MySQL 10

PLAN

OBJECTIFS

- Maîtriser l'accès aux bases de données via un programme PHP et l'outil phpMyAdmin.

10.1 Prérequis

Cette section présente l'accès à une base de données MySQL *via* des programmes PHP. Elle ne présente pas les requêtes SQL, *il est donc important que le lecteur maîtrise déjà ce langage*.

10.2 PHP et les bases de données

Principe

Le langage PHP est un moyen efficace pour accéder aux bases de données et en particulier à MySQL. Les sites web ainsi créés sont dynamiques et accèdent aux données stockées dans des bases. La figure 1.12 présentée au chapitre 1 résume cette architecture.

Outils

Le langage PHP avec son environnement propose les outils suivants :

- **phpMyAdmin** : cette interface de gestion de bases de données MySQL est installée en standard avec les paquetages tels que XAMPP ;
- des **fonctions** ou **méthodes** d'accès aux bases de données : il en existe trois familles :
  - les fonctions `mysql_` : leur nom commence par `mysql_`. Elles sont considérées comme obsolètes depuis PHP 5.5.0 ;

- les fonctions `mysqli_` : leur nom commence par `mysqli_`. Elles proposent de nombreuses fonctionnalités. Leur syntaxe est procédurale ou objet ;
- **PDO** (PHP Data Objects*)*. c'est un ensemble complet de méthodes objets donnant accès à n'importe quel SGBDR tel que MySQL, PostgreSQL ou Oracle. Ce chapitre présente PDO.

10.3 phpMyAdmin

Présentation

phpMyAdmin est installé en standard avec XAMPP. C'est un ensemble de programmes PHP facilitant la gestion de bases de données MySQL.

Lors de l'installation, le mot de passe de l'administrateur « root » n'est pas renseigné, ni pour phpMyAdmin, ni pour MySQL. Cela ne pose pas de problème pour une utilisation locale, mais dans les autres cas il est impératif de corriger cette faille de sécurité et de définir un mot de passe (*cf.* Documentation d'installation téléchargeable).

URL d'accès

Si le mot de passe est identique entre MySQL et phpMyAdmin (fichier de configuration `phpmyadmin/config.inc.php`), l'accès à phpMyAdmin est obtenu *via* l'URL : `http://localhost/phpmyadmin`. La figure 10.1 présente cet écran.

Figure 10.1 – phpMyAdmin : écran d'accueil.

Notion de base de données, de table et d'enregistrement

Une *base de données* est une structure contenant des *tables*, chaque table contient des *enregistrements*, chaque enregistrement est constitué de *champs*. Pour comprendre cette organisation, prenons l'exemple d'une **base du personnel**.

La **première table** contient les *profils de poste* : secrétaire, technicien, directeur… Pour chaque poste (enregistrement) sont indiquées les informations (champs)

comme : un identifiant unique (1, 2, 3…), le nom du profil (SECRETAIRE, DIRECTEUR…), un niveau de rémunération (100, 350…), etc.

La **seconde table** contient le *personnel* : chaque enregistrement contient les informations d'une personne : un identifiant unique (1, 2, 3…), son nom (MARTIN, DUPONT…), son prénom (PIERRE, JEAN…), son poste (identifiant d'un profil de poste = 3, 2…), sa date de naissance, sa ville de naissance, son adresse…

La base de données correspond à un conteneur de tables, chaque table contient un ensemble « cohérent » d'information : les profils de poste ou le personnel.

Chaque enregistrement possède une clef unique, son identifiant. Cette clef est utilisable dans toutes les tables. Ainsi, dans la table des personnels, pour chaque personne on fait référence à son profil de poste *via* cet identifiant, et non en y reportant les informations. La figure 10.2 présente cette architecture.

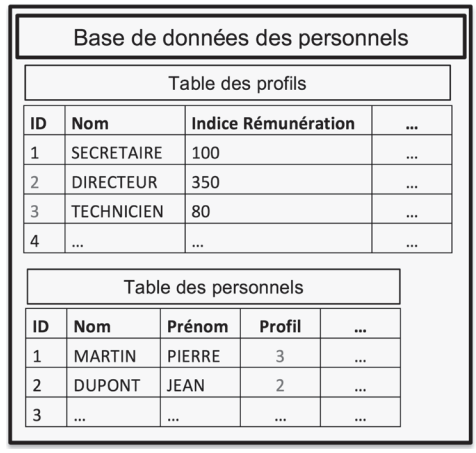

Figure 10.2 – **Architecture d'une base de données.**

Gestion d'une base de données

Cette section présente la création, la suppression, l'exportation et l'importation d'une base de données *via* l'outil phpMyAdmin.

Création

Pour créer la base de données « CoursPHP », cliquez sur l'onglet « Bases de données » puis sur « Nouvelle base de données ».

Indiquez le nom « CoursPHP » et l'interclassement « utf8_general_ci » (1).

Cliquez sur le bouton « Créer ». Un écran de confirmation apparaît (2) (figure 10.3). Les bases « information_schema », « mysql », et « performance_schema » sont propres à MySQL.

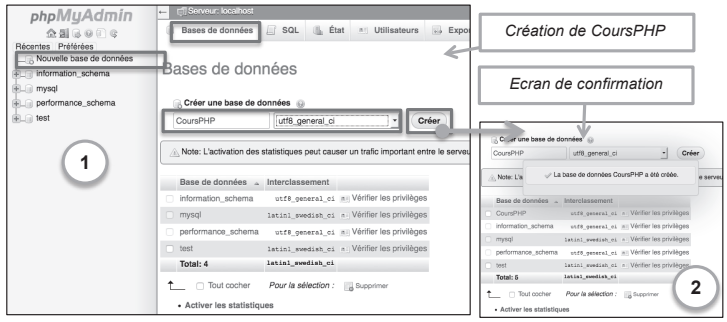

Figure 10.3 – phpMyAdmin : création d'une base de données.

Suppression

Pour supprimer la base de données « CoursPHP », cliquez sur l'onglet « Bases de données ». Sélectionnez la base à supprimer, puis cliquez sur « Supprimer ». Confirmez la suppression dans l'écran suivant. Un écran de confirmation apparaît.

Exportation

phpMyAdmin permet la sauvegarde d'une base de données, avec toutes ses tables. Dans l'exemple suivant, la base de données « CoursPHP » contient plusieurs tables. Sélectionnez la base de données à sauvegarder, puis cliquez sur l'onglet « Exporter », puis sur « Exécuter » (1). Sur l'écran suivant, cliquez sur « OK » (2). La sauvegarde est un fichier `CoursPHP.sql` qui doit être téléchargé (3) (figure 10.4).

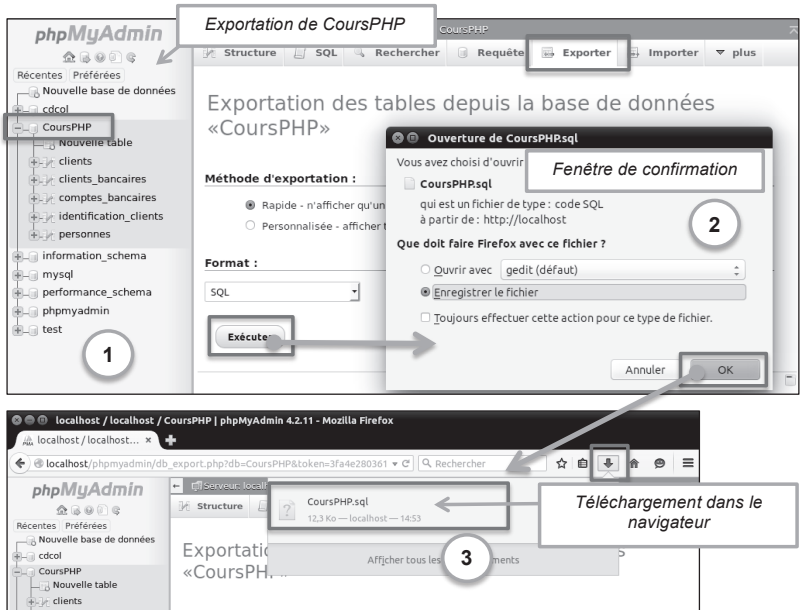

Figure 10.4 – phpMyAdmin : exportation d'une base de données.

Importation

Cliquez sur la base de données à restaurer « CoursPHP » (elle doit exister et être vide), puis sur l'onglet « Importer » et sur « Parcourir ». Dans la fenêtre qui apparaît, sélectionnez le fichier de restauration, et cliquez sur « Ouvrir ».

Le fichier sélectionné apparaît à droite de « Parcourir ». Cliquez sur « Exécuter ». Les tables apparaissent dans la base de données « CoursPHP ».

Gestion d'une table

Cette section présente la création, la suppression, l'affichage, la modification et le vidage d'une table *via* l'outil phpMyAdmin.

Création

- *Présentation*

Pour créer la table « personnes » dans la base de données « CoursPHP », cliquez sur la base « CoursPHP » (1). Dans la nouvelle fenêtre, entrez le nom de la table à créer, par exemple « personnes », indiquez le nombre de colonnes, par exemple 4 (2) (figure 10.5), puis cliquez sur « Exécuter ».

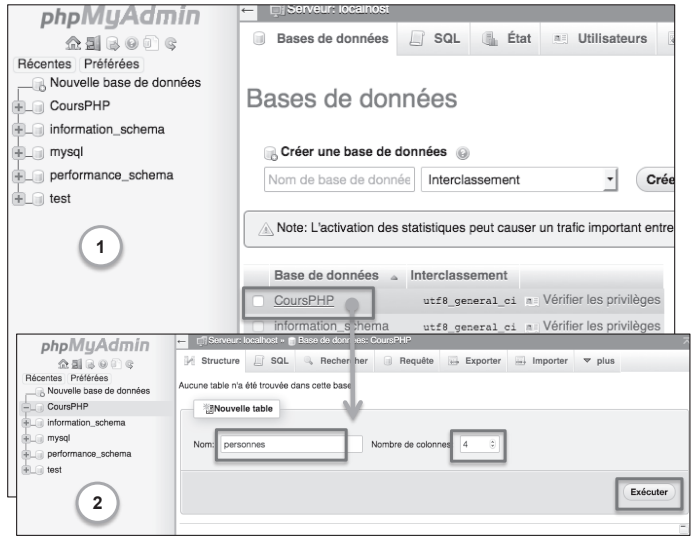

Figure 10.5 – phpMyAdmin : création d'une table-étape1.

Pour chaque colonne ou champ, indiquez :

- son *nom* : par exemple **ID, Nom, Prenom** ou **Age** ;
- son *type* : par exemple **INT** (pour ID et Age), **VARCHAR** (Nom et Prenom) ;
- sa *taille* : par exemple **255** (pour VARCHAR), rien pour ID et Age ;

- *l'interclassement* : pour le codage des caractères. Par exemple **utf8_general_ci** qui indique « Unicode multilingue Insensible à la casse ».
- *Index* : si c'est un index Primaire (clef), par exemple pour **ID**, sélectionnez **PRIMARY.** Ceci indique que la valeur de la clef est unique. PRIMARY ne peut être positionné que sur un seul champ, contrairement à UNIQUE, qui indique aussi que chaque valeur de ce champ est unique, mais cet attribut peut être positionné sur plusieurs champs.
- *A_I* pour **Auto_incrément**, si on veut une numérotation automatique séquentielle des enregistrements. Cochez cette case pour le champ **ID**.

  Cliquez sur « Sauvegarder », la table « personnes » est créée (2) (figure 10.6).

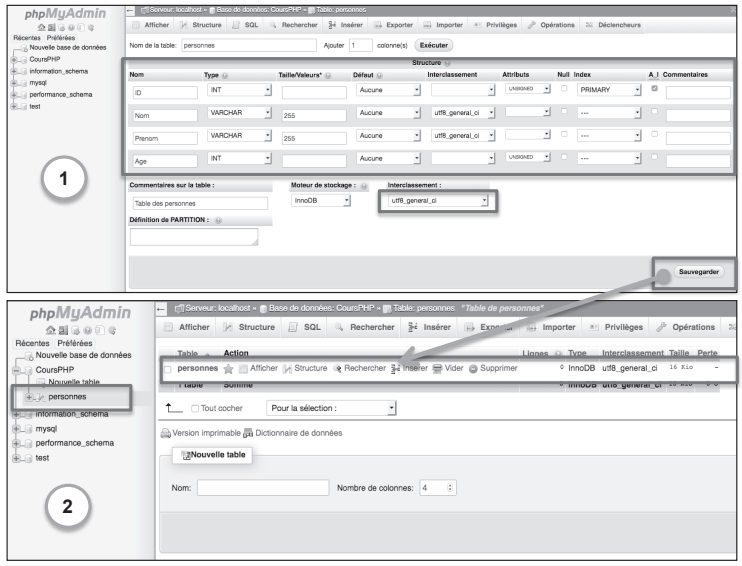

Figure 10.6 – phpMyAdmin : création d'une table-étape2.

- *Le moteur de stockage*

Dans les écrans précédents, le moteur de stockage utilisé par défaut est : **InnoDB**. Le moteur de stockage est l'ensemble des méthodes ou algorithmes mis en œuvre pour stocker les données et y accéder au moyen de requêtes SQL. Il est choisi lors de la création de la table par l'administrateur. Son choix impacte directement *l'efficacité des requêtes SQL* ou *l'activation de certaines possibilités comme les transactions*. Avant MySQL 5.5, le moteur par défaut était MylSAM, depuis la version 5.5, le moteur par défaut est **InnoDB**.

Affichage ou modification

Pour afficher la structure de la table, et éventuellement la modifier, cliquez sur la table « personnes », puis cliquez sur l'onglet « Structure ». La structure apparaît.

On peut modifier, supprimer des champs ou changer leurs attributs en cliquant sur le lien adéquat du champ. Il est également possible d'ajouter des colonnes (champs), en début ou fin de table, ou après un champ. Dans ce cas d'ajout d'un champ, il faut renseigner la valeur de ce nouveau champ pour tous les enregistrements déjà présents dans la table.

Suppression complète de la table

Pour supprimer la table « personnes » dans la base « CoursPHP », cochez la table « personnes », puis cliquez sur l'action « Supprimer ». Un message d'alerte apparaît. Cliquez sur « OK », un message confirme la suppression.

Vider la table de ses données

Pour vider la table « personnes » dans la base « CoursPHP », cochez la table « personnes », puis cliquez sur l'action « Vider ». Un message d'alerte apparaît, cliquez sur « OK ». Un message confirmant la suppression des données apparaît.

Gestion des données

Cette section présente la *mise à jour des données* (enregistrements) d'une *table*.

Insertion de données

La première méthode (1) pour insérer des données dans une table consiste à cliquer sur le nom de la base de données « CoursPHP », puis à cliquer sur le lien « insérer » de la table « personnes ». La seconde méthode (2) consiste à cliquer sur le nom de la table « personnes », puis à cliquer sur l'onglet « insérer » du menu (figure 10.7).

Figure 10.7 – phpMyAdmin : insertion de données dans une table-1.

L'écran d'insertion des données apparaît (1). Par défaut, il autorise la saisie de deux enregistrements. Renseignez la « valeur » des champs **Nom**, **Prenom** et **Age**. N'indiquez rien pour **ID** (auto-incrémenté à partir de 1), puis cliquez sur « Exécuter ». L'écran suivant (2) indique que deux personnes ont été insérées dans la table « personnes » ; la requête SQL ayant permis de faire cette insertion apparaître au centre. Un clic sur le lien « [Créer source PHP] » affiche le code source (3) de la syntaxe PHP qui exécute cette requête SQL (figure 10.8).

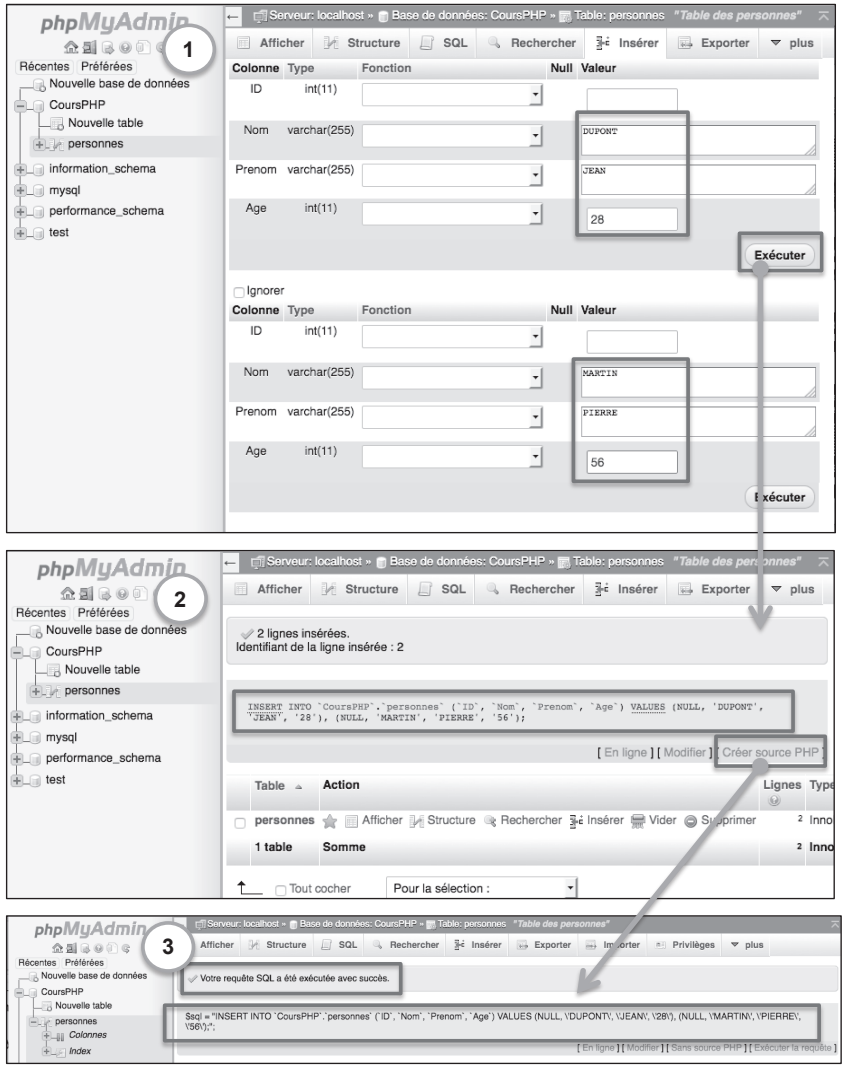

Figure 10.8 – phpMyAdmin : insertion de données dans une table-2.

Affichage

Pour afficher le contenu de la table, cliquez sur le nom de la table « personnes ». La liste des enregistrements apparaît sur la droite (figure 10.9).

