref
afficher_departement_dune_region( $dbh, $nom_region_choisie );
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

La méthode **get_reply** du module **Term::ReadLine** nous permet de poser une question dans la console en proposant une liste de choix (nos régions de France) grâce à l'option **choices** et en donnant une sélection par défaut (**default**). Nous obtenons ceci :

```
1- Listons tous les départements d'une région
  1> ALSACE
  2> AQUITAINE
  3> AUVERGNE
  4> BASSE-NORMANDIE
  5> BOURGOGNE
  6> BRETAGNE
  7> CENTRE
  8> CHAMPAGNE-ARDENNE
  9> CORSE
 10> FRANCHE-COMTE
 11> GUADELOUPE
 12> GUYANE
 13> HAUTE-NORMANDIE
 14> ILE-DE-FRANCE
 15> LA REUNION
 16> LANGUEDOC-ROUSSILLON
 17> LIMOUSIN
 18> LORRAINE
 19> MARTINIQUE
 20> MIDI-PYRENEES
 21> NORD-PAS-DE-CALAIS
 22> PAYS DE LA LOIRE
 23> PICARDIE
 24> POITOU-CHARENTES
 25> PROVENCE-ALPES-COTE D'AZUR
 26> RHONE-ALPES


Choisissez une région de France :  [23]:
```

Une fois notre choix effectué, nous utiliserons cette région pour lister les départements à partir de la procédure **afficher_departement_dune_region** suivante :

```
1. # Procédure pour afficher les départements d'une région
2. sub afficher_departement_dune_region {
3.   my ( $dbh, $nom_region_choisie ) = @_;
4.
5.   my $requete_departement_cheflieu = <<"SQL";
6.     SELECT Departements.nccenr as NomDep, Departements.cheflieu as cheflieu
7.     FROM Departements
8.     INNER JOIN Regions ON Regions.id_region = Departements.id_region
9.     WHERE Regions.ncc = "$nom_region_choisie"
10. SQL
11.   my $prep = $dbh->prepare($requete_departement_cheflieu) or die $dbh->errstr;
12.   $prep->execute() or die "Echec requête : $requete_departement_cheflieu\n";
13.
14.   print "La région $nom_region_choisie possède ", $prep->rows, " département(s)\n";
15.   while ( my $ref_donnees = $prep->fetchrow_hashref ) {
16.     print "\t- $ref_donnees->{NomDep}, chef-lieu $ref_donnees->{cheflieu}\n";
17.   }
18.   $prep->finish();
19.   print "\n";
20.   return;
21. }
```

Cette procédure nous permet d'utiliser **fetchrow_hashref** à la place de **fetchrow_array**

Pense-bête :

- **fetch** => récupérer ;
- **row** => ligne ;
- **hashref** => référence de hash.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

Ligne 15
```
while ( my $ref_donnees = $prep->fetchrow_hashref ) {
```

Chaque ligne de résultat est retournée sous forme de référence de hash au lieu de liste comme précédemment. Pour accéder à la valeur de chaque champ, il suffit de passer à notre référence la clé qui correspondant au nom de chaque champ spécifié dans notre requête SQL (**NomDep** et **cheflieu**) :

```
SELECT Departements.nccenr as NomDep, Departements.cheflieu as cheflieu
FROM Departements
INNER JOIN Regions ON Regions.id_region = Departements.id_region
WHERE Regions.ncc = "$nom_region_choisie"
```

D'où en ligne 16 : **$ref_donnees->{NomDep}** et **$ref_donnees->{cheflieu}**.
Si vous avez des difficultés à comprendre la notion de référence, lisez la FAQ , section : **Les références**.
En choisissant la région ILE-DE-FRANCE (14), nous obtenons :

Départements d'ile de France
```
Choisissez une région de France :  [23]: 14
La région ILE-DE-FRANCE possède 8 département(s)
        - Paris, chef-lieu 75056
        - Seine-et-Marne, chef-lieu 77288
        - Yvelines, chef-lieu 78646
        - Essonne, chef-lieu 91228
        - Hauts-de-Seine, chef-lieu 92050
        - Seine-Saint-Denis, chef-lieu 93008
        - Val-de-Marne, chef-lieu 94028
        - Val-d'Oise, chef-lieu 95500
```