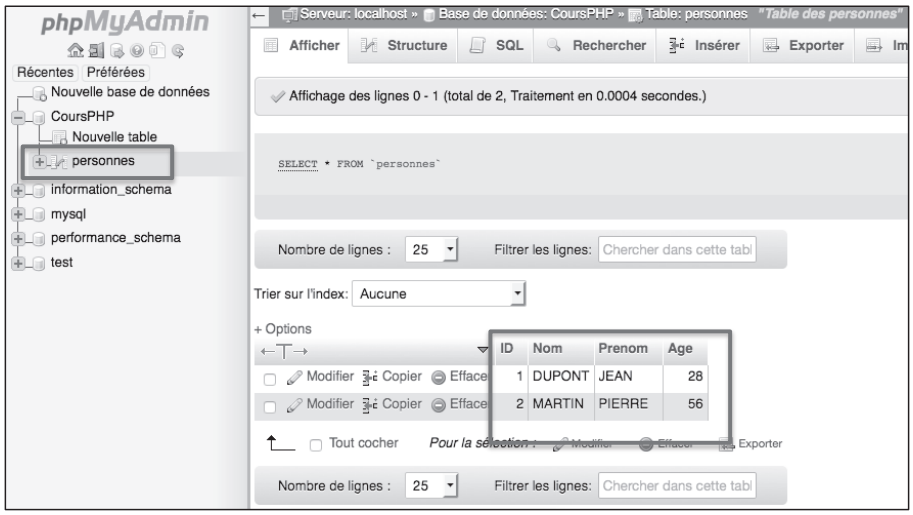

Figure 10.9 – phpMyAdmin : affichage des données d'une table.

Modification

Pour modifier un enregistrement, cliquez sur le crayon « Modifier » de l'enregistrement sur la page d'affichage. Sur l'écran de modification, saisissez les nouvelles valeurs. L'écran résultat montre la modification.

Suppression

Pour supprimer un enregistrement de la table, cliquez sur le nom de la table « personnes » ou sur l'onglet « Afficher ». Puis cliquez sur le lien « Effacer » de la ligne de l'enregistrement à supprimer. Un écran de confirmation apparaît.

Exportation

- *Les formats de fichier*

L'exportation des données d'une table peut se faire sous différents formats, comme « CSV pour Excel » ou « SQL ». Pour en voir la liste, sélectionnez la table « personnes » puis cliquez sur le lien « Exporter ». La liste déroulante « Format » présente tous les formats.

- *Fichier texte de requêtes SQL*

La figure 10.10 détaille l'exportation au *format SQL* de la table « personnes » *via* l'onglet « Exporter » (1). La sélection du bouton « Exécuter » fait apparaître une fenêtre de sauvegarde (2). Le fichier d'exportation se nomme `personnes.sql`, et sera placé dans le répertoire `Téléchargement` sous Linux. Le fichier texte `personnes.sql` contient les requêtes SQL permettant de recréer la table « personnes » et d'y insérer les données.

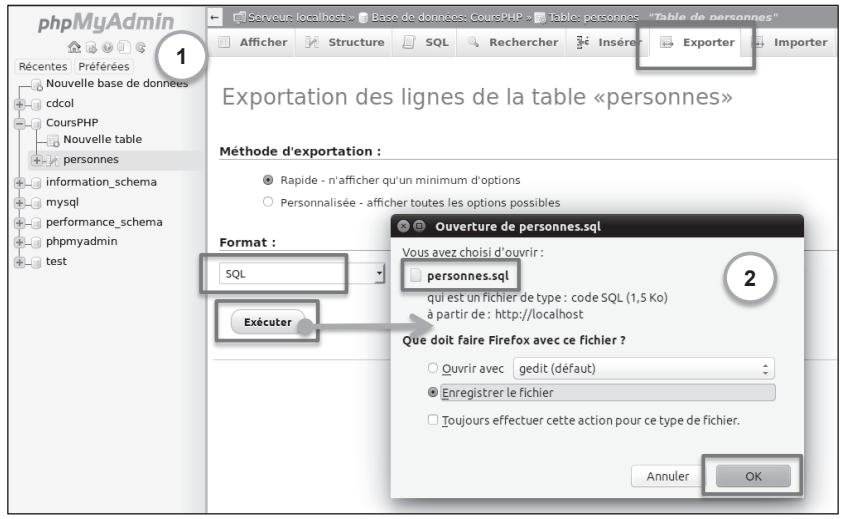

Figure 10.10 – phpMyAdmin : exportation d'une table au format SQL.

- *Fichier CSV pour Excel*

Pour exporter au format CSV pour Excel il suffit de sélectionner ce type de fichier dans le champ « Format ». Le fichier d'exportation se nomme `personnes.csv`.

Importation

Cette section présente l'importation de données (et de tables) à partir d'un fichier.

- *Fichier texte de requêtes SQL*

La figure 10.11 détaille l'importation au format SQL de la table « personnes » *via* l'onglet « Importer » (1). La sélection du bouton « Exécuter » fait apparaître la sélection du fichier (2). Le choix de `personnes.sql`, situé dans le répertoire `Téléchargement` puis de « Ouvrir » fait apparaître la fenêtre de confirmation (3).

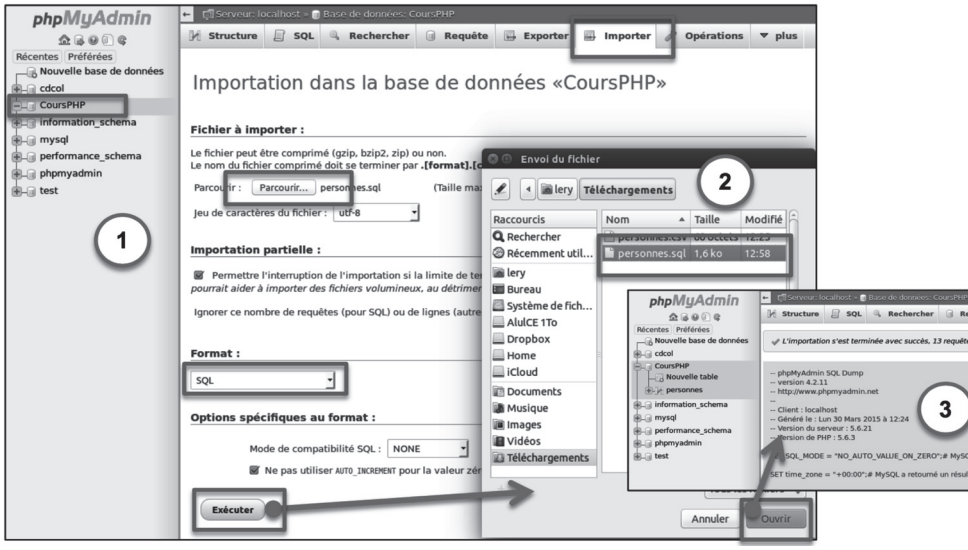

Figure 10.11 – **phpMyAdmin : importation d'une table au format SQL.**

- *Fichier CSV pour Excel*

Pour importer un fichier au format CSV pour Excel, sélectionner ce type dans le champ « Format » de la figure 10.11.

La figure 10.12 présente en exemple le fichier « personnes.csv » contenant quatre lignes.

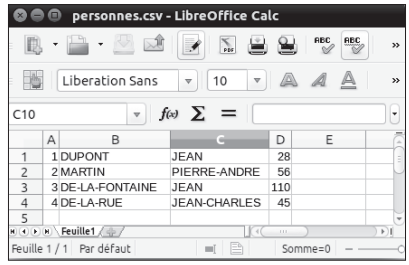

Figure 10.12 – **phpMyAdmin : importation d'une table au format CSV.**

Gestion des utilisateurs

Cette section présente la gestion des utilisateurs ou « comptes ». Elle détaille : l'affichage, la création, la suppression d'un compte utilisateur et l'affectation des privilèges qui définissent les droits sur les bases de données et les tables.

Principe

MySQL gère les utilisateurs *via* un *identifiant* qui est de la forme : **dupont@ordinateur.fr**. Celui-ci est composé :

- d'un login : **dupont**, dans notre exemple, l'utilisateur ;
- d'un poste de connexion : **ordinateur.fr** dans notre exemple. C'est l'adresse IP du poste de la personne. Il peut être « localhost » ou 127.0.0.1, si le poste de travail est le serveur MySQL. La syntaxe « **%** » indique n'importe quel poste.

  À cet *identifiant* on associe :

- un mot de passe ;
- des privilèges : ce sont les droits d'accès, de modification, d'insertion ou autres, sur les bases de données et/ou sur les tables.

Un même login peut avoir différents *privilèges selon le poste de connexion*. Il peut, par exemple, posséder plus de privilèges pour une connexion en local, et aucun lors d'un accès à partir d'un poste distant.

Affichage des utilisateurs existants

- *L'onglet « Utilisateur »*

Pour afficher la liste des utilisateurs ou « comptes », il suffit de cliquer sur l'onglet « Utilisateurs » (figure 10.13).

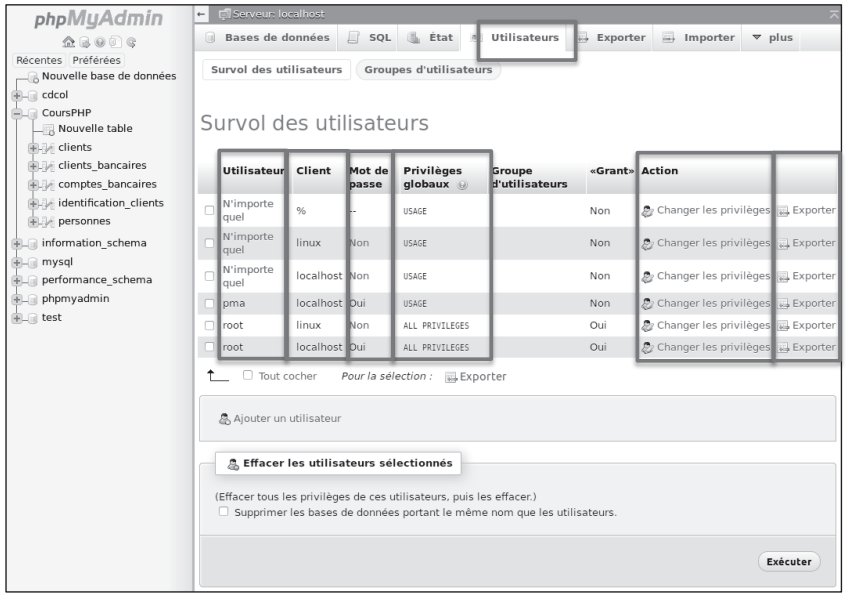

Figure 10.13 – phpMyAdmin : affichage des comptes utilisateurs.

L'écran fait apparaître différentes colonnes :

- la colonne « **Utilisateur** » donne le login ;
- la colonne « **Client** » indique le poste de connexion. La syntaxe « **%** » indique de n'importe quel client ;
- la colonne « **Mot de passe** » précise si un mot de passe est défini ;
- la colonne des « **Privilèges globaux** » précise les privilèges appliqués à toutes les bases de données. Le terme « USAGE » est l'absence de privilèges globaux.

- *L'utilisateur anonyme*

La figure 10.13 montre trois lignes dont la colonne « User » affiche « N'importe quel ». Il s'agit de l'utilisateur « anonyme » créé à l'installation de MySQL, tout comme la base de données « test ». Cet utilisateur représente n'importe quel utilisateur non enregistré. Il donne accès à la base « test » et à toutes les bases dont le nom commence par les cinq caractères « test_ ».

- *La table mysql.user*

Les utilisateurs sont gérés dans la table « mysql.user ». Pour en voir le détail technique, il faut sélectionner la base « mysql ». Puis faire défiler les tables vers le bas et cliquer sur la table « user ». En faisant défiler l'écran à droite, on voit l'ensemble des privilèges des utilisateurs. Voici quelques paramètres d'utilisateur :

- *les privilèges* : Select_priv, Insert_priv, Update_priv, Delete_priv, etc. ;
- *les valeurs maximales* : max_questions, max_updates, max_connections, max_user_connections ;
- *les paramètres d'authentification* : authentification_string, password_expired.

Création d'un compte utilisateur

Pour créer un nouvel utilisateur, sélectionnez l'onglet « Utilisateur » puis cliquez sur « Ajouter un utilisateur » (1) (figure 10.14).

Nous présentons la création de l'utilisateur « **personnesadm** », qui pourra gérer la table « personnes » de la base « CoursPHP ». Dans l'écran suivant (2), remplissez les champs : « Nom d'utilisateur », et « Mot de passe » (à saisir deux fois). Le champ « Client » contient l'adresse IP du poste de connexion à partir duquel cet utilisateur devra se connecter pour avoir ces privilèges. On peut laisser « % » pour indiquer une connexion possible à partir de tous les postes clients. Puis cliquez sur « Exécuter ». La liste des utilisateurs montre que cet utilisateur a été créé (3).

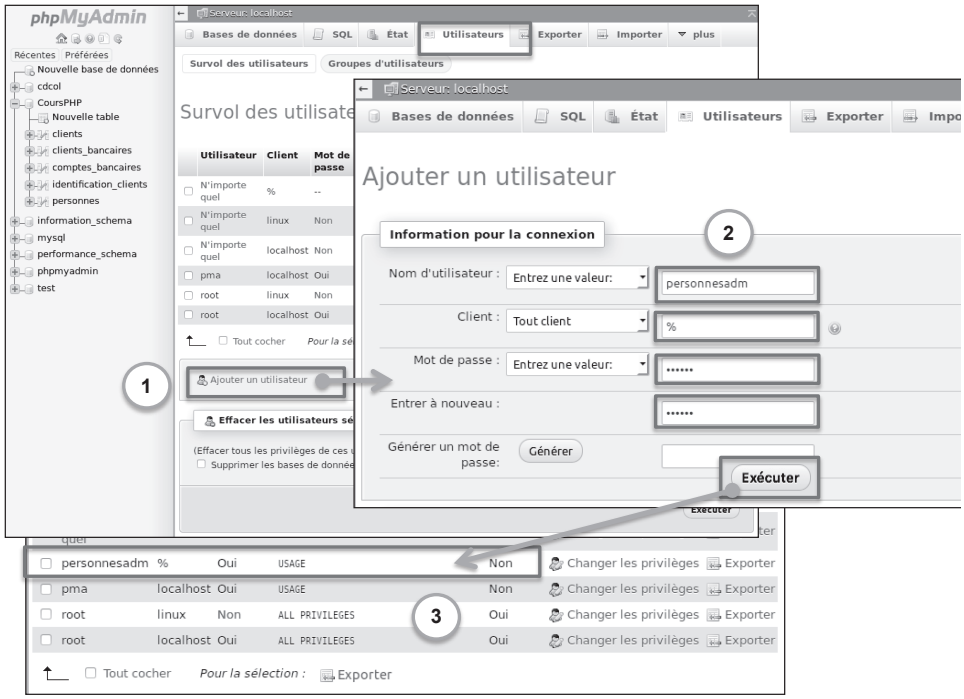

Figure 10.14 – **phpMyAdmin : création d'un utilisateur.**

Créons un nouvel utilisateur, identique au précédent, dont les droits seront différents lors d'une connexion à partir de « **localhost** ». Le processus est identique, le mot de passe peut être différent, et le champ « **Client** » doit indiquer « **localhost** ».

Gestion des privilèges

• *Présentation des privilèges*

MySQL attribue des privilèges selon différents contextes :

- **les privilèges administratifs** : ils s'appliquent aux opérations d'administration. Ils sont globaux, et non liés aux bases de données. C'est par exemple le privilège de gérer des utilisateurs ;
- **les privilèges sur les bases de données** : ils portent sur tous les objets d'une base de données. S'ils s'appliquent à toutes les bases, ils deviennent globaux ;
- **les privilèges sur les objets des bases de données** : ils s'appliquent à certains objets comme une table, une colonne, un index, une vue ou une procédure stockée. Cela peut s'appliquer à tous les objets d'un même type, par exemple toutes les tables d'une base de données.

La liste des privilèges proposés par MySQL est disponible à l'URL : *https://dev.mysql.com/doc/refman/5.5/en/privileges-provided.html*

Remarque

Le privilège « USAGE » n'a aucun privilège global. Le compte possède les droits sur la base « test » et sur les bases dont le nom commence par « test_ ». Il possède aussi le droit d'exécuter les transactions : SHOW GLOBAL VARIABLES, SHOW GLOBAL STATUS.

- *Ajout de privilèges*

L'ajout de privilèges correspond à la syntaxe SQL « **GRANT** ».

Pour le compte personnesadm@%

Les figures 10.15 et 10.16 présentent l'affectation du privilège de **consulter** la table « personnes » (**SELECT**) à l'utilisateur « personnesadm » depuis n'importe quel poste de travail « % ».

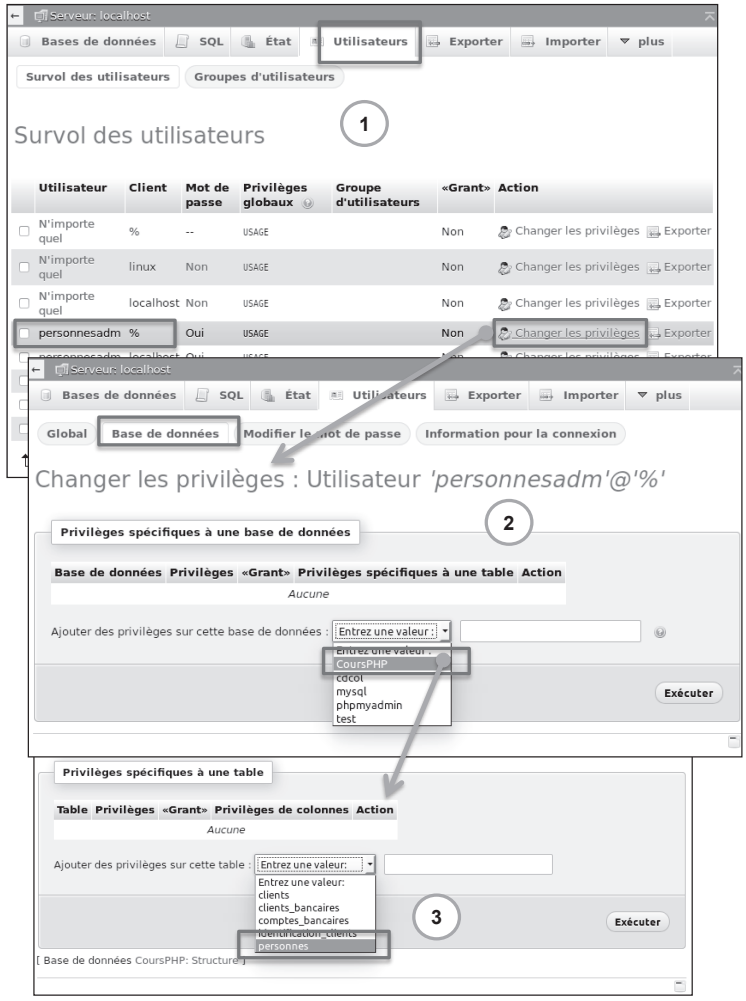

Figure 10.15 – phpMyAdmin : changer les privilèges d'un utilisateur-1.

Cliquez sur « Changer les privilèges » de « personnesadm@% » (1).

Sélectionnez « Base de données » puis « CoursPHP » (2).

En bas de l'écran suivant, sélectionnez la table « personnes » (3).

Dans la colonne « SELECT », sélectionnez « ID », « Nom », « Prenom » et « Age » (4), puis cliquez sur « Exécuter ». L'écran suivant confirme la modification (5) et affiche la syntaxe SQL « GRANT … ».

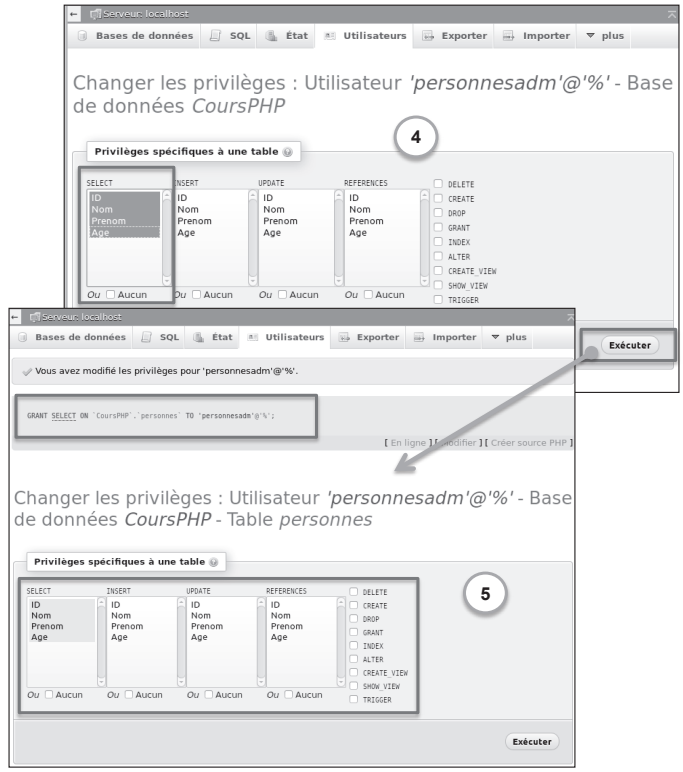

Figure 10.16 – **phpMyAdmin : changer les privilèges d'un utilisateur-2.**

Pour le compte personnesadm@localhost

De la même manière, affectons les privilèges de **consultation, d'ajout, de modification et de suppression** (**SELECT**, **INSERT**, **UPDATE**, **DELETE**) sur les données de cette table pour « personnesadm » quand il travaille depuis « localhost ».

Les étapes décrites dans la figure 10.16 sont identiques. Il faut sélectionner « ID », « Nom », « Prenom » et « Age » dans les colonnes « SELECT, INSERT, UPDATE » et cocher « DELETE », puis « Exécuter ».

- *Retrait de privilèges*

Le retrait de privilèges correspond à la syntaxe SQL « **REVOKE** ». Il suit le même processus. Il suffit de désélectionner un privilège et de cliquer sur « Exécuter ».

Gestion des paramètres de connexion

Chaque utilisateur possède des paramètres de connexion comme le *nombre maximal de connexions autorisées*, définis dans la table « mysql.user ».

- *Problématique*

La limitation du nombre maximal de connexions (par heure) peut provoquer des erreurs lorsque de multiples accès sont effectués *via* un même compte. Plusieurs solutions résolvent ce problème :

- revoir le programme PHP afin qu'une seule connexion soit maintenue ouverte durant la session de travail de l'utilisateur ;
- fermer la connexion à chaque fin de traitement. Ainsi, la connexion ne reste pas inutilement ouverte, mais cela peut pénaliser les performances du programme ;
- modifier le paramètre du nombre de connexions simultanées (par heure) autorisées pour l'utilisateur.

À l'inverse, si aucune limitation n'est définie, un nombre maximal de requêtes (par heure) incontrôlé aboutira à une saturation du serveur et à une dégradation de son temps de réponse.

- *Modification des paramètres*

Les différents paramètres de connexion sont :

- `MAX_QUERIES_PER_HOUR` : le nombre de requêtes envoyées au serveur, qu'un utilisateur peut exécuter par heure ;
- `MAX_UPDATES_PER_HOUR` : le nombre de commandes modifiant une table ou base de données, qu'un utilisateur peut exécuter par heure ;
- `MAX_CONNECTIONS_PER_HOUR` : le nombre de nouvelles connexions qu'un utilisateur peut démarrer, par heure ;
- `MAX_USER_CONNECTIONS` : le nombre de connexions simultanées pour un utilisateur.

La figure 10.17 présente la modification de ces paramètres.

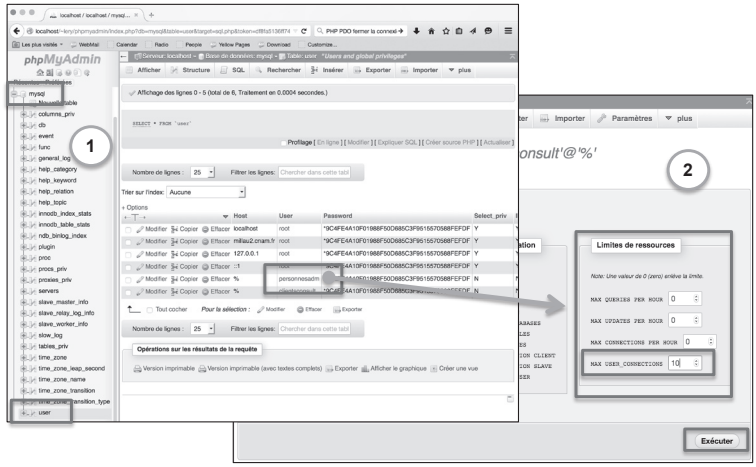

Figure 10.17 – phpMyAdmin : modification des paramètres de connexion.

Sélectionnez la base « mysql », puis la table « user ». Cliquez sur l'utilisateur à modifier (1). Dans l'écran suivant (2), modifiez les privilèges globaux en indiquant la valeur désirée dans la colonne « Limites de ressources », par exemple 10 pour « MAX USER_CONNECTIONS », puis cliquez sur le bouton « Exécuter »

Suppression d'un compte utilisateur

Pour supprimer un compte, cliquez sur l'onglet « Utilisateur », sélectionnez le compte, par exemple « personnesadm@% » et cochez la case « Supprimer les bases de données portant le même nom que les utilisateurs ». Cliquez sur « Exécuter ». Une fenêtre d'avertissement demande de confirmer la suppression.

10.4 PDO – PHP Data Objects

Présentation

PDO est un outil complet donnant accès à n'importe quel type de base de données, comme MySQL, PostgeSQL ou Oracle. Il présente une syntaxe objet et propose des classes et des méthodes prédéfinies. Une documentation complète est disponible à l'URL : *http://php.net/manual/fr/book.pdo.php.*

Les classes et les méthodes

PDO propose les classes et les méthodes suivantes :

- *La classe PDO* : les objets et les méthodes de cette classe portent sur la connexion entre le programme PHP et le serveur de base de donnée. Voici ses méthodes :
  - `PDO::beginTransaction()` : démarre une transaction ;

- `PDO::commit()` : valide une transaction ;
- `PDO::__construct()` : constructeur de l'objet ;
- `PDO::errorCode()` : retourne le code de l'erreur de la dernière opération sur la base ;
- `PDO::errorInfo()` : retourne les informations de l'erreur de la dernière opération ;
- `PDO::exec()` : exécute une requête SQL et retourne le nombre de lignes ;
- `PDO::getAttribute()` : récupère un attribut d'une connexion à une base ;
- `PDO::getAvailableDrivers()` : retourne la liste des pilotes PDO disponibles ;
- `PDO::inTransaction()` : vérifie si nous sommes dans une transaction ;
- `PDO::lastInsertId()` : retourne l'identifiant de la dernière ligne insérée ou la valeur d'une séquence ;
- `PDO::prepare()` : prépare une requête à l'exécution et retourne un objet PDOStatement ;
- `PDO::query()` : exécute une requête SQL, retourne un jeu de résultats en tant qu'objet PDOStatement ;
- `PDO::quote()` : protège une chaîne pour l'utiliser dans une requête SQL ;
- `PDO::rollBack()` : annule une transaction ;
- `PDO::setAttribute()` : configure un attribut ;

- *La classe PDOStatement* : les objets et les méthodes de cette classe traitent d'une requête et du jeu de résultat associé.
  - `PDOStatement::bindColumn` : lie une colonne à une variable PHP ;
  - `PDOStatement::bindParam` : lie un paramètre à un nom de variable ;
  - `PDOStatement::bindValue` : associe une valeur à un paramètre ;
  - `PDOStatement::closeCursor` : ferme le curseur, permettant à la requête d'être de nouveau exécutée ;
  - `PDOStatement::columnCount` : retourne le nombre de colonnes dans le jeu de résultats ;
  - `PDOStatement::debugDumpParams` : détaille une commande préparée ;
  - `PDOStatement::errorCode` : récupère le code de l'erreur associée à la dernière opération sur la requête ;
  - `PDOStatement::errorInfo` : récupère les informations sur l'erreur associée lors de la dernière opération sur la requête ;
  - `PDOStatement::execute` : exécute une requête préparée ;
  - `PDOStatement::fetch` : récupère la ligne suivante d'un jeu de résultats ;
  - `PDOStatement::fetchAll` : retourne un tableau contenant toutes les lignes du jeu de résultats ;

- `PDOStatement::fetchColumn` : retourne une colonne depuis la ligne suivante d'un jeu de résultats ;
- `PDOStatement::fetchObject` : récupère la prochaine ligne et la retourne en tant qu'objet ;
- `PDOStatement::getAttribute` : récupère un attribut de requête ;
- `PDOStatement::getColumnMeta` : retourne les métadonnées pour une colonne d'un jeu de résultats ;
- `PDOStatement::nextRowset` : avance à la prochaine ligne de résultats d'un gestionnaire de résultats multiples ;
- `PDOStatement::rowCount` : retourne le nombre de lignes affectées par le dernier appel à la fonction `PDOStatement::execute()` ;
- `PDOStatement::setAttribute` : définit un attribut de requête ;
- `PDOStatement::setFetchMode` : définit le mode de récupération par défaut pour cette requête ;

- *La classe PDOException* : les objets et méthodes de cette classe traitent des erreurs émises par PDO.

Les constantes

Le tableau 10.1 présente quelques constantes prédéfinies. La liste exhaustive est disponible à l'URL : http://php.net/manual/fr/pdo.constants.php.

Tableau 10.1 – Les constantes PDO

Constante	Signification
`PDO::PARAM_BOOL`	Type de données booléen
`PDO::PARAM_NULL`	Type de données NULL SQL
`PDO::PARAM_INT`	Type de données INTEGER SQL
`PDO::PARAM_STR`	Types de données CHAR, VARCHAR ou les autres types de données sous forme de chaîne de caractères SQL.
`PDO::FETCH_ASSOC`	La méthode de récupération retourne chaque ligne dans un tableau associatif, indexé par les noms des colonnes.
`PDO::FETCH_BOUND`	La méthode de récupération retourne TRUE et assigne les valeurs des colonnes dans les variables PHP auxquelles elles sont liées avec la méthode `bindParam()` ou `bindColumn()`
`PDO::FETCH_COLUMN`	La méthode de récupération retourne une seule colonne demandée depuis la prochaine ligne du jeu de résultats.
`PDO::ATTR_PERSISTENT`	Demande une connexion persistante.
`PDO::MYSQL_ATTR_INIT_COMMAND`	Commande à exécuter lors de la connexion au serveur MySQL. Sera automatiquement réexécuté si reconnexion.

Connexion et déconnexion à la base de données

Connecter une base de données

- *Ouvrir la connexion*

L'accès à une base de données nécessite d'établir une connexion. Pour cela, on crée une instance de la classe PDO en indiquant :

- le *type* de la base de données : mysql, oracle, PostgreSQL ;
- le *nom du serveur* : c'est l'adresse internet du serveur hébergeant le SGBD, par exemple `serv1.domaine.fr`. « localhost » indique que le serveur Apache, PHP et le SGBD sont hébergés sur le même serveur ;
- le *nom de la base de données* : c'est la base de données dans laquelle se trouvent les tables à accéder ;
- le *login* : il identifie l'accès au SGBDR ;
- le *mot de passe* : il authentifie l'accès au SGBDR.

Voici un exemple de syntaxe :

```
$bdd = new PDO('mysql:host=localhost;dbname=CoursPHP','root',
'xxxx');
```

L'objet `$bdd` est une instance de la classe PDO. Il établit une connexion avec la base de données « CoursPHP » gérée par MySQL, située sur le serveur « localhost ». Le login et le mot de passe sont « root » et « xxxx » (xxxx remplace le vrai mot de passe).

- *Gestion des erreurs de connexion*

Si une erreur se produit lors de la connexion à la base de données, ou d'une requête SQL, un objet de la classe PDOException peut être capturé.

Erreurs non contrôlées

Si l'erreur n'est pas capturée, un message d'erreur apparaît à l'écran, sans aucun contrôle. Le programme `MySQL_PDO_connexion_exception1_shell.php` tente de connecter la base dont le nom est saisi par l'utilisateur, sans aucun contrôle.

Listing 10.1 – Programme MySQL_PDO_connexion_exception1_shell.php

```
<?php
 echo "Nom de la base de données : ";
 $Nomdbb=fgets(STDIN); // Saisie du nom de la base de données
 $Nomdbb=trim($Nomdbb); //Suppression espaces (début,fin)
 // -- Connexion à la base de données --
 $bdd = new PDO('mysql:host=localhost;dbname='.$Nomdbb, 'root', 'xxxx');
 var_dump($bdd);
?>
```

La saisie d'une base inexistante affiche le message d'erreur :

Listing 10.2 – Exécution de MySQL_PDO_connexion_exception1_shell.php

```
$ php MySQL_PDO_connexion_exception1_shell.php
Nom de la base de données : Cours
Fatal error: Uncaught exception 'PDOException' with message
'SQLSTATE[HY000] [1049] Unknown database 'cours'' in /.../MySQL_PDO_
connexion_exception1.php:6 ...
```

Contrôle des erreurs

Le contrôle du message d'erreur utilise la syntaxe `try…catch()` de gestion des exceptions qui a été détaillée dans le chapitre 9. Le programme `MySQL_PDO_connexion_exception2_shell.php` capture l'exception de la classe PDOException levée lors d'une erreur de connexion à la base de données.

Listing 10.3 – Programme MySQL_PDO_connexion_exception2_shell.php

```
<?php
 echo "Nom de la base de données : ";
 $Nomdbb=fgets(STDIN); // Saisie du nom de la base de données
 $Nomdbb=trim($Nomdbb); // Suppression espaces (début,fin)
 // Connexion à la base de données
 try {
  $bdd = new PDO('mysql:host=localhost;dbname='.$Nomdbb, 'root', 'xxxx');
  var_dump($bdd);
 }
 catch(PDOException $e) {
  echo 'Erreur de connexion avec la base : '.$Nomdbb.PHP_EOL;
  echo 'Message : '.$e->getMessage().PHP_EOL;
 }
?>
```

La saisie d'un nom erroné affiche le message d'erreur.

Listing 10.4 – Exécution de MySQL_PDO_connexion_exception2_shell.php

```
$ php MySQL_PDO_connexion_exception2_shell.php
Nom de la base de données : Cours
Erreur de connexion avec la base : Cours
Message : SQLSTATE[HY000] [1049] Unknown database 'cours'
```

Remarques

La classe « Exception » plus générique peut remplacer la classe « PDOException ».

- *Connexion persistante*

Une connexion persistante maintient la connexion, même après la fin du programme PHP. Cela est utile avec les d'applications web ou chaque page lance un programme PHP différent. La connexion ouverte est réutilisable par un autre programme, dès lors qu'il tente d'établir une connexion avec les mêmes paramètres. La

connexion mise en cache est utilisée, l'accès est plus rapide. La syntaxe d'ouverture d'une connexion persistante est :

```
$bdd = new PDO('mysql:host=localhost;dbname=CoursPHP',
          'root', 'xxxx', array(PDO::ATTR_PERSISTENT => true));
```

Le langage PHP prévoit une autre syntaxe *via* la méthode `setAttribute()` :

```
$bdd = new PDO('mysql:host=localhost;dbname=CoursPHP', 'root', 'xxxx');
$bdd->setAttribute(PDO::ATTR_PERSISTENT, true);
```

Remarque

Selon le pilote de base de données, cette dernière syntaxe peut ne pas fonctionner. Il est préférable d'utiliser ATTR_PERSISTENT à l'ouverture de la connexion.

- *Codage des caractères en UTF-8*

Avec les caractères accentués, l'affichage peut être correct sur un poste de travail et erroné sur un autre. Plusieurs normes ont tenté de mettre un peu d'ordre dans cet imbroglio de tables de codages (*cf.* chapitre 5 sur le codage des caractères). Le codage UTF-8 est en passe d'apporter une solution pérenne car il est adopté par tous les environnements récents. Encore faut-il qu'il soit utilisé par toutes les briques logiciels d'un site web. Afin de garantir la gestion des caractères en UTF-8, il faut que ce codage soit utilisé par :

- la base de données, la table ou le champ, c'est le paramètre « interclassement » ;
- le mode d'accès à la base de données par le programme PHP et l'objet PDO ;
- la définition du « charset » pour la page HTML ou le programme PHP.

Interclassement UTF-8 pour la base de données ou la table

La base ou la table doivent utiliser un interclassement comme « utf8_general_ci » tel que cela est présenté dans les sections sur la création d'une base ou d'une table.

Encodage de la connexion à la base de données

Il faut indiquer explicitement que la table est codée en UTF-8, après ou au moment de son ouverture. Pour cela on exécute une requête SQL après l'ouverture de la connexion à la base *via* `exec()` comme cela est présenté dans les syntaxes suivantes. Cette méthode est valide pour MySQL et POSTGRESQL.

```
// -- Connexion de la base de données --
 $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP',
        'root', 'xxxx');
 // -- Définition du codage en UTF8 --
 $bdd->exec("SET CHARACTER SET utf8");
```

La seconde méthode, spécifique à MySQL, s'appuie sur un attribut particulier de l'ouverture de la connexion :

```
// Connexion de la base et définition du codage en UTF8
 $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP','root',
'xxxx',array(PDO::MYSQL_ATTR_INIT_COMMAND=>"SET CHARACTER SET utf8"));
```

Encodage de la page HTML ou PHP

Enfin, le `charset utf-8` doit être précisé dans les pages HTML ou les programmes PHP *via* la balise `meta` dont voici la syntaxe :

```
<html>
 <head> <!-- Entête HTML -->
  <meta charset="utf-8" />
  ...
 </head>
 ...
```

Fermer la connexion

La connexion à la base de données est active tant que l'objet de la classe PDO créé par `new` existe. La suppression de l'objet par `unset()` pour toutes ses références ou l'affectation de la valeur `null` clôt la connexion. Sans clôture explicite, PHP ferme la connexion à la fin du programme PHP.

```
$bdd = null;
```

Les requêtes sur une base de données

Principe

Cette section présente **l'usage de requêtes** dans un programme PHP.