Ouf, la première question de notre exercice est terminée ! J'espère que vous êtes encore en forme pour continuer  !

b - Afficher le nom du département, de la région et le nombre de communes en fonction d'un numéro de département

Maintenant, nous n'utiliserons plus les méthodes **fetchXXX_YYY**, mais les méthodes **selectXXX_YYY**. Pour ce faire, procédons comme ci-dessus en créant une interaction et une procédure pour cet affichage (**afficher_departement_dune_region**).

```
# 4- Afficher le nom du département, de la région et le nombre de
# communes en fonction d'un numéro de département - méthode selectrow_array,
my $numero_departement = $term->get_reply(
  print_me =>

  "2- Trouvons le nom d'un département, de la région et le nombre de communes à partir d'un numéro de département",
  prompt  => 'Donnez un numéro de département : ',
  default => 75,
);

afficher_informations_dun_departement( $dbh, $numero_departement );
```

```
1. # Procédure pour afficher toutes les informations d'un numéro de département
2. sub afficher_informations_dun_departement {
3.   my ( $dbh, $numero_departement ) = @_;
4.
5.   my $requete_dep_cheflieu_region = <<"SQL";
6.   SELECT
7.        Departements.nccenr   AS NomDep,
8.        Departements.cheflieu AS cheflieu,
9.        Regions.ncc           AS NomRegion
10.   FROM Departements
11.   INNER JOIN Regions ON Regions.id_region = Departements.id_region
12.   WHERE Departements.id_departement = '$numero_departement'
13. SQL
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

```
14.   my $requete_nombre_commune = <<"SQL";
15.     SELECT COUNT( id_communes ) as NbrCommunes
16.     FROM communes
17.     WHERE id_departement = "$numero_departement"
18. SQL
19.   my ( $nom_departement, $nom_cheflieu, $nom_region
 ) = $dbh->selectrow_array($requete_dep_cheflieu_region);
20.
21.   if ( defined $nom_cheflieu ) {
22.     print "\t - Numéro du département : $numero_departement\n";
23.     print "\t - Nom du département : $nom_departement\n";
24.     print "\t - Nom du chef-lieu : $nom_cheflieu\n";
25.     print "\t - Nom de la région : $nom_region\n";
26.
27.     # Calculer le nombre de communes
28.     my ($ref_commune) = $dbh->selectrow_hashref($requete_nombre_commune);
29.     print "\t - Nombre de communes : ", $ref_commune->{NbrCommunes}, "\n\n";
30.   }
31.   else {
32.     print "Numéro de département inconnu\n";
33.   }
34.
35.   return;
36. }
```

Dans le code ci-dessus, nous utilisons les méthodes **selectrow_array** et **selectrow_hashref** pour récupérer le nom de la région, du département et le nombre de communes pour le numéro de département saisi par l'utilisateur.

Ah !! Je vais donc vous expliquer comment écrire moins de code  !

Pense-bête :

- **select** => sélectionne ;
- **row** => ligne ;
- **hashref** => référence de hash ;
- **array** => tableau.

En fait, les méthodes de type **selectXXX_YYY** combinent les méthodes **prepare**, **execute** et **fetchXXX_YYY**.

select	fetch
selectrow_array	combine **prepare**, **execute** et **fetchrow_array**
selectrow_arrayref	combine **prepare**, **execute** et **fetchrow_arrayref**
selectrow_hashref	combine **prepare**, **execute** et **fetchrow_hashref**

- **Avantage :**

La combinaison de trois méthodes permet de densifier le code. Par exemple :

ligne 19
```
my ( $nom_departement, $nom_cheflieu, $nom_region
 ) = $dbh->selectrow_array($requete_dep_cheflieu_region);
```

Ici, nous récupérons un tableau contenant trois cases en une seule ligne, via la méthode **selectrow_array**. Alors qu'ici :

ligne 28
```
my ($ref_commune) = $dbh->selectrow_hashref($requete_nombre_commune);
```

nous récupérons une référence de hash grâce à la méthode **selectrow_hashref**.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

Je vous conseille d'utiliser les méthodes **selectrow_YYY** lorsque vous souhaitez récupérer une seule ligne de résultat.

Exemple de résultat :
```
Donnez un numéro de département :  [75]: 48
         - Numéro du département : 48
         - Nom du département : Lozère
         - Nom du chef-lieu : 48095
         - Nom de la région : LANGUEDOC-ROUSSILLON
         - Nombre de communes : 185
```

c - Trouver le nom du département et de la région d'une commune donnée

Maintenant, nous utiliserons la méthode **selectall_YYY**. Pour ce faire, procédons comme ci-dessus en créant une interaction et une procédure pour cet affichage (**afficher_informations_dune_commune**).