L'accès aux données

La méthode `query()` appliquée à l'objet `$bdd` précédent de la classe PDO effectue une requête SQL. Il faut ensuite interpréter le résultat retourné.

- *query()*

Cette méthode exécute une requête SQL `SELECT` sur la base dont la connexion a été établie et qui est représentée par l'objet `$bdd`. Sa syntaxe est de la forme :

```
$reponse = $bdd->query('SELECT * FROM personnes');
```

Elle retourne un objet de la classe PDOStatement, `$reponse`.

- *fetch()*

La méthode `fetch()` de la classe PDOStatement retourne *une ligne* du jeu de résultat, soit les informations *d'une seule personne* dans notre exemple.

```
$une_personne = $reponse->fetch() ;
```

Par défaut, l'information retournée est un tableau contenant, pour chaque entrée de la table, sa valeur dans une case avec indice numérique et dans une case avec une étiquette au nom du champ. Le tableau retourné contient par exemple, pour DUPONT, JEAN âgé de 28 ans dont l'identifiant est le n° 1 :

- l'identifiant « 1 » dans la case « ID » et dans la case « 0 » ;
- le nom « DUPONT » dans la case « Nom » et dans la case « 1 » ;
- le prénom « JEAN » dans la case « Prenom » et dans la case « 2 » ;
- l'âge « 28 » dans la case « Age » et dans la case « 3 » ;

Voici le contenu de ce tableau affiché par `var_dump($une_personne)` :

```
array(8) {
  ["ID"]=>
  string(1) "1"
  [0]=>
  string(1) "1"
  ["Nom"]=>
  string(6) "DUPONT"
  [1]=>
  string(6) "DUPONT"
  ["Prenom"]=>
  string(4) "JEAN"
  [2]=>
  string(4) "JEAN"
  ["Age"]=>
  string(2) "28"
  [3]=>
  string(2) "28"
}
```

Pour spécifier le retour d'un tableau associatif ayant uniquement des cases étiquetées aux noms des champs, il suffit d'utiliser la syntaxe suivante :

```
$une_personne = $reponse->fetch(PDO::FETCH_ASSOC) ;
```

L'instruction `var_dump($une_personne)` affiche le tableau retourné :

```
array(4) {
  ["ID"]=>
  string(1) "1"
  ["Nom"]=>
  string(6) "DUPONT"
  ["Prenom"]=>
  string(4) "JEAN"
```

```
  ["Age"]=>
  string(2) "28"
}
```

Pour activer ce mode par défaut, il faut utiliser la méthode `setFetchMode()`.

```
$une_personne = $reponse->fetch(PDO::FETCH_ASSOC) ;
```

s'écrit également :

```
$reponse->setFetchMode(PDO::FETCH_ASSOC);
$une_personne = $reponse->fetch()) ;
```

- *fetchAll()*

La méthode `fetchAll()`de la classe PDOStatement retourne *toutes les lignes* du jeu de résultats, soit *toutes les personnes*. Voici son utilisation :

```
$toutes_les_personnes = $reponse->fetchAll() ;
```

La méthode `fetchAll()` retourne un tableau avec deux entrées pour chaque valeur : une entrée numérotée et une entrée avec comme étiquette le nom du champ. Le fonctionnement par défaut de `fetchAll()` peut être modifié afin que seul un tableau associatif soit retourné. La syntaxe devient :

```
$toutes_les_personnes = $reponse->fetchAll(PDO::FETCH_ASSOC) ;
```

ou bien :

```
$reponse->setFetchMode(PDO::FETCH_ASSOC);
$toutes_les_personnes = $reponse->fetchAll() ;
```

- *closeCursor()*

La méthode `closeCursor()`de la classe PDOStatement ferme le curseur de la requête, permettant ainsi à la requête d'être à nouveau exécutée. Voici sa syntaxe :

```
$reponse->closeCursor();
```

- *Exemples*

Les exemples suivants présentent `query()`, `fetch()` et `fetchAll()`. La table « personnes » existe dans la base de données « CoursPHP ».

query()* et *fetch()

Dans le programme `MySQL_PDO_query_fetch_personnes1_shell.php`, l'objet `$bdd` contient la connexion vers la base « CoursPHP ». L'objet `$reponse` contient le résultat de la requête SQL « `SELECT * FROM personnes` ». Une boucle `while` récupère dans le tableau `$une_personne`, une ligne du jeu de résultat de la requête SQL, soit une seule personne. La méthode `closeCursor()` termine la requête et autorise une nouvelle requête avec `$reponse`. L'instruction `$bdd=NULL` déconnecte la base de données.

Listing 10.5 – Programme MySQL_PDO_query_fetch_personnes1_shell.php

```
<?php
try { // -- Connexion de la base de données --
 $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP', 'root', 'xxxx');
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Exécution de la requête --
 $reponse = $bdd->query('SELECT * FROM personnes');
 if (!$reponse){ // Traitement des erreurs de retour
   throw new Exception('Problème de requête sur la table.'.PHP_EOL);
 }
 // -- Affichage des données retournées --
 echo "-- Contenu de la table personnes --".PHP_EOL;
 // -- Boucle de traitement de chaque personne --
 while ($une_personne = $reponse->fetch()) {
   echo $une_personne['ID']."\t";
   echo $une_personne['Nom']."\t";
   echo $une_personne['Prenom']."\t";
   echo $une_personne['Age']."\t";
   echo PHP_EOL;
 }
 $reponse->closeCursor();// Fermeture de la requête
 $bdd = NULL; // Fermeture de la connexion
}
catch(Exception $e){echo 'Erreur : '.$e->getMessage(); }
?>
```

Voici son exécution :

Listing 10.6 – Exécution de MySQL_PDO_query_fetch_personnes1_shell.php

```
$ php MySQL_PDO_query_fetch_personnes1_shell.php
-- Contenu de la table personnes --
1   DUPONT          JEAN            28
...
16  MARTIN          ALBERT          25
```

Le programme `MySQL_PDO_query_fetch_personnes1_web.php` est la version web du programme précédent. L'en-tête HTML est ajouté. Le résultat est présenté sous la forme d'un tableau HTML. Les éventuels messages d'erreur sont entourés d'un « fieldset ». La mise en forme pour le web est présentée en gras. La figure 10.18 présente le résultat de son exécution.

Tableau des personnes

ID	Nom	Prénom	Age
1	DUPONT	JEAN	28
2	JACQUENOD	JEAN-CHRISTOPHE	54
3	MURCIAN	CAROLE	44
4	LERY	JEAN-MICHEL	25
5	DE-LA-RUE	JEAN-CHRISTOPHE	27
6	MARTIN	PIERRE-DAVID	27
7	MARTIN	PIERRE	56
8	JACQUENOD	FREDERIC	25
9	JACQUENOD	LAURENCE	24
10	DUMOULIN	JEAN-CHRISTOPHE	54
11	LABONNE-JAYAT	OLIVIER	54
12	DE-LA-FONTAINE	JEAN	110
13	LEVY	SAMUEL	56
14	DE-LA-RUE	LAURENCE	25
15	DUPONT	JEAN	54
16	MARTIN	ALBERT	25

Figure 10.18 – **Affichage web query-fetch.**

Listing 10.7 – Programme MySQL_PDO_query_fetch_personnes1_web.php

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Affichage de la table personnes</title>
     <link href="../../CSS/MySQL.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
   <?php
   define("WEB_EOL","<br/>");
   try { // -- Connexion de la base de données --
    $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP', 'root', 'xxxx');
    $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
    // -- Exécution de la requête --
    $reponse = $bdd->query('SELECT * FROM personnes');
    if (!$reponse){ // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
    }
    // -- Affichage des données sous la forme d'un tableau --
    ?>
    <table summary="Tableau des personnes">
    <caption>Tableau des personnes</caption>
    <thead>
    <tr>
    <!-- entête du tableau -->
        <th>ID</th>
        <th>Nom</th>
        <th>Pr&eacute;nom</th>
        <th>Age</th>
        </tr>
```

```
    </thead>
    <?php
    // -- Boucle de traitement de chaque personne --
    while ($une_personne = $reponse->fetch()){
     echo "<tr>";
     echo "<td>".$une_personne['ID']."</td>";
     echo "<td>".$une_personne['Nom']."</td>";
     echo "<td>".$une_personne['Prenom']."</td>";
     echo "<td>".$une_personne['Age']."</td>";
     echo "</tr>";
    }
    ?>
    </table>
    <?php
    $reponse->closeCursor();// Fermeture de la requête
    $bdd = NULL; // Fermeture de la connexion
   }
   catch(Exception $e) {
     echo "<fieldset>";
     echo "<legend>Erreur d'accès à la base de données :</legend>".WEB_EOL;
     echo 'Erreur : '.$e->getMessage().WEB_EOL;
     echo "</fieldset>";
   }
   ?>
 </body>
</html>
```

Le programme `MySQL_PDO_query_fetch_personnes2_shell.php` est une adaptation du programme précédent. La méthode `fetch()`utilise la constante `PDO::FETCH_ASSOC`. À l'intérieur de la boucle `while`, une boucle `foreach` parcourt chaque champ de la personne. Seules les parties modifiées sont reprises :

Listing 10.8 – Programme MySQL_PDO_query_fetch_personnes2_shell.php

```
<?php
try { // -- Connexion de la base de données --
 ...
 while ($une_personne = $reponse->fetch(PDO::FETCH_ASSOC)) {
   foreach($une_personne as $NomChamp => $ContenuChamp) {
    echo "$une_personne[$NomChamp]"."\t";
   }
   echo PHP_EOL;
 }
 ...
}
catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
?>
```

Le troisième exemple `MySQL_PDO_query_fetch_personnes3_shell.php` utilise la méthode `setFetchMode()`pour modifier le comportement de `fetch()`afin

d'obtenir un tableau associatif. Il récupère le nom des champs de la table, à partir de sa structure (`DESCRIBE`), pour les afficher en première ligne de résultat.

Listing 10.9 – Programme MySQL_PDO_query_fetch_personnes3_shell.php

```
<?php
try { // -- Connexion de la base de données --
 $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP', 'root', 'xxxx');
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8 --
 // = On récupère la structure de la table (nom des champs) =
 $struture_table = $bdd->query('DESCRIBE personnes');
 if (!$struture_table)
     throw new Exception('Problème de requête sur la table.');
 // -- Par défaut, fetch retourne un tableau associatif --
 $struture_table->setFetchMode(PDO::FETCH_ASSOC);
 echo "------------------------------".PHP_EOL;
 // -- Boucle d'affichage du nom des champs --
 while ($Liste_Champs = $struture_table->fetch()) {
  $NomChamp=$Liste_Champs['Field'];
  echo "$NomChamp\t";
 }
 echo PHP_EOL;
 echo "------------------------------".PHP_EOL;
 // == On récupère les données de la table ==
 $reponse = $bdd->query('SELECT * FROM personnes');
 if (!$reponse) // Traitement des erreurs de retour
   throw new Exception('Problème de requête sur la table.');
 // -- Affichage des données retournées --
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 // -- Boucle de traitement de chaque personne --
 while ($une_personne = $reponse->fetch()) {
   foreach($une_personne as $NomChamp => $ContenuChamp) {
    echo "$une_personne[$NomChamp]"."\t";
   }
   echo PHP_EOL;
 }
 $reponse->closeCursor(); // Fermeture de la requête
 $bdd = NULL            ; // Fermeture de la connexion
}
catch(Exception $e) { echo 'Erreur : '.$e->getMessage();}
?>
```

query() et *fetchAll()*

Le programme `MySQL_PDO_query_fetchAll_personnes_shell.php` utilise `fetchAll()`. Toutes les personnes sont retournées dans le tableau associatif `$reponse`. Une boucle `foreach` récupère chaque personne.

Listing 10.10 – Programme MySQL_PDO_query_fetchAll_personnes_shell.php

```
<?php
try { // -- Connexion de la base de données --
```

```
$bdd = new PDO('mysql:host=localhost;dbname=CoursPHP', 'root', 'xxxx');
$bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
// -- Exécution de la requête --
$reponse = $bdd->query('SELECT * FROM personnes');
if (!$reponse){ // Traitement des erreurs de retour
  throw new Exception('Pb de requête sur la table.'.PHP_EOL);
}
// -- Affichage des données retournées --
echo "-- Contenu de la table personnes ---".PHP_EOL;
$reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
// -- Boucle de traitement de chaque personne --
foreach ($reponse->fetchAll() as $etiquette => $une_personne)
{ echo $une_personne['ID']."\t";
  echo $une_personne['Nom']."\t";
  echo $une_personne['Prenom']."\t";
  echo $une_personne['Age']."\t";
  echo PHP_EOL;
}
$reponse->closeCursor();// Fermeture de la requête
$bdd = NULL; // Fermeture de la connexion
}
catch(Exception $e) { echo 'Erreur : '.$e->getMessage();}
?>
```

Les critères de sélection

Cette section présente des exemples de critères de sélection. Le programme `MySQL_PDO_query_fetch_where_personnes_shell.php` utilise la clause `WHERE`. Il affiche les personnes de la table « personnes », demande l'âge minimal comme critère, puis affiche la liste des personnes sélectionnées.

La fonction `affichage_liste_personnes()`, contenue dans le fichier `MySQL_include_sprog_commun_shell.php`, s'adapte à la liste des champs retournée. Les paramètres de connexion sont dans le fichier `MySQL_include_param_dbb.php`.

Listing 10.11 – Programme MySQL_PDO_query_fetch_where_personnes_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try { // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
              $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Exécution de la requête --
 $reponse = $bdd->query('SELECT * FROM personnes');
 if (!$reponse) // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 $tab_personnes=$reponse->fetchAll();// Toutes les personnes
```

```
 // -- Affichage des données retournées --
 affichage_liste_personnes("Liste des personnes",$tab_personnes);
 $reponse->closeCursor();// Fermeture de la requête
 // == Saisie du filtre ==
 echo PHP_EOL;
 echo "Entrez la valeur minimale de l'âge à afficher : ";
 $saisie  = fgets(STDIN)   ;
 $Age_Min = intval($saisie);
 // == Affichage filtré de la table personnes ==
 // -- Exécution de la requête --
 $reponse = $bdd->query('SELECT * FROM personnes WHERE Age>='.$Age_Min);
 if (!$reponse) // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 $tab_personnes=$reponse->fetchAll();// Toutes les personnes
 // -- Affichage des données retournées --
 affichage_liste_personnes("Personnes ayant un Age >= $Age_Min",$tab_
personnes);
 $reponse->closeCursor();// Fermeture de la requête
 $bdd = NULL; // Fermeture de la connexion à la base
}
catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
?>
```

Voici le contenu du fichier `MySQL_include_sprog_commun_shell.php`.

Listing 10.12 – Fichier MySQL_include_sprog_commun_shell.php

```
<?php
// -- Fonction d'affichage du tableau --
function affichage_liste_personnes($texte,$tab_mixte)
{if (count($tab_mixte)==0)
        throw new Exception('Aucun élément à afficher.');
 // -- Affichage de l'en-tête du tableau --
 reset($tab_mixte);
 $une_personne=current($tab_mixte);
 $liste_champs=array_keys($une_personne);
 echo "------------------------------".PHP_EOL;
 echo " $texte".PHP_EOL;
 echo "------------------------------".PHP_EOL;
 foreach($liste_champs as $NomChamp) {
  echo "$NomChamp\t";
 }
 echo PHP_EOL;
 echo "------------------------------".PHP_EOL;
 // -- Boucle de traitement de chaque personne --
 foreach ($tab_mixte as $une_personne)
 { // -- On affiche le contenu des champs --
   foreach($liste_champs as $NomChamp) {
    echo $une_personne[$NomChamp]."\t";
   }
```

```
    echo PHP_EOL;
  }
}
?>
```

Voici le contenu du fichier `MySQL_include_param_dbb.php`.

Listing 10.13 – Fichier MySQL_include_param_dbb.php

```
<?php
// -- paramètres de connexion à la base de données --
$TYPE_DBB="mysql";
$SERVEUR="localhost";
$BASEDD="CoursPHP";
$LOGIN_ADM="root";
$MDP_ADM="xxxx";
?>
```

Voici un exemple d'exécution :

Listing 10.14 – Exécution de MySQL_PDO_query_fetch_where_personnes_shell.php

```
$ php MySQL_PDO_query_fetch_where_personnes_shell.php
------------------------------
ID  Nom  Prenom  Age
------------------------------
1   DUPONT         JEAN             28
2   JACQUENOD      JEAN-CHRISTOPHE  54
3   MURCIAN        CAROLE           44
4   LERY           JEAN-MICHEL      25
5   DE-LA-RUE      JEAN-CHRISTOPHE  27
6   MARTIN         PIERRE-DAVID     27
7   MARTIN         PIERRE           56
8   JACQUENOD      FREDERIC         25
9   JACQUENOD      LAURENCE         24
10  DUMOULIN       JEAN-CHRISTOPHE  54
11  LABONNE-JAYAT  OLIVIER          54
12  DE-LA-FONTAINE JEAN             110
13  LEVY           SAMUEL           56
14  DE-LA-RUE      LAURENCE         25
15  DUPONT         JEAN             54
16  MARTIN         ALBERT           25

Entrez la valeur minimale de l'âge à afficher : 33
------------------------------
ID  Nom  Prenom  Age
------------------------------
2   JACQUENOD      JEAN-CHRISTOPHE  54
3   MURCIAN        CAROLE           44
7   MARTIN         PIERRE           56
10  DUMOULIN       JEAN-CHRISTOPHE  54
```

```
11  LABONNE-JAYAT  OLIVIER          54
12  DE-LA-FONTAINE JEAN             110
13  LEVY           SAMUEL           56
15  DUPONT         JEAN             54
```

Le programme `MySQL_PDO_query_fetch_where_personnes_web.php` est la version web. Les parties spécifiques à la syntaxe web sont mises en gras. Le début du programme vérifie si l'exécution provient de la sélection du bouton validé :

```
    if (empty($_POST['valider']))
```

Si ce n'est pas le cas, on affiche la liste de toutes les personnes ainsi que le formulaire de saisie du critère de sélection. Sinon, on récupère le critère de sélection transmis *via* la méthode POST.

```
     $Age_Min = $_POST['Age_Min'];
     $Age_Min = intval($Age_Min);
```

Puis on affiche le résultat du filtrage selon l'âge minimal indiqué.

Listing 10.15 – Programme MySQL_PDO_query_fetch_where_personnes_web.php

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Affichage de la table personnes</title>
    <link href="../../CSS/MySQL.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
   <?php
   define("WEB_EOL","<br/>");
   include '../../INCLUDE/MySQL_include_param_dbb.php';
   include '../../INCLUDE/MySQL_include_sprog_commun_web.php';
   try {
    // -- Affichage de la liste complète des personnes --
    if (empty($_POST['valider']))
    { // -- Connexion de la base de données --
$bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
      $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
     $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
     // -- Exécution de la requête --
     $reponse = $bdd->query('SELECT * FROM personnes');
     // -- Traitement des erreurs de retour sur la requête --
     if (!$reponse)
        throw new Exception('Problème de requête sur la table.');
     // -- Retourne un tableau associatif --
     $reponse->setFetchMode(PDO::FETCH_ASSOC);
     // -- Retour de toutes les personnes --
     $tab_personnes=$reponse->fetchAll();
     // -- Affichage des données retournées --
```

```
 affichage_liste_personnes("Liste des personnes",$tab_personnes);
 $reponse->closeCursor();// Fermeture de la requête
 ?>
 <!--
 -- formulaire de saisie du critère de filtrage --
 -->
 <form action="MySQL_PDO_query_fetch_where_personnes_web.php"
 method="post">
  <fieldset>
  <legend>Saisissez les donn&eacute;es pour un filtrage :
  </legend><br/>
  Entrez l'&acirc;ge de s&eacute;lection (ex : 54) : <input
  type="text" name="Age_Min" size="3" maxlength="3" pattern="[1-9]
  [0-9]{1,2}" /><br/><br/>
  <input type="submit" name="valider" value="Valider le filtrage" />
  <input type="reset" value="Effacer le formulaire" />
  </fieldset>
 </form>
 <?php
}
else
{
 // -- Liste des personnes selon le critère de sélection
 // -- Récupération de la variable Age_Min --
 $Age_Min = $_POST['Age_Min'];
 $Age_Min = intval($Age_Min);
 // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_
 DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
  $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Exécution de la requête --
 $reponse = $bdd->query('SELECT * FROM personnes WHERE Age>='.$Age_
 Min);
 // -- Traitement des erreurs de retour sur la requête --
 if (!$reponse)
    throw new Exception('Problème de requête sur la table.');
 // -- Retourne un tableau associatif --
 $reponse->setFetchMode(PDO::FETCH_ASSOC);
 // -- Retour de toutes les personnes --
 $tab_personnes=$reponse->fetchAll();
 // -- Affichage des données retournées --
 affichage_liste_personnes("Personnes ayant un Age >= $Age_
 Min",$tab_personnes);
 $reponse->closeCursor();// Fermeture de la requête
 }
}
catch(Exception $e) {
  echo "<fieldset>";
  echo "<legend>Erreur d'accès à la base de données :</legend>".WEB_EOL;
```

```
      echo 'Erreur : '.$e->getMessage().WEB_EOL;
      echo "</fieldset>";
    }
    ?>
 </body>
</html>
```

Voici le contenu du fichier `MySQL_include_sprog_commun_web.php`.

Listing 10.16 – Fichier MySQL_include_sprog_commun_web.php

```
<?php
// -- Fonction d'affichage du tableau --
function affichage_liste_personnes($texte,$tab_mixte)
{   // -- Affichage entête du tableau --
    reset($tab_mixte);
    $une_personne=current($tab_mixte);
    $liste_champs=array_keys($une_personne);
    ?>
    <table summary="<?php echo $texte;?>">
    <caption><?php echo $texte;?></caption>
    <?php
    echo "<thead>";
    echo "<tr>";
    foreach($liste_champs as $NomChamp) {
      echo "<th>$NomChamp</th>";
    }
    echo "</tr>";
    echo "</thead>";
    // -- Boucle de traitement de chaque personne --
    foreach ($tab_mixte as $une_personne) {
     // -- On affiche le contenu des champs --
     echo "<tr>";
     foreach($liste_champs as $NomChamp) {
      echo "<td>".$une_personne[$NomChamp]."</td>";
     }
     echo "</tr>";
    }
    ?>
    </table>
    <?php
}
?>
```

Le fichier `MySQL_include_param_dbb.php` est identique au précédent. Voici les écrans de l'exécution du programme. Le premier écran (1) de la figure 10.19 affiche le contenu de la table « personnes », puis le formulaire de saisie du critère de sélection. Après validation, le résultat apparaît sur le deuxième écran (2).

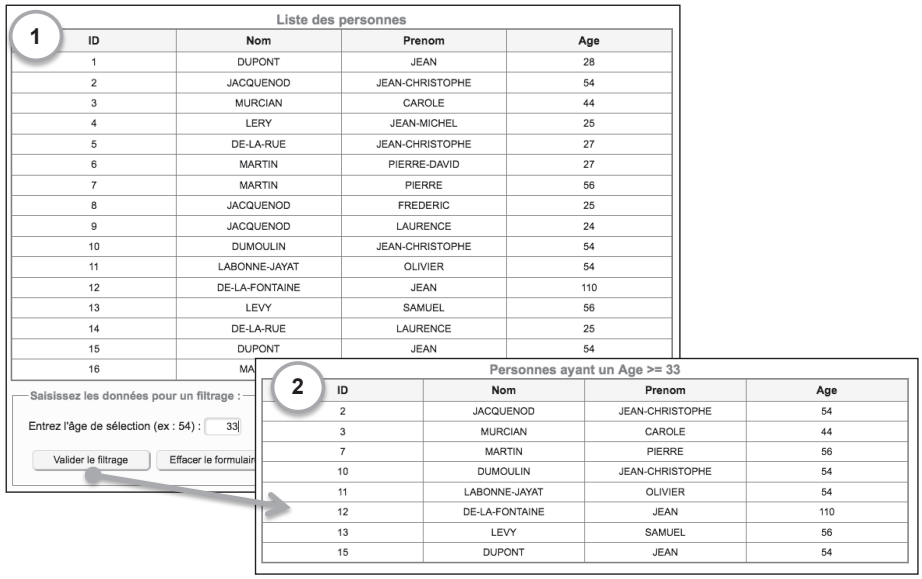

Figure 10.19 – Affichage web query-fetch-where.

D'autres programmes, téléchargeables sur le site de l'éditeur, présentent les versions shell et web pour un usage en SQL :

- de la clause `WHERE` avec un `ORDER BY` et `LIMIT`. sur l'âge ;
- des fonctions d'agrégat `AVG()`, `ROUND()`, `SUM()`avec `GROUP BY` ;
- des fonctions sur les chaînes de caractères `CONCAT()`, `LOWER()` ;
- de la fonction mathématique `TRUNCATE()` ;
- de la fonction de format de date `DATE_FORMAT()`.

Les jointures internes

Cette section présente la **jointure interne** avec la clausse `WHERE`. Les deux tables servant de support aux jointures sont :

- `clients_bancaires` : contient la liste des clients ;
- `comptes_bancaires` : contient la liste des comptes bancaires. Un des champs indique l'ID du propriétaire.

Le programme suivant est une jointure interne avec la clause `WHERE`.

Listing 10.17 – Programme MySQL_PDO_query_fetch_jointure_interne_where1_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try { // === connexion de la base de données ===
```

```
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_
ADM,
    $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // --- exécution de la requête ---
 $reponse = $bdd->query('SELECT clients_bancaires.Nom,clients_
bancaires.Prenom,comptes_bancaires.libelle,comptes_bancaires.Solde
FROM clients_bancaires, comptes_bancaires WHERE clients_bancaires.
ID_Clt=comptes_bancaires.ID_Clt');
 if (!$reponse) // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 // --- Boucle de traitement de chaque client ---
 $tab_clients=$reponse->fetchAll();
 // --- Affichage des données retournées ---
 affichage_liste_personnes("Solde par compte bancaire et
 propriétaire",$tab_clients);
 $reponse->closeCursor();// Fermeture de la requête
 $bdd = NULL; // Fermeture de la connexion à la base
}
catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
?>
```

Voici son exécution. Certaines lignes sont remplacées par des « … ».

Listing 10.18 – Exécution de MySQL_PDO_query_fetch_jointure_interne_where1_shell.php

```
$ php MySQL_PDO_query_fetch_jointure_interne_where1_shell.php
------------------------------------------------------------
 Solde par compte bancaire et propriétaire
------------------------------------------------------------
Nom             Prenom             libelle        Solde
------------------------------------------------------------
DUPONT          JEAN            Compte de dépô   750.98
DUPONT          JEAN            Livret A         765.32
...
MARTIN          ALBERT          Compte de dépô   363.49
MARTIN          ALBERT          Carte débit di   -150
```

D'autres programmes téléchargeables sur le site de l'éditeur montrent comment calculer le solde total par proriétaire de compte.

La jointure avec la clause `WHERE` est devenue obsolète, il faut maintenant utiliser la syntaxe `INNER JOIN`. La réécriture de la requête SQL :

```
SELECT cb.ID_Clt,cl.Nom,cl.Prenom,ROUND(SUM(cb.Solde),2) Solde_Total
FROM comptes_bancaires cb,clients_bancaires cl WHERE cb.ID_Clt=cl.
ID_Clt GROUP BY cb.ID_Clt;
```

se note :

```
SELECT cb.ID_Clt,cl.Nom,cl.Prenom,ROUND(SUM(cb.Solde),2) Solde_Total
FROM comptes_bancaires cb INNER JOIN clients_bancaires cl ON cb.ID_
Clt=cl.ID_Clt GROUP BY cb.ID_Clt;
```

Les jointures externes

Les jointures externes sélectionnent toutes les données, mêmes celles qui sont absentes dans l'autre table. Les deux syntaxes sont :

- `LEFT JOIN` : toutes les données de la table située à gauche de `JOIN` sont affichées, même celles n'ayant aucune correspondance dans la table de droite ;
- `RIGHT JOIN` : toutes les données de la table située à droite de `JOIN` sont affichées, même celles n'ayant aucune correspondance dans la table de gauche.

La syntaxe suivante basée sur `LEFT JOIN` affiche le solde total de chaque compte (table de gauche `comptes_bancaires`), y compris ceux n'ayant aucun propriétaire connu dans la table des clients.

```
$reponse = $bdd->query('SELECT cb.ID_Clt,cl.Nom,cl.Prenom,ROUND(SUM(cb.
Solde),2) Solde_Total FROM comptes_bancaires cb LEFT JOIN clients_
bancaires cl ON cb.ID_Clt=cl.ID_Clt GROUP BY cb.ID_Clt');
```

La syntaxe suivante basée sur la clause `RIGHT JOIN` affiche le solde total de chaque client (table de droite `clients_bancaires`), y compris ceux qui n'ont pas de compte dans la table des comptes.

```
$reponse = $bdd->query('SELECT cl.ID_Clt,cl.Nom,cl.Prenom,ROUND(SUM(cb.
Solde),2) Solde_Total FROM comptes_bancaires cb RIGHT JOIN clients_
bancaires cl ON cb.ID_Clt=cl.ID_Clt GROUP BY cl.ID_Clt');
```

La modification des données

- *exec()*

La méthode `exec()` exécute une requête SQL telle que `INSERT`, `UPDATE` ou `DELETE`. Elle retourne le nombre de lignes affectées par la requête. Cette méthode ne s'applique pas à l'instruction SQL `SELECT`, pour laquelle il faut privilégier les méthodes `query()`ou `prepare()`. La syntaxe de `exec()` est de la forme :

```
$requete="INSERT INTO personnes (Nom, Prenom, Age) VALUES
('$Nom','$Prenom',$Age)";
$reponse = $bdd->exec($requete);
```

ou

```
$bdd->exec("SET CHARACTER SET utf8");
```

- *L'insertion*

Cette section présente l'insertion de données *via* la requête SQL `INSERT INTO`. La table utilisée comme support est celle des « personnes » ayant comme champs :

- ID : l'identifiant de la personne ;
- Nom : le nom de la personne ;
- Prenom : le prénom de la personne ;
- Age: l'âge de la personne.

Le programme `MySQL_PDO_insertion_personnes_shell.php` reprend l'exercice n° 1 du chapitre 8 pour l'adapter à l'insertion dans la table « personnes ».

Listing 10.19 – Programme MySQL_PDO_insertion_personnes_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try { // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
        $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Affichage de toute la table des personnes --
 affichage_table($bdd,"Liste des personnes","personnes");
 // -- Saisie d'une liste de personnes --
 $saisie="saisie non vide";
 echo PHP_EOL;
 echo "-----------------------------------".PHP_EOL;
 echo "-- Saisie de nouvelles personnes --".PHP_EOL;
 echo "-----------------------------------".PHP_EOL;
 while (!empty($saisie))
 {echo "Entrez un nom,un prenom,un âge (ex:Dupont;Jean;28): ";
  $saisie=fgets(STDIN);
  // -- Traitement de la chaîne lue --
  $saisie=trim($saisie); //Suppression espaces (début,fin)
  // Remplace les espaces multiples par un seul espace
  $saisie= preg_replace('/\s{2,}/',' ',$saisie);
  // Remplace ; suivi d'un espace par ;
  $saisie= preg_replace('/;\s/',';',$saisie);
  // Remplace un espace suivi d'un ; par ;
  $saisie= preg_replace('/\s;/',';',$saisie);
  // On vérifie que la saisie n'est pas vide
  if (!empty($saisie))
  {// Rangement dans les variables
   list($Nom,$Prenom,$Age)=explode(';',$saisie);
   $Nom    = normalisation_nom($Nom)      ;
   $Prenom = normalisation_nom($Prenom)   ;
   $Age    = normalisation_numerique($Age);
   $Age    = intval($Age)                 ;
   // -- Insertion dans la base de données --
   if (!empty($Nom) && !empty($Prenom) && !empty($Age) )
   {echo "Insertion de : $Nom,$Prenom,$Age".PHP_EOL;
    // -- Préparation de la requête --
    $requete="INSERT INTO personnes (Nom, Prenom, Age) VALUES
    ('$Nom','$Prenom',$Age)";
```

```
     echo "requete = $requete".PHP_EOL;
     $reponse = $bdd->exec($requete); // Exécution
     if (!$reponse) {
      throw new Exception('Problème de requête sur la table.'.PHP_EOL);
     }
    }
    else {
      throw new Exception('Un des champs est vide.'.PHP_EOL);
    }
   }
  }
  echo PHP_EOL;
  // -- Affichage de toute la table des personnes --
  affichage_table($bdd,"Liste des personnes","personnes");
 }
 catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
 ?>
```

La requête `INSERT INTO` est exécutée avec la méthode `exec`.

Ce programme utilise les fonctions suivantes qui sont dans le fichier `MySQL_include_sprog_commun_shell.php` :

- `affichage_table($bdd,"Liste des personnes","personnes")` : affiche la table dont le nom est indiqué en 3e argument, avec le texte précisé en 2e argument. L'objet pointant sur la base contenant la table est indiqué en 1er ;
- `normalisation_nom($Nom)` : normalise le nom ou le prénom, sans accent, toutes les lettres sont en majuscules, et les espaces sont remplacés par des « - ». Les accents sont remplacés par le caractère majuscule équivalent grâce à la fonction `supprime_accent($chaine)` ;
- `normalisation_numerique ($Age)` : normalise le numérique entier ou réel avec le point ou la virgule décimale.

Voici le code source de ces fonctions. Dans la fonction `supprime_accents()`, certains accents sont remplacés par des « … ».

Listing 10.20 – Fichier MySQL_include_sprog_commun_shell.php

```
// -- Fonction d'affichage du contenu de la table --
function affichage_table($bdd,$texte,$NomTable)
{// -- Exécution de la requête --
 $reponse = $bdd->query('SELECT * FROM '.$NomTable);
 if (!$reponse) // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 $tab_clients=$reponse->fetchAll();// Toutes les personnes
 // -- Affichage des données retournées --
 affichage_liste_personnes("$texte",$tab_clients);
 $reponse->closeCursor();// Fermeture de la requête
}
```

```
// -- Fonction outil de suppression des accents --
function supprime_accent($chaine)
{// Tableau des caractères accentués à remplacer
  $caracteres_a_remplacer = array('À','Á','Â','Ã','Ä','Å',
  'Æ','Ç','È','É','Ê','Ë', ... ,'Ǽ','ǽ','Ǿ','ǿ');
  // Tableau des caractères sans accent de remplacement
  $caracteres_de_remplacement=array('A','A','A','A','A','A',
  'AE','C','E','E','E','E', ... ,'AE','ae','O','o');
  return str_replace($caracteres_a_remplacer, $caracteres_de_
  remplacement, $chaine);
}
// -- Fonction outil de normalisation des noms  --
function normalisation_nom($chaine)
{// Tableau des motifs de recherche
 $TabMotif=array('/[^a-zA-Z -]/', '/[ -]+/', '/^-|-$/');
 // Tableau des caractères de remplacement
 $TabRemplace=array('', '-', '')     ;
 // Chaine sur laquelle s'effectue le remplacement
 $chaine_contexte=supprime_accent($chaine);
 // Retour de la fonction
 return strtoupper(preg_replace($TabMotif,$TabRemplace, $chaine_
 contexte));
}
// -- Fonction outil de normalisation des numériques  --
function normalisation_numerique($numero)
{// Tableau des motifs de recherche
 $TabMotif=array('/[^0-9,.]/');
 // Tableau des caractères de remplacement
 $TabRemplace=array('')     ;
 // Retour de la fonction
 $NumR=intval(preg_replace($TabMotif,$TabRemplace,$numero));
 return $NumR;
}
```

La requête d'insertion est conservée dans la variable `$requete` qui est affichée à l'écran. Certaines lignes sont remplacées par des « … ».

Listing 10.21 – Exécution de MySQL_PDO_insertion_personnes_shell.php

```
$ php MySQL_PDO_insertion_personnes_shell.php
---------------------------------------
 Liste des personnes
---------------------------------------
ID  Nom             Prenom           Age
---------------------------------------
1   DUPONT          JEAN              28
...
16  MARTIN          ALBERT            25

-----------------------------------
-- Saisie de nouvelles personnes --
```

```
-----------------------------------
Entrez un nom, un prenom, un âge(ex:Dupont;Jean;28) : lemy;kévin;25
Insertion de : LEMY,KEVIN,25
requete = INSERT INTO personnes (Nom, Prenom, Age) VALUES
('LEMY','KEVIN',25)
Entrez un nom,un prenom,un âge (ex:Dupont;Jean;28): kaczma ; sylvie
samantha ; 52
Insertion de : KACZMA,SYLVIE-SAMANTHA,52
requete = INSERT INTO personnes (Nom, Prenom, Age) VALUES
('KACZMA','SYLVIE-SAMANTHA',52)
Entrez un nom,un prenom,un âge (ex:Dupont;Jean;28):