```perl
# 5- Trouver le nom du département et de la région d'une commune donnée
my $commune = $term->get_reply(
  print_me => "3- Trouvons le nom du département et de la région d'une commune",
  prompt   => "Nom d'une commune : ",
  default  => 'glandage',
);

# Les noms des communes (ncc) sont en majuscules dans la base
$commune = uc $commune;
# Les espaces dans les noms des communes sont remplacés par un tiret dans la base
$commune =~ s{\s}{-}g;
afficher_informations_dune_commune( $dbh, $commune );

# Déconnexion de la base de données
$dbh->disconnect();
```

```perl
# Procédure pour afficher toutes les informations d'une commune
sub afficher_informations_dune_commune {
  my ( $dbh, $commune ) = @_;

  my $requete_commune = <<"SQL";
    SELECT
         Departements.id_departement    AS id_departement,
         Departements.nccenr            AS NomDep,
         Departements.cheflieu          AS cheflieu,
         Regions.ncc                    AS NomRegion,
         Communes.ncc                   AS commune
    FROM Communes
    INNER JOIN Regions ON Regions.id_region = Communes.id_region
    INNER JOIN Departements ON Departements.id_departement = Communes.id_departement
    WHERE Communes.ncc LIKE "%$commune%"
SQL

  my $hash_ref_donnee = $dbh->selectall_hashref($requete_commune, 'id_departement');
  foreach my $id_dep ( sort { $a <=> $b } keys %{$hash_ref_donnee} ) {
    print "\t - Nom de la commune : $hash_ref_donnee->{$id_dep}{commune}\n";
    print "\t - Nom du département : $hash_ref_donnee->{$id_dep}{NomDep}\n";
    print "\t - Nom du chef-lieu : $hash_ref_donnee->{$id_dep}{cheflieu}\n";
    print "\t - Nom de la région : $hash_ref_donnee->{$id_dep}{NomRegion}\n\n\n";
  }

  return;
}
```

Dans le code ci-dessus, nous utilisons la méthode **selectall_hashref** pour récupérer le nom du département, de la région, le chef-lieu pour une commune saisie par l'utilisateur.

Pense-bête :

- **select** => sélectionne ;

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

- **all** => toutes les lignes ;
- **hashref** => référence de hash ;

En fait, les méthodes de type **selectall_YYY** combinent les méthodes **prepare**, **execute** et **fetchall_YYY**.

select	fetch
selectall_arrayref	combine **prepare**, **execute** et **fetchall_arrayref**
selectall_hashref	combine **prepare**, **execute** et **fetchall_hashref**

select_all_hashref renvoie une référence de hash (hashref) qui contient toutes les lignes de résultats. Chaque ligne est identifiable par une clé correspondant à la valeur du champ que nous avons spécifié (généralement un champ dont les valeurs sont uniques) et en valeur une référence de hash avec pour clé le nom du champ et en valeur la donnée.