---------------------------------------
 Liste des personnes
---------------------------------------
ID  Nom             Prenom           Age
---------------------------------------
1   DUPONT          JEAN              28
...
16  MARTIN          ALBERT            25
17  LEMY            KEVIN             25
18  KACZMA          SYLVIE-SAMANTHA   52
```

- *La mise à jour*

Cette section présente la mise à jour des données *via* la requête SQL `UPDATE`. Le programme `MySQL_PDO_modification_personnes_shell.php` reprend l'exercice n° 1 du chapitre 8 et l'adapte à la modification de la table « personnes ».

Listing 10.22 – Programme MySQL_PDO_modification_personnes_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try { // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
   $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8");// Codage en UTF8
 // -- Affichage de toute la table des personnes --
 affichage_table($bdd,"Liste des personnes","personnes");
 // -- Saisie d'une liste de personnes --
 $saisie="saisie non vide";
 echo PHP_EOL;
 echo "-----------------------------------".PHP_EOL;
 echo "-- Modification d'une personne --".PHP_EOL;
 echo "-----------------------------------".PHP_EOL;
 $tab_critere=choix_critere();
 if (count($tab_critere) != 0)
 {  $critere = $tab_critere[0];
    $ValRech = $tab_critere[1];
```

```
// -- Recherche sur critère --
$tab_personnes_trouvees=Recherche_sur_
critere($bdd,'personnes',$critere,$ValRech);
// -- Traitement du tableau retourné --
if (count($tab_personnes_trouvees) == 0) {
  echo "Aucune personne trouvée avec ce critère".PHP_EOL;
}
else {
  affichage_liste_personnes("Liste des personnes trouvées",$tab_
  personnes_trouvees);
  echo PHP_EOL;
  unset($numero);
  echo "Sélectionnez le numéro (ID) de la personne à modifier : ";
  fscanf(STDIN,"%d", $numero);
  if ( (empty($numero))||(!array_key_exists($numero,$tab_personnes_
  trouvees)) )
    echo "Aucune sélection".PHP_EOL;
  else {
    unset($tab_une_personne);
    $une_personne=$tab_personnes_trouvees[$numero];
    $tab_une_personne[]=$une_personne;
    affichage_liste_personnes("Personne à modifier",$tab_une_
    personne);
    echo "Confirmez la modification (o/n) : ";
    fscanf(STDIN,"%s",$reponse);
    if ($reponse == "o") {
      // Importation des variables via le nom des champs
      extract($une_personne,EXTR_OVERWRITE);
      // Initialisation des éléments de la requete UPDATE
      $MAJ_Nom    ='';
      $MAJ_Prenom ='';
      $MAJ_Age    ='';
      echo "Saisissez les nouvelles informations (vide pour
      inchangé) !".PHP_EOL;
      // -- Saisie du champ Nom --
      fprintf(STDOUT, "Nom    actuel : %-20s Nouveau Nom    : ",$Nom);
      $entree=fgets(STDIN) ;
      // Normalisation du nom
      $entree=normalisation_nom($entree);
      // On vérifie que la saisie n'est pas vide
      if (!empty($entree)) {
        $Nom=$entree ;
        $MAJ_Nom =',Nom=\''.$Nom.'\'';
      }
      else echo "Pas de modification du Nom".PHP_EOL;
      // -- Saisie du champ prénom --
      fprintf(STDOUT, "Prénom actuel : %-20s Nouveau Prénom :
      ",$Prenom);
      $entree=fgets(STDIN) ;
      // Normalisation du prénom
      $entree=normalisation_nom($entree);
```

```
            // On vérifie que la saisie n'est pas vide
            if (!empty($entree)) {
              $Prenom=$entree ;
              $MAJ_Prenom =',Prenom=\''.$Prenom.'\'';
            }
            else echo "Pas de modification du prénom".PHP_EOL;
            // -- Saisie du champ âge --
            fprintf(STDOUT, "Age    actuel : %-20d Nouvel Age     : ",$Age);
            $entree=fgets(STDIN) ;
            // Suppression espaces (début,fin) et saut de ligne
            $entree=trim($entree);
            // On vérifie que la saisie n'est pas vide
            if (!empty($entree)) {
             $entree=intval($entree);
             if ($entree == 0) {
               echo "  Erreur : Valeur du champ âge doit être un entier !
               => Pas de modification ".PHP_EOL;
             }
             else { // On met à jour la donnée
               $Age=$entree ;
               $MAJ_Age =',Age='.$Age;
             }
            }
            else echo "Pas de modification de l'âge".PHP_EOL;
            // -- Mise à jour des données saisies --
            if (!empty($MAJ_Nom)||!empty($MAJ_Prenom)||!empty($MAJ_Age))
            {// -- Préparation de la requête --
             $requete='UPDATE personnes SET '.$MAJ_Nom.$MAJ_Prenom.$MAJ_
             Age.' WHERE ID='.$numero;
             // -- On supprime les espaces multiples --
             $requete= preg_replace('/\s{2,}/',' ',$requete);
             // -- Suppression de l'apostrophe juste apres SET
             $requete= str_replace('SET ,','SET ',$requete);
             echo $requete.PHP_EOL;
             $reponse = $bdd->exec($requete); // Exécution
             if (!$reponse) {
              throw new Exception('Problème de requête sur la
              table.'.PHP_EOL);
             }
            }
            else echo "Aucun champ n'a été modifié".PHP_EOL;
          }
          else echo "Aucune donnée modifiée".PHP_EOL;
        }
      }
  }
  // -- Affichage de toute la table des personnes --
  affichage_table($bdd,"Liste des personnes","personnes");
}
catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
?>
```

La requête `UPDATE` est exécutée avec la méthode `exec`.

Ce programme utilise de nouvelles fonctions présentes dans le fichier `MySQL_include_sprog_commun_shell.php` :

- `Recherche_sur_critere($bdd,'personnes',$critere,$ValRech)` : recherche la valeur du 4e argument, dans la table dont le nom est indiqué en 2e argument, selon le critère précisé en 3e argument.
  - Si l'argument est le nom et le prénom, la recherche se fait sur une partie (utilisation de la clause SQL `WHERE … LIKE`). Si l'argument est l'ID ou l'âge, la recherche se fait sur les valeurs exactes.
  - Le retour est un tableau contenant les personnes trouvées. L'objet PDO pointant sur la base de données contenant la table est indiqué en 1er argument.
- `choix_critere()` : affiche le menu du choix du critère de recherche et de la valeur à rechercher. Voici le code source de ces deux fonctions :

Listing 10.23 – Fonctions outils

```
// -- Outil de sélection du choix de recherche par critère --
function choix_critere()
{ $tab_critere_retour=array();
  echo "Critère de recherche : ".PHP_EOL;;
  echo "    -a- Identifiant (ID)".PHP_EOL;
  echo "    -b- Nom".PHP_EOL;
  echo "    -c- Prénom".PHP_EOL;
  echo "    -d- Age".PHP_EOL;
  echo "Entrez le critère (a,b,c ou d) : ";
  fscanf(STDIN,"%s",$critere);
  $critere=$critere[0];
  switch ($critere)
  { case 'a' : echo "Identifiant : "; break;
    case 'b' : echo "Nom         : "; break;
    case 'c' : echo "Prénom      : "; break;
    case 'd' : echo "Age         : "; break;
    default  : echo "Choix erroné !".PHP_EOL; break;
  }
  if (($critere >='a') && ($critere <='d'))
  {$ValRech=fgets(STDIN);
   trim($ValRech);
   $tab_critere_retour[0]=$critere;
   $tab_critere_retour[1]=$ValRech;
  }
 return $tab_critere_retour;
}
// -- Recherche sur critere --
function Recherche_sur_critere($bdd,$NomTable,$CritRech,$ValRech)
{
  switch($CritRech)
  { case 'a':$NomChamp='ID'    ;
```

```
            $ValRech=intval($ValRech);break;
    case 'b':$NomChamp='Nom'   ;
            $ValRech="'%".normalisation_nom($ValRech)."%'";
            break;
    case 'c':$NomChamp='Prenom';
            $ValRech="'%".normalisation_nom($ValRech)."%'";
            break;
    case 'd':$NomChamp='Age'   ;
            $ValRech=intval($ValRech);
            break;
  }
 // -- Préparation de la requête --
 $requete='SELECT * FROM '.$NomTable.' WHERE '.$NomChamp.' LIKE
 '.$ValRech;
 $reponse = $bdd->query($requete); // Exécution
 if (!$reponse) // Traitement des erreurs de retour
     throw new Exception('Problème de requête sur la table.');
 $reponse->setFetchMode(PDO::FETCH_ASSOC);//Tableau associatif
 // -- Boucle de traitement de chaque personne --
 $tab_personnes=$reponse->fetchAll();
 unset($tab_personnes_retourne);
 foreach($tab_personnes as $une_personne)
 {  $numero=$une_personne['ID'];
    $tab_personnes_retourne[$numero]=$une_personne;
 }
 return $tab_personnes_retourne;
}
```

La requête est conservée dans la variable `$requete` qui est affichée à l'écran.

Listing 10.24 – Exécution de MySQL_PDO_modification_personnes_shell.php

```
$ php MySQL_PDO_modification_personnes_shell.php
------------------------------
 Liste des personnes
-------------------------------------------
ID  Nom                 Prenom           Age
-------------------------------------------
1   DUPONT              JEAN              28
2   JACQUENOD           JEAN-CHRISTOPHE   54
3   MURCIAN             CAROLE            44
4   LERY                JEAN-MICHEL       25
5   DE-LA-RUE           JEAN-CHRISTOPHE   27
...
10  DUMOULIN            JEAN-CHRISTOPHE   54
...
---------------------------------
-- Modification d'une personne --
---------------------------------
Critère de recherche :
    -a- Identifiant (ID)
```

```
    -b- Nom
    -c- Prénom
    -d- Age
Entrez le critère (a,b,c ou d) : c
Prénom      : jean chris
------------------------------
 Liste des personnes trouvées
-----------------------------------------
ID  Nom                 Prenom           Age
-----------------------------------------
2   JACQUENOD           JEAN-CHRISTOPHE   54
5   DE-LA-RUE           JEAN-CHRISTOPHE   27
10  DUMOULIN            JEAN-CHRISTOPHE   54

Sélectionnez le numéro (ID) de la personne à modifier : 5
------------------------------
 Personne à modifier
-----------------------------------------
ID  Nom                 Prenom           Age
-----------------------------------------
5   DE-LA-RUE           JEAN-CHRISTOPHE   27
Confirmez la modification (o/n) : o
Saisissez les nouvelles informations (vide pour inchangé) !
Nom    actuel : DE-LA-RUE            Nouveau Nom    : du boulevard
Prénom actuel : JEAN-CHRISTOPHE      Nouveau Prénom : jean pascal
Age    actuel : 27                   Nouvel Age     : 57
UPDATE personnes SET Nom='DU-BOULEVARD',Prenom='JEAN-PASCAL',Age=57
WHERE ID=5
------------------------------
 Liste des personnes
-----------------------------------------
ID  Nom                 Prenom           Age
-----------------------------------------
...
5   DU-BOULEVARD        JEAN-PASCAL       57
...
```

- *La suppression*

Cette section présente la suppression de données *via* la requête SQL `DELETE`. Le programme `MySQL_PDO_suppression_personnes_shell.php` reprend l'exercice n° 1 du chapitre 8 pour supprimer une entrée dans la table « personnes ».

Listing 10.25 – Programme MySQL_PDO_suppression_personnes_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try {// -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
```

```
  $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
$bdd->exec("SET CHARACTER SET utf8");  // Codage en UTF8
// -- Affichage de toute la table des personnes --
affichage_table($bdd,"Liste des personnes","personnes");
// -- Saisie d'une liste de personnes --
$saisie="saisie non vide";
echo PHP_EOL;
echo "--------------------------------".PHP_EOL;
echo "-- Suppression d'une personne --".PHP_EOL;
echo "--------------------------------".PHP_EOL;
$tab_critere=choix_critere();
if (count($tab_critere) != 0)
{  $critere = $tab_critere[0];
   $ValRech = $tab_critere[1];
   // -- Recherche sur critère --
   $tab_personnes_trouvees=Recherche_sur_
   critere($bdd,'personnes',$critere,$ValRech);
   // -- Traitement du tableau retourné --
   if (count($tab_personnes_trouvees) == 0)
     echo "Aucune personne trouvée avec ce critère".PHP_EOL;
   else
   { affichage_liste_personnes("Liste des personnes trouvées",$tab_
   personnes_trouvees);
     echo PHP_EOL;
     unset($numero);
     echo "Sélectionnez le numéro (ID) de la personne à supprimer : ";
     fscanf(STDIN,"%d", $numero);
     if ( (empty($numero))||(!array_key_exists($numero,$tab_personnes_
     trouvees)) )
       echo "Aucune sélection".PHP_EOL;
     else
     { unset($tab_une_personne);
       $tab_une_personne[]=$tab_personnes_trouvees[$numero];
       affichage_liste_personnes("Personne à supprimer",$tab_une_personne);
       echo "Confirmez la suppression (o/n) : ";
       fscanf(STDIN,"%s",$reponse);
       if ($reponse == "o")
       {// -- Préparation de la requête --
        $requete='DELETE FROM personnes WHERE ID='.$numero;
        echo $requete.PHP_EOL;
        $reponse = $bdd->exec($requete); // Exécution
        if (!$reponse){
         throw new Exception('Problème de requête sur la table.'.PHP_
         EOL);
        }
       }
       else echo "Aucune donnée supprimée".PHP_EOL;
     }
   }
}
```

```
 // -- Affichage de toute la table des personnes --
 affichage_table($bdd,"Liste des personnes","personnes");
}
catch(Exception $e) {echo 'Erreur : '.$e->getMessage();}
?>
```

La requête DELETE est exécutée avec la méthode exec. La requête est conservée dans la variable $requete qui est affichée à l'écran.

Listing 10.26 – Exécution de MySQL_PDO_suppression_personnes_shell.php

```
$ php MySQL_PDO_suppression_personnes_shell.php
------------------------------
 Liste des personnes
--------------------------------------------
ID  Nom                 Prenom           Age
--------------------------------------------
1   DUPONT              JEAN              28
2   JACQUENOD           JEAN-CHRISTOPHE   54
...
8   JACQUENOD           FREDERIC          25
9   JACQUENOD           LAURENCE          24
...
--------------------------------
-- Suppression d'une personne --
--------------------------------
Critère de recherche :
    -a- Identifiant (ID)
    -b- Nom
    -c- Prénom
    -d- Age
Entrez le critère (a,b,c ou d) : b
Nom         : jacque
------------------------------
 Liste des personnes trouvées
--------------------------------------------
ID  Nom                 Prenom           Age
--------------------------------------------
2   JACQUENOD           JEAN-CHRISTOPHE   54
8   JACQUENOD           FREDERIC          25
9   JACQUENOD           LAURENCE          24

Sélectionnez le numéro (ID) de la personne à supprimer : 8
------------------------------
 Personne à supprimer
--------------------------------------------
ID  Nom                 Prenom           Age
--------------------------------------------
8   JACQUENOD           FREDERIC          25
Confirmez la suppression (o/n) : o
DELETE FROM personnes WHERE ID=8
```

```
 ----------------------------
 Liste des personnes
-------------------------------------------
ID  Nom                 Prenom          Age
-------------------------------------------
...
7   MARTIN              PIERRE           56
9   JACQUENOD           LAURENCE         24
...
```

Les requêtes préparées

- *Principe*

Il est courant d'effectuer plusieurs fois la même requête avec différentes valeurs. Le langage SQL permet de préparer à l'avance une requête (instruction SQL `PREPARE`) et de lui affecter un nom. Elle devient réutilisable et s'exécute (instruction SQL `EXECUTE`) sur les valeurs fournies au moment de son utilisation. Cette *requête préparée* n'existe que durant la session dans laquelle elle est créée. Elle utilise des variables de l'utilisateur créé *via* l'instruction SQL `SET`.

Le langage PHP propose deux méthodes pour gérer les *requêtes préparées* :

- `prepare()` : prépare la requête (classe PDO) ;
- `execute()` : exécute la requête (classe PDOStatement) ;

Ces méthodes sont équivalentes aux requêtes SQL `PREPARE` et `EXECUTE`.

Avec marqueur nommé

Avec la méthode `prepare()`, la syntaxe de la requête SQL doit être adaptée. Ainsi si la préparation de la requête dans le langage SQL s'écrit :

```
mysql> PREPARE selection_nom FROM 'SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=?';
```

la syntaxe PHP avec la méthode `prepare` se note :

```
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=:Nom');
```

L'information transmise à la requête préparée est un tableau associatif contenant dans l'entrée '**:Nom**' la valeur du paramètre **Nom** de la requête. Le marqueur dans la requête SQL se note « **:Nom** ». L'exécution de la requête dans le langage SQL :

```
mysql> SET @Nom='MARTIN';
mysql> EXECUTE selection_nom USING @Nom;
```

se note en PHP avec la méthode `execute` :

```
$Nom="MARTIN";
$RequetePreparee->execute(array(':Nom'=>$Nom));
```

Avec deux variables, la syntaxe SQL devient :

```
mysql> PREPARE selection_nom_like_prenom FROM 'SELECT ID,Nom,Prenom,Age
FROM personnes WHERE Nom=? AND Prenom LIKE ?';
mysql> SET @Nom='MARTIN';
mysql> SET @Prenom='%PIERRE%';
mysql> EXECUTE selection_nom_like_prenom USING @Nom, @Prenom;
```

Elle se note en PHP :

```
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=:Nom AND Prenom LIKE :Prenom');
$tab_param[':Nom']="MARTIN";
$tab_param[':Prenom']="%PIERRE%";
$RequetePreparee->execute($tab_param);
```

Remarque

La requête préparée doit aussi être fermée avec la méthode `closeCursor()`.

Avec marqueur anonyme

IL est possible de conserver les « ? » comme marqueurs dans la syntaxe SQL, même si cela est moins « parlant ». La requête dans le langage SQL suivant :

```
mysql> PREPARE selection_nom FROM 'SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=?';
```

se traduit avec la méthode `prepare` :

```
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=?');
```

Il faut alors indiquer dans le tableau des paramètres les différentes valeurs **par ordre d'utilisation**. Le tableau devient numérique et son premier indice est 0.

L'exécution de la requête dans le langage SQL :

```
mysql> SET @Nom='MARTIN';
mysql> EXECUTE selection_nom USING @Nom;
```

se note en PHP avec la méthode `execute()` :

```
$Nom="MARTIN";
$RequetePreparee->execute(array(0=>$Nom));
```

Avec deux variables, la syntaxe SQL devient :

```
mysql> PREPARE selection_nom_like_prenom FROM 'SELECT ID,Nom,Prenom,Age
FROM personnes WHERE Nom=? AND Prenom LIKE ?';
mysql> SET @Nom='MARTIN';
mysql> SET @Prenom='%PIERRE%';
mysql> EXECUTE selection_nom_like_prenom USING @Nom, @Prenom;
```

Elle se note en PHP :

```
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=? AND Prenom LIKE ?');
$tab_param[]="MARTIN";
$tab_param[]="%PIERRE%";
$RequetePreparee->execute($tab_param);
```

- *Exemple d'utilisation de prepare et execute*

Le programme `MySQL_PDO_prepare1_personnes_shell.php` présente ces méthodes avec une seule variable et affiche le résultat :

Listing 10.27 – Programme MySQL_PDO_prepare1_personnes_shell.php

```
<?php
include '../../INCLUDE/MySQL_include_param_dbb.php';
setlocale (LC_ALL, 'fr_FR.UTF-8');
try { // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
   $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Préparation de la requête --
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=:Nom');
 // -- Préparation de la variable --
 $tab_param[':Nom']="MARTIN";
 // -- Exécution de la requête préparée --
 $RequetePreparee->execute($tab_param);
 // -- Affichage du résultat de la requête --
 while ($donnees = $RequetePreparee->fetch())
 {  echo $donnees['ID']."\t"    ;
    echo $donnees['Nom']."\t"   ;
    echo $donnees['Prenom']."\t";
    echo $donnees['Age'].PHP_EOL;
 }
 // -- Fermeture de la requête préparée --
 $RequetePreparee->closeCursor();
}
catch(Exception $e) { echo 'Erreur : '.$e->getMessage();}
?>
```

Voici le résultat de son exécution :

Listing 10.28 – Exécution de MySQL_PDO_prepare1_personnes_shell.php

```
$ php MySQL_PDO_prepare1_personnes_shell.php
6   MARTIN   PIERRE-DAVID   27
7   MARTIN   PIERRE         56
16  MARTIN   ALBERT         25
```

Le programme `MySQL_PDO_prepare2_personnes_shell.php` présente ces deux méthodes avec deux variables transmises.

Listing 10.29 – Programme MySQL_PDO_prepare2_personnes_shell.php

```
<?php
 ...
 // -- Préparation de la requête --
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=:Nom AND Prenom LIKE :Prenom');
 // -- Préparation des variables --
 $tab_param[':Nom']    = "MARTIN"   ;
 $tab_param[':Prenom'] = "%PIERRE%" ;
 // -- Exécution de la requête préparée --
 $RequetePreparee->execute($tab_param);
 ...
?>
```

Voici son exécution :

Listing 10.30 – Exécution de MySQL_PDO_prepare2_personnes_shell.php

```
$ php MySQL_PDO_prepare2_personnes_shell.php
6   MARTIN   PIERRE-DAVID   27
7   MARTIN   PIERRE         56
```

- *Liaison des paramètres*

La classe PDOStatement propose trois méthodes pour lier les paramètres avec les variables ou données PHP : `bindParam()`, `bindValue()`, `bindColumn()`.

La méthode *bindParam()*

Elle associe un paramètre à **une variable** PHP. Lors de l'exécution de la requête préparée (sans argument), la valeur de la variable est affectée au paramètre associé.

L'association peut préciser le type de la variable *via* les constantes telles que `PDO::PARAM_INT` ou `PDO::PARAM_STR`, et indiquer la longueur de la donnée.

Avec marqueur nommé

Le programme `MySQL_PDO_prepare3_personnes_shell.php` présente l'usage de `bindParam()` sur un paramètre.

- Le paramètre « :Nom » est lié à la variable PHP `$Nom` ;
- Le type du paramètre est indiqué comme `PDO::PARAM_STR` ;
- Sa longueur est de 50 caractères.

Voici la syntaxe qui met en œuvre cette liaison :

```
$RequetePreparee->bindParam(':Nom',$Nom,PDO::PARAM_STR,50);
```

La première exécution est effectuée avec la variable `$Nom` contenant «MARTIN». La seconde est effectuée avec la variable `$Nom` contenant «DUPONT».

Listing 10.31 – Programme MySQL_PDO_prepare3_personnes_shell.php

```
<?php
 ...
 // -- Préparation de la requête --
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=:Nom');
 // -- Liaison avec le paramètre --
 $RequetePreparee->bindParam(':Nom',$Nom,PDO::PARAM_STR,50);
 $Nom="MARTIN"; // Préparation de la variable
 $RequetePreparee->execute(); // Exécution de la requête
 affiche($RequetePreparee)  ; // Affichage du résultat
 $Nom="DUPONT"; // Préparation de la variable
 $RequetePreparee->execute();// Exécution de la requête
 affiche($RequetePreparee)  ; // Affichage du résultat
 $RequetePreparee->closeCursor();// Fermeture de la requête
}
catch(Exception $e){echo 'Erreur : '.$e->getMessage();}
// -- Fonction d'affichage --
function affiche($RequetePreparee)
{echo "-- Affichage du résultat ---".PHP_EOL;
 while ($donnees = $RequetePreparee->fetch())
 {  echo $donnees['ID']."\t"    ;
    echo $donnees['Nom']."\t"   ;
    echo $donnees['Prenom']."\t";
    echo $donnees['Age'].PHP_EOL;
 }
}
?>
```

Voici son exécution :

Listing 10.32 – Excécution de MySQL_PDO_prepare3_personnes_shell.php

```
$ php MySQL_PDO_prepare3_personnes_shell.php
-- Affichage du résultat --
6   MARTIN   PIERRE-DAVID   27
7   MARTIN   PIERRE         56
16  MARTIN   ALBERT         25
-- Affichage du résultat --
1   DUPONT   JEAN           28
15  DUPONT   JEAN           54
```

Le programme `MySQL_PDO_prepare4_personnes_shell.php` montre comment appliquer `bindParam()` sur deux variables `$Nom` et `$Prenom`.

Listing 10.33 – Programme MySQL_PDO_prepare4_personnes_shell.php

```
<?php
 ...
 // -- Préparation de la requête --
```

```
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=:Nom AND Prenom LIKE :Prenom');
 // -- Liaison avec les paramètres --
 $RequetePreparee->bindParam(':Nom',$Nom,PDO::PARAM_STR,50);
 $RequetePreparee->bindParam(':Prenom', $Prenom, PDO::PARAM_STR, 50);
 // -- Préparation des variables --
 $Nom="MARTIN";
 $Prenom="%PIERRE%";
 // -- Exécution de la requête préparée --
 $RequetePreparee->execute();
 ...
?>
```

Voici son exécution :

Listing 10.34 – Exécution de MySQL_PDO_prepare4_personnes_shell.php

```
$ php MySQL_PDO_prepare4_personnes_shell.php
-- Affichage du résultat --
6   MARTIN   PIERRE-DAVID   27
7   MARTIN   PIERRE         56
```

Remarque

L'association peut se faire sur une variable inexistante. La variable doit seulement être définie au moment de l'exécution de la requête préparée.

Avec marqueur anonyme

Le marqueur anonyme « ? » appliqué à `bindParam()` sur un paramètre s'écrit :

```
// -- Préparation de la requête --
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=?');
// -- Liaison avec le paramètre --
$RequetePreparee->bindParam(1, $Nom, PDO::PARAM_STR, 50);
```

De même, avec deux paramètres, la syntaxe devient :

```
 // -- Préparation de la requête --
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=? AND Prenom LIKE ?');
 // -- Liaison avec les paramètres --
 $RequetePreparee->bindParam(1, $Nom, PDO::PARAM_STR, 50);
 $RequetePreparee->bindParam(2, $Prenom, PDO::PARAM_STR, 50);
```

La méthode *bindValue()*

Cette méthode associe *une valeur* à un paramètre. L'association peut préciser le type *via* les constantes comme `PDO::PARAM_INT` ou `PDO::PARAM_STR`.

Avec marqueur nommé

La syntaxe suivante présente l'usage de `bindValue()` sur un paramètre.

- Le paramètre « `:Nom` » est lié à la variable PHP `$Nom`.
- Le type du paramètre est indiqué comme `PDO::PARAM_STR`

Il n'y a pas d'argument indiquant la taille, puisque la liaison porte sur la valeur, la taille est donc connue.

```
$RequetePreparee->bindValue(':Nom', $Nom, PDO::PARAM_STR);
```

Remarque

binValue() lie une valeur et un paramètre. Si la valeur est une variable, elle doit être définie avant l'association. Le changement du contenu de la variable n'a aucun impact sur une nouvelle exécution car la liaison est effectuée sur sa valeur au moment de l'association.

Le programme `MySQL_PDO_prepare6_personnes_shell.php` présente l'usage de `bindValue()` sur deux paramètres.

Listing 10.35 – Programme MySQL_PDO_prepare6_personnes_shell.php

```
<?php
 ...
 // -- Préparation de la requête --
 $RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
 personnes WHERE Nom=:Nom AND Prenom LIKE :Prenom');
 // -- Préparation des variables --
 $Nom="MARTIN";
 $Prenom="%PIERRE%";
 // -- Liaison avec les paramètres --
 $RequetePreparee->bindValue(':Nom', $Nom, PDO::PARAM_STR);
 $RequetePreparee->bindValue(':Prenom', $Prenom, PDO::PARAM_STR);
 // -- Exécution de la requête préparée --
 $RequetePreparee->execute();
 ...
?>
```

Avec marqueur anonyme

Les syntaxes suivantes utilisent le marqueur anonyme « ? » pour un paramètre.

```
// -- Préparation de la requête --
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=?');
// -- Liaison avec le paramètre --
$RequetePreparee->bindValue(1, $Nom, PDO::PARAM_STR);
```

Les syntaxes suivantes utilisent deux paramètres.

```
// -- Préparation de la requête --
$RequetePreparee = $bdd->prepare('SELECT ID,Nom,Prenom,Age FROM
personnes WHERE Nom=? AND Prenom LIKE ?');
// -- Liaison avec les paramètres --
$RequetePreparee->bindValue(1, $Nom, PDO::PARAM_STR);
$RequetePreparee->bindValue(2, $Prenom, PDO::PARAM_STR);
```

La méthode *bindColumn()*

Elle associe *une colonne* de la table à une variable PHP. L'appel de `fetch()` ou `fetchAll()` met à jour la variable avec la valeur de la colonne associée. Si une table SQL est constituée par 4 colonnes, ID, Nom, Prenom et Age, et que la première est associée à la variable PHP `$ID`, à chaque itération d'une boucle `while` utilisant `fetch` traitant d'une ligne de la table, la variable `$ID` recevra la valeur de la colonne `ID` de la ligne en cours de traitement. La colonne à associer peut être indiquée par son numéro comme dans les deux syntaxes suivantes :

```
$RequetePreparee->bindColumn(1, $ID, PDO::PARAM_INT);
$RequetePreparee->bindColumn(2, $Nom,PDO::PARAM_STR);
```

Ou bien par son nom comme dans ces deux autres syntaxes :

```
$RequetePreparee->bindColumn('Prenom', $Prenom, PDO::PARAM_STR);
$RequetePreparee->bindColumn('Age', $Age,PDO::PARAM_INT);
```

Le programme `MySQL_PDO_prepare7_personnes_shell.php` présente `bindColumn()`. La constante `PDO::FETCH_BOUND` indique à `fetch()` de retourner la valeur TRUE et d'assigner les valeurs des colonnes aux variables PHP liées.

Listing 10.36 – Programme MySQL_PDO_prepare7_personnes_shell.php

```
<?php
 ...
 // -- Préparation de la requête --
 $requete_sql = 'SELECT ID,Nom,Prenom,Age FROM personnes';
 $RequetePreparee = $bdd->prepare($requete_sql);
 $RequetePreparee->execute();// Exécution de la requête
 // -- Lien par les numéros de colonnes --
 $RequetePreparee->bindColumn(1, $ID, PDO::PARAM_INT);
 $RequetePreparee->bindColumn(2, $Nom,PDO::PARAM_STR);
 // -- Lien par les noms de colonnes --
 $RequetePreparee->bindColumn('Prenom', $Prenom, PDO::PARAM_STR);
 $RequetePreparee->bindColumn('Age', $Age,PDO::PARAM_INT);
 // -- Affichage du résultat de la requête --
 while ($ligne = $RequetePreparee->fetch(PDO::FETCH_BOUND))
 {  echo $ID."\t"    ;
    echo $Nom."\t"   ;
    echo $Prenom."\t";
    echo $Age.PHP_EOL;
 }
 $RequetePreparee->closeCursor();
}
catch(Exception $e){echo 'Erreur : '.$e->getMessage();}
?>
```

Les problèmes de sécurité

Les données provenant d'une interface web doivent être considérées comme « non sûres », et doivent être contrôlées. Si elles sont numériques, il faut les traiter avec des fonctions comme `intval()`ou `foatval()` pour forcer leur typage. Si elles sont de type texte, cette méthode de contrôle n'est plus applicable.

Comme avec l'injection HTML, l'utilisateur peut saisir dans un champ texte des syntaxes SQL à « injecter » dans les requêtes, et récupérer des informations sensibles. C'est *l'injection SQL*. Les syntaxes délictueuses contiennent généralement des apostrophes pour terminer les données des requêtes du programme et ajouter une syntaxe SQL complémentaire, modifiant la requête elle-même.

Pour éviter cette faille de sécurité, il faut s'assurer que les données passées aux requêtes *via* PDO sont bien assimilées à des « données » et ne peuvent en aucun cas être interprétées comme des parties de requêtes SQL.

- *Sécurisation par quote()*

La solution la plus « classique » modifie la donnée de type chaînes de caractères avant de l'utiliser dans une requête, en l'entourant d'apostrophes (quotes) et en déspécialisant (neutralisant) les caractères spéciaux. La classe PDO propose la méthode `quote()` qui effectue ce traitement. Sa syntaxe est :

```
$NomQuote=$bdd->quote($Saisie);
```

où `$Saisie` est la variable chaîne de caractères contenant la saisie de l'utilisateur. Mais à cette méthode « simpliste » il est préférable d'utiliser les requêtes préparées, mieux protégées.

- *Sécurisation par requête préparée*

Les requêtes préparées évitent la faille de sécurité par « Injection SQL ». Pour cela, il faut respecter les règles suivantes :

1. Ne passer aucune donnée saisie par l'utilisateur à la méthode `prepare()`.
2. Passer les valeurs saisies par l'utilisateur seulement au moment de l'exécution.
3. Utiliser les méthodes `bindParam()`, `bindValue()` ou `bindColumn()` qui typent les données afin de les sécuriser totalement.

La gestion des exceptions

Principe

La notion d'exception a été présentée à la section 9.4 du chapitre 9. Nous détaillons dans cette partie la classe `PDOException` et la méthode `setAttribute()`.

La classe PDOException

`PDOException` est une extension de la classe `RuntimeException`, elle-même étant une extension de la classe `Exception`. Ses méthodes usuelles sont :

- `PDO::errorinfo()` : elle retourne un tableau numérique, relatif à la dernière opération sur la base de données ;
- `PDOStatement::errorinfo()` : elle retourne la même information que la méthode précédente mais à partir d'un objet de la classe `PDOStatement`. Elle est applicable à un objet obtenu avec la méthode `prepare()`, mais pas avec `query()` qui retourne NULL en cas d'erreur ;
- `Exception::getCode()` : elle récupère le code de l'exception ;
- `Exception::getMessage()` : elle récupère le message de l'exception.

setAttribute()

Cette méthode configure un attribut du gestionnaire de base de données PDO. Elle admet deux paramètres : le nom de l'attribut, et sa valeur. Sa syntaxe est :

```
$bdd = new PDO('mysql:host=localhost;dbname=CoursPHP','root', 'xxxx');
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Le tableau 10.2 présente quelques attributs et les valeurs possibles.

Tableau 10.2 – Quelques attributs de setAttribute()

Attribut	Valeur	Signification
PDO::ATTR_AUTOCOMMIT	0 (faux) 1 (vrai)	Activation de l'autocommit
PDO::ATTR_CASE	PDO::CASE_LOWER	Colonnes en minuscules
PDO::ATTR_CASE	PDO::CASE_UPPER	Colonnes en majuscules
PDO::ATTR_ERRMODE	PDO::CASE_SILENT	Assigne le code d'erreur
PDO::ATTR_ERRMODE	PDO::CASE_WARNING	Émet un E_WARNING
PDO::ATTR_ERRMODE	PDO::CASE_EXCEPTION	Émet une exception
PDO::ATTR_DEFAULT_FETCH_MODE	PDO::FETCH_ASSOC, …	Définit le mode de récupération par défaut

Par défaut, l'attribut PDO::ATTR_ERRORMODE est en mode silence (PDO::CASE_SILENT).

Pour générer une exception quand une erreur se produit avec PDO, il faut donc utiliser la syntaxe suivante :

```
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Exercices

10.1 Reprenez l'exercice n° 1 du chapitre 8 pour gérer les données dans la table « personnes » de la base de données « CoursPHP ». Chaque personne sera caractérisée par son **identifiant (ID)**, son **nom**, son **prénom** et son **âge**. Son identifiant (ID) sera la clef primaire auto-incrémentée. Voici le menu principal :

```
-1- Saisie d'une liste de personnes
-2- Affichage de toutes les personnes
-3- Supprimer une personne
-4- Modifier une personne
-5- Recherche de personnes, d'après un champ
-0- Quitter
Choix (1,2,3,...) :
```

Chaque choix appellera une procédure ou une fonction à développer séparément. L'écriture de chaque fonction constitue un exercice à part entière.

Choix 1 : Voici un exemple de la saisie des personnes :

```
Choix (1,2,3,...) : 1
Entrez un un nom, un prénom et un âge (ex : Dupont;Jean;28) :
dupont;jean;28
Entrez un un nom, un prénom et un âge (ex : Dupont;Jean;28) : jacquenod
; jean christophe ; 54
Entrez un un nom, un prénom et un âge (ex : Dupont;Jean;28) : murcianc
; carole; 44
Entrez un un nom, un prénom et un âge (ex : Dupont;Jean;28) :
```

Les contraintes à respecter sont :

- une ligne vide provoque une fin de saisie ;
- la saisie des noms et prénoms composés devra être possible ;
- les noms et prénoms seront normalisés en majuscules, sans accent, ni valeur numérique, ni apostrophe. Les espaces multiples seront remplacés par un seul espace, puis chaque espace d'un nom ou d'un prénom composé sera remplacé par un tiret « - ». Par exemple « Léry » sera normalisé en « LERY », « dE la fONtaine » deviendra « DE-LA-FONTAINE » ;
- l'identifiant sera automatiquement attribué par la table « personnes » ;
- si un des éléments est vide, ou si l'âge n'est pas un numérique, un message d'erreur apparaît, et la personne n'est pas traitée ;
- les informations d'une personne saisie sont rangées dans un tableau associatif `$une_personne[]` (variable locale) avec les étiquettes « Nom », « Prenom », « Age », puis chaque personne sera ensuite rangée dans un tableau numérique de personnes, `$tab_personnes[]` (variable locale) ;

- à la fin de la saisie, si le tableau `$tab_personnes[]` n'est pas vide, chacun de ses éléments sera inséré dans la table « personnes ».

La figure 10.20 présente le tableau des personnes.

$tab_personnes

colonnes

Lignes		*Nom*	*Prenom*	*Age*
	0	DUPONT	JEAN	28
	1	JACQUENOD	JEAN-CHRISTOPHE	54
	2	MURCIANC	CAROLE	44

Figure 10.20 – Tableaux des personnes.

Choix 2 : Voici un exemple de l'affichage des personnes :

```
Choix (1,2,3,...) : 2
-----------------------------------------------------------------
 ID Nom                  Prénom                 Age
-----------------------------------------------------------------
 1  DUPONT               JEAN                   28
 2  JACQUENOD            JEAN-CHRISTOPHE        54
...
```

L'affichage utilisera une procédure ayant un tableau à afficher en argument.

Choix 3 : Voici un exemple de la suppression d'une personne :