```
my $hash_ref_donnee = $dbh->selectall_hashref($requete_commune, 'id_departement');
```

Ici, nous précisons que la clé qui identifiera chaque ligne de résultat correspondra à la valeur de l'identifiant de département (nous savons qu'elle est unique à chaque ligne de résultat dans notre table). Le choix de la clé est important sous peine d'écraser les informations dans le hash et avoir une incohérence de résultat. Nous choisissons également **id_departement** car nous l'avons sélectionné dans notre requête SQL. Donc faites attention !!
Ensuite, il faut parcourir ce hash pour afficher les informations.

```
my $hash_ref_donnee = $dbh->selectall_hashref($requete_commune, 'id_departement');
foreach my $id_dep ( sort { $a <=> $b } keys %{$hash_ref_donnee} ) {
  print "\t - Nom de la commune : $hash_ref_donnee->{$id_dep}{commune}\n";
  print "\t - Nom du département : $hash_ref_donnee->{$id_dep}{NomDep}\n";
  print "\t - Nom du chef-lieu : $hash_ref_donnee->{$id_dep}{cheflieu}\n";
  print "\t - Nom de la région : $hash_ref_donnee->{$id_dep}{NomRegion}\n\n\n";
}
```

Avantage :

- c'est donc une façon élégante de récupérer les résultats d'une requête ;
- en terme de lisibilité et de maintenance, on y gagne grandement ;
- il n'est pas nécessaire d'utiliser la méthode **finish**, DBI le gérera pour nous.

Inconvénient :

- il ne faut pas les utiliser si notre requête génère beaucoup de données, car tout sera en mémoire dans le hash ou array ;
- **selectall_hashref** permet d'avoir un champ à utiliser comme clé. Il faut donc bien vérifier que les valeurs de ce champ sont uniques pour éviter de mauvaises surprises ;
- il faut bien maîtriser l'utilisation des références pour s'y retrouver. En soi, ce n'est pas vraiment un inconvénient car c'est plutôt utile en Perl !!

Exemple de résultat :

```
Nom d'une commune :  [glandage]:
         - Nom de la commune : GLANDAGE
         - Nom du département : Drôme
         - Nom du chef-lieu : 26362
         - Nom de la région : RHONE-ALPES
```

3-4 - Téléchargement des programmes

Vous trouverez **ici** les deux programmes et trois fichiers de l'INSEE pour résoudre l'exercice.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/

C - Liens

Liens utiles pour compléter vos connaissances :

- module **DBI** ;
- **site officiel de Perl DBI** ;
- des **cours SQL** ;
- des **cours SGBD** ;
- **MySQL** ;
- **PostgreSQL** ;
- **EasyPHP** ;
- **installation des modules Perl** ;
- **INSEE** ;
- **Term::UI** ;
- **Les références**.

D - Conclusion

Avant toute chose, pensez à écrire correctement vos requêtes SQL ! Testez-les en console ou via les logiciels comme *phpMyAdmin* ou *phpPgAdmin*. Choisissez correctement vos méthodes DBI. Lisez la documentation CPAN de **DBI**, vous y trouverez d'autres méthodes intéressantes comme **COMMIT**. Cette dernière permet de protéger vos transferts de données en cas d'arrêt brutal du serveur pour une raison quelconque.
Profitez des cours **SQL** et **Perl** qui sont sur ce site et n'oubliez pas qu'il y a aussi les forums pour vous aider !
N.B. : Ces scripts ont été testés sous MySQL et sous PostgreSQL.
En ce qui concerne Oracle, je ne sais pas s'il y a des modifications à prévoir dans les codes Perl en dehors de la connexion à la base de données. Faites-moi un retour si nécessaire.
J'espère que cet article vous a aidé à comprendre l'utilisation du module DBI pour interagir avec les bases de données en Perl.
N'hésitez pas à faire des remarques, corrections ou appréciations.

E - Remerciements

Je remercie l'équipe Perl et SQL de dvp.com pour les remarques et la relecture de ce tutoriel, notamment **ClaudeLELOUP** et **stoyak**.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011djibril. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://djibril.developpez.com/tutoriels/perl/perl-dbi/