```
Choix (1,2,3,...) : 3
Critère de recherche :
    -a- Identifiant (ID)
    -b- Nom
    -c- Prénom
    -d- Age
Entrez le critère (a,b,c ou d) : b
Nom         : dupont
-----------------------------------------------------------------
 Liste des personnes trouvées
-----------------------------------------------------------------
 ID Nom                  Prénom                 Age
-----------------------------------------------------------------
 1  DUPONT               JEAN                   28
 19 DUPONT-DE-NEMOURS    JEAN-CHARLES           38
-----------------------------------------------------------------
Sélectionnez l'ID de la personne à supprimer : 19
-----------------------------------------------------------------
    ID Nom                  Prénom                 Age
-----------------------------------------------------------------
    19 DUPONT-DE-NEMOURS    JEAN-CHARLES           38
-----------------------------------------------------------------
Confirmez la suppression (o/n) : o
DUPONT-DE-NEMOURS  JEAN-CHARLES supprimé
```

Les contraintes à respecter sont :

- la recherche sera multicritère ;
- le résultat de la recherche sera affiché ;
- la personne sélectionnée sera supprimée, après confirmation.

Choix 4 : Voici un exemple de la modification d'une personne :

```
Choix (1,2,3,...) : 4
Critère de recherche :
    -a- Identifiant (ID)
    -b- Nom
    -c- Prénom
    -d- Age
Entrez le critère (a,b,c ou d) : c
Prénom      : jean
-----------------------------------------------------------------
Liste des personnes trouvées
-----------------------------------------------------------------
 ID Nom                  Prénom                Age
-----------------------------------------------------------------
 1  DUPONT               JEAN                  28
 2  JACQUENOD            JEAN-CHRISTOPHE       54
 4  DE-LA-RUE            JEAN-PAUL             27
-----------------------------------------
Sélectionnez l'ID de la personne à modifier : 4
-----------------------------------------------------------------
    ID Nom                  Prénom                Age
-----------------------------------------------------------------
     4 DE-LA-RUE            JEAN-PAUL              27
-----------------------------------------------------------------
Confirmez la modification (o/n) : o
Saisissez les nouvelles informations (vide pour inchangé) !
Nom    actuel : DE-LA-RUE     Nouveau Nom    : la rue
Prénom actuel : JEAN-PAUL     Nouveau Prénom : christophe
Age    actuel : 27            Nouvel Age     :
Pas de modification de l'âge
LA-RUE  CHRISTOPHE modifié
```

Les contraintes à respecter sont :

- la recherche sera multicritère ;
- le résultat de la recherche sera affiché ;
- chaque champ peut être modifié. Une saisie vide laisse la valeur inchangée.

Choix 5 : Voici un exemple de la recherche de personnes d'après un champ :

```
Choix (1,2,3,...) : 5
Critère de recherche :
    -a- Identifiant (ID)
```

```
    -b- Nom
    -c- Prénom
    -d- Age
Entrez le critère (a,b,c ou d) : c
Prénom      : jean
-----------------------------------------------------------------
 Liste des personnes trouvées
-----------------------------------------------------------------
 ID  Nom                 Prénom               Age
-----------------------------------------------------------------
 1   DUPONT              JEAN                 28
 2   JACQUENOD           JEAN-CHRISTOPHE      54
```

Les contraintes à respecter sont :

- le critère (a, b, c, d) sera saisi, puis la valeur à rechercher pour ce champ ;
- ce choix appellera la fonction `Recherche_sur_critere()`, commune aux choix 3 et 4, avec les deux arguments : le critère de la recherche soit la lettre saisie (« a » pour ID, « b » pour nom, « c » pour prenom, « d » pour age), puis la valeur à rechercher ;
- la recherche sur le nom ou le prénom retournera les enregistrements dont le champ contient le texte saisi ;
- la fonction recherche retourne un tableau de personnes ;
- l'affichage du résultat utilisera la fonction d'affichage du choix 2.

Choix 0 : Voici un exemple de la fin du programme :

```
Choix (1,2,3,...) : 0
Au revoir !
```

Solutions

10.1 Le programme `MySQL_PDO_liste_personnes_shell.php` présente le menu principal. Chaque choix appelle une fonction spécifique.

Listing 10.37 – Programme MySQL_PDO_liste_personnes_shell.php

```
<?php
include './INCLUDE/MySQL_include_param_dbb.php';
include './INCLUDE/MySQL_include_sprog_commun_shell.php';
include './INCLUDE/MySQL_include_sprog_liste_personnes_shell.php';
$choix=-1               ;
// == Menu ==
while ($choix !=0)
```

```
{ $choix=-1;
  echo PHP_EOL;
  echo "-1- Saisie d'une liste de personnes".PHP_EOL;
  echo "-2- Affichage de toutes les personnes".PHP_EOL;
  echo "-3- Supprimer une personne".PHP_EOL;
  echo "-4- Modifier une personne".PHP_EOL;
  echo "-5- Recherche de personnes, d'après un champ".PHP_EOL;
  echo "-0- Quitter".PHP_EOL;
  echo "Choix (1,2,3,...) : ";
  fscanf(STDIN,"%d",$choix);
  switch($choix)
  { case 1 : Saisie()    ; break ;
    case 2 : Affichage() ; break ;
    case 3 : Supprimer() ; break ;
    case 4 : Modifier()  ; break ;
    case 5 : $tab_critere=choix_critere();
             if (count($tab_critere) != 0)
             {$critere = $tab_critere[0];
              $ValRech = $tab_critere[1];
              unset($tab_resultat);
              $tab_resultat=Recherche_sur_critere($critere,$ValRech);
              if (count($tab_resultat) == 0)
                  echo "Aucune personne trouvée avec ce critère".PHP_EOL;
              else
                  affichage_liste_personnes('Liste des personnes
                  trouvées',$tab_resultat);
             }
             break;
    case 0 : quitter()   ; break ;
    default :echo "Choix impossible !".PHP_EOL ; break ;
  }
}
?>
```

Le fichier `MySQL_include_sprog_liste_personnes_shell.php` contient les fonctions appelées directement dans le programme principal.

Listing 10.38 – MySQL_include_sprog_liste_personnes_shell.php

```
<?php
// == Saisie d'une liste de personnes ==
function Saisie()
{global $TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES, $LOGIN_ADM,$MDP_ADM;
 // -- Saisie des données dans un tableau $tab_personnes --
 $entree="saisie non vide";
 while (!empty($entree))
 {echo "Entrez un un Nom, un Prénom et un Age (ex : Dupont;Jean;28) : ";
  $entree=fgets(STDIN) ;
  $entree=trim($entree); //Suppression espaces (début,fin)
  if (!empty($entree))   // Vérification saisie non vide
  {$erreur_saisie=false;
```

```
    // Rangement dans le tableau associatif
    list($une_personne['Nom'],$une_personne['Prenom'],$une_
    personne['Age'])=explode(';',$entree);
    $indice=count($une_personne);
    $nb_champs=$indice+1;
    if ($indice != 3)
    {echo "  Erreur : La saisie ne contient $nb_champs champs ! ".PHP_EOL;
     $erreur_saisie=true;
    }
    else
    {// Normalisation et vérification de chaque champ
     foreach($une_personne as $etiquette => $val_champ)
     {if (($etiquette == 'Nom') || ($etiquette == 'Prenom') )
      {$une_personne[$etiquette]=normalisation_Nom($val_champ);
      }
      if  ($etiquette == 'Age')
      {$val_champ=intval($val_champ)       ;
       $une_personne[$etiquette]=$val_champ;
       if ($val_champ <= 0)
       {echo "  Erreur : Valeur du champ $etiquette doit être un entier
       positif ! ".PHP_EOL;
        $erreur_saisie=true;
       }
      }
      if (empty($val_champ))
      {echo "  Erreur : Valeur du champ $etiquette  est non valide !
      ".PHP_EOL;
       $erreur_saisie=true;
      }
     }
    }
    if (!$erreur_saisie)
    { // Renverse $une_personne car ordre inverse de list
     $tab_personnes[]=array_reverse($une_personne);
    }
    else
    {echo "  Erreur : Merci de saisir à nouveau les informations !".PHP_EOL;
    }
   }
  }
  // -- Insertion des données dans $tab_personnes --
  if (count($tab_personnes) != 0)
  {try
   {// -- Contexte pour le message d'erreur --
    $contexte="Problème d'insertion dans la  table.";
    // -- Affichage de la liste des personnes  --
    // -- Connexion de la base de données --
    $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
           $MDP_ADM,array(PDO::ATTR_PERSISTENT => true));
    // -- Initialisation des Exceptions PDO  --
    $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
    $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
    // -- Préparation de la requête --
    $requete_sql="INSERT INTO ".$TABLEPERSONNES." (Nom,Prenom,Age)
    VALUES (:Nom,:Prenom,:Age)";
    $RequetePreparee = $bdd->prepare($requete_sql);
    // -- Liaison avec les paramètres --
    $RequetePreparee->bindParam(':Nom',$Nom,PDO::PARAM_STR,30);
    $RequetePreparee->bindParam(':Prenom', $Prenom, PDO::PARAM_STR, 30);
    $RequetePreparee->bindParam(':Age',$Age,PDO::PARAM_INT,3);
    // -- Insertion de chaque personne du tableau --
    foreach ($tab_personnes as $une_personne)
    {// Importe les variables dans la table des symboles
     extract($une_personne)     ;
     $RequetePreparee->execute(); // Exécution
    }
   }
   catch(Exception $e)
   {echo $contexte.PHP_EOL;
    echo 'Erreur : '.$e->getMessage().PHP_EOL;
   }
  }
  else echo "Aucune personne à ajouter !".PHP_EOL;
}
// == Affichage de la liste des personnes ==
function Affichage()
{global $TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_ADM,$MDP_ADM;
 try
 {// -- Contexte pour le message d'erreur --
  $contexte="Problème de requête sur la table.";
  // -- Affichage de la liste des personnes  --
  // -- Connexion de la base de données --
  $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
        $MDP_ADM,array(PDO::ATTR_PERSISTENT => true));
  // -- Initialisation des Exceptions PDO  --
  $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
  // -- Préparation de la requête --
  $requete_sql="SELECT * FROM ".$TABLEPERSONNES;
  $RequetePreparee = $bdd->prepare($requete_sql);
  // -- Exécution de la requête préparée --
  $RequetePreparee->execute();
  // -- Retourne un tableau associatif --
  $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
  // -- Boucle de traitement de chaque personne --
  $tab_personnes=$RequetePreparee->fetchAll();
  sort($tab_personnes); // Tri du tableau et retrait doublons
  $tab_personnes=array_unique($tab_personnes,SORT_REGULAR);
  $nbpersonnes=count($tab_personnes);
  if ($nbpersonnes == 0)
   echo "Aucune personne à afficher !".PHP_EOL;
  else
```

```
   {Affichage_liste_personnes("Liste des personnes",$tab_personnes);
   }
   $RequetePreparee->closeCursor(); // Fermeture de la requête
  }
  catch(Exception $e)
  {echo $contexte.PHP_EOL;
   echo 'Erreur : '.$e->getMessage().PHP_EOL;
  }
}
// == Suppression d'une personne ==
function Supprimer()
{global $TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_ADM,$MDP_ADM;
 // -- Menu des critères --
 $tab_critere=choix_critere();
 if (count($tab_critere) != 0)
 {$critere = $tab_critere[0];
  $ValRech = $tab_critere[1];
  // -- Recherche multi critères --
  $tab_resultat=Recherche_sur_critere($critere,$ValRech);
  if (count($tab_resultat) == 0)
  {echo "Aucune personne trouvée avec ce critère".PHP_EOL;
  }
  else
  {affichage_liste_personnes("Liste des personnes trouvées",$tab_resultat);
   echo "Sélectionnez l'ID de la personne à supprimer : ";
   fscanf(STDIN,"%d", $numero_ID);
   if (empty($numero_ID)) echo "Aucune sélection".PHP_EOL;
   else
   {// -- On récupère la colonne des ID --
    $colone_ID=array_column($tab_resultat,'ID');
    // -- On récupère l'indice numérique de la case --
    $numcase=array_search($numero_ID,$colone_ID);
    // -- Array_search() retourne le numéro de la case --
    // -- Ou false en cas d'échec --
    if ($numcase === false)
    {echo "ID $numero_ID non trouvé dans la sélection !".PHP_EOL;
    }
    else
    {$une_personne=$tab_resultat[$numcase];
     extract($une_personne,EXTR_OVERWRITE);
     echo "----------------------------------------".PHP_EOL;
     fprintf(STDOUT,"%6s %-20s %-20s %4s
     %s","ID","Nom","Prénom","Age",PHP_EOL);
     echo "----------------------------------------".PHP_EOL;
     fprintf(STDOUT,"%6d %-20s %-20s %4d %s",$ID,$Nom,$Prenom,$Age,PHP_
     EOL)    ;
     echo "----------------------------------------".PHP_EOL;
     echo "Confirmez la suppression (o/n) : ";
     fscanf(STDIN,"%s",$reponse);
     if ($reponse == "o")
     {// -- On supprime dans la table personnes --
```

```
       try
       {// -- Contexte pour le message d'erreur --
        $contexte="Problème de suppression dans la  table.";
        // -- Affichage de la liste des personnes  --
        // -- Connexion de la base de données --
        $bdd = new PDO($TYPE_
        DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
               $MDP_ADM,array(PDO::ATTR_PERSISTENT => true));
        // -- Initialisation des Exceptions PDO  --
        $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
        // -- Préparation de la requête --
        $requete_sql="DELETE FROM ".$TABLEPERSONNES." WHERE ID=:ID";
        $RequetePreparee = $bdd->prepare($requete_sql);
        // -- Liaison avec les paramètres --
        $RequetePreparee->bindParam(':ID', $ID, PDO::PARAM_INT);
        // -- Suppression dans la table --
        $RequetePreparee->execute(); // Exécution
        echo "$Nom  $Prenom supprimé".PHP_EOL    ;
       }
       catch(Exception $e)
       {echo $contexte.PHP_EOL;
        echo 'Erreur : '.$e->getMessage().PHP_EOL;
       }
      }
      else  echo "Aucune donnée supprimée".PHP_EOL;
     }
    }
   }
  }
}
// == Modification d'une personne ==
function Modifier()
{global $TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_ADM,$MDP_ADM;
 // -- Menu des critères --
 $tab_critere=choix_critere();
 if (count($tab_critere) != 0)
 {$critere = $tab_critere[0];
  $ValRech = $tab_critere[1];
  // -- Recherche multi critères --
  $tab_resultat=Recherche_sur_critere($critere,$ValRech);
  if (count($tab_resultat) == 0)
  {echo "Aucune personne trouvée avec ce critère".PHP_EOL;
  }
  else
  {affichage_liste_personnes("Liste des personnes trouvées",$tab_resultat);
   echo "Sélectionnez l'ID de la personne à modifier : ";
   fscanf(STDIN,"%d", $numero_ID);
   if (empty($numero_ID)) echo "Aucune sélection".PHP_EOL;
   else
   {// -- On récupère la colonne des ID --
```

```
$colone_ID=array_column($tab_resultat,'ID');
// -- On récupère l'indice numérique de la case --
$numcase=array_search($numero_ID,$colone_ID);
if ($numcase === false)
{echo "ID $numero_ID non trouvé dans la sélection !".PHP_EOL;
}
else
{$une_personne=$tab_resultat[$numcase];
 extract($une_personne,EXTR_OVERWRITE);
 echo "----------------------------------------".PHP_EOL;
 fprintf(STDOUT,"%6s %-20s %-20s %4s
 %s","ID","Nom","Prénom","Age",PHP_EOL);
 echo "----------------------------------------".PHP_EOL;
 fprintf(STDOUT,"%6d %-20s %-20s %4d %s",$ID,$Nom,$Prenom,$Age,PHP_
 EOL)    ;
 echo "----------------------------------------".PHP_EOL;
 echo "Confirmez la modification (o/n) : ";
 fscanf(STDIN,"%s",$reponse);
 if ($reponse == "o")
 {// -- On initialise les différents éléments de la requete SQL
 UPDATE --
  $MAJ_Nom    ='';
  $MAJ_Prenom ='';
  $MAJ_Age    ='';
  echo "Saisissez les nouvelles informations (vide pour inchangé)
  !".PHP_EOL;
  // -- Saisie du nom --
  fprintf(STDOUT, "Nom    actuel : %-20s Nouveau Nom    : ",$Nom);
  $entree=fgets(STDIN) ;
  // Normalisation du nom
  $entree=normalisation_nom($entree);
  if (!empty($entree)) // Vérification saisie non vide
  {$Nom=$entree ;
   $MAJ_Nom =',Nom=:Nom';
  }
  else
  {echo "Pas de modification du nom".PHP_EOL;
  }
  // -- Saisie du prénom --
  fprintf(STDOUT, "Prénom actuel : %-20s Nouveau Prénom : ",$Prenom);
  $entree=fgets(STDIN) ;
  // Normalisation du prénom
  $entree=normalisation_nom($entree);
  if (!empty($entree)) // Vérification saisie non vide
  {$Prenom=$entree ;
   $MAJ_Prenom =',Prenom=:Prenom';
  }
  else
  {echo "Pas de modification du prénom".PHP_EOL;
  }
  // -- Saisie de l'âge --
```

```
fprintf(STDOUT, "Age    actuel : %-20d Nouvel Age     : ",$Age);
$entree=fgets(STDIN) ;
$entree=trim($entree); //Suppression espaces (début,fin)
if (!empty($entree))   // Vérification saisie non vide
{$entree=intval($entree)       ;
 if ($entree == 0)
 {echo "  Erreur : Valeur du champ âge doit être un entier ! =>
 Pas de modification ".PHP_EOL;
 }
 else // On met à jour la donnée
 {$Age=$entree ;
  $MAJ_Age =',Age=:Age';
 }
}
else
{echo "Pas de modification de l'âge".PHP_EOL;
}
// -- Un des champs à été mis à jour : le traitement --
if (!empty($MAJ_Nom) || !empty($MAJ_Prenom) || !empty($MAJ_Age))
{// -- Préparation de la requête --
 $requete_sql='UPDATE '.$TABLEPERSONNES.' SET '.$MAJ_Nom.$MAJ_
 Prenom.$MAJ_Age.' WHERE ID='.$ID;
 // -- On supprime les espaces multiples --
 $requete_sql= preg_replace('/\s{2,}/',' ',$requete_sql);
 // -- Supprime le caractère apostrophe après SET --
 $requete_sql= str_replace('SET ,','SET ',$requete_sql);
 // -- Modification dans la table personnes --
 try
 {// -- Contexte pour le message d'erreur --
  $contexte="Problème de modification dans la  table.";
  // -- Affichage de la liste des personnes  --
  // -- Connexion de la base de données --
  $bdd = new PDO($TYPE_
  DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
         $MDP_ADM,array(PDO::ATTR_PERSISTENT => true));
  // -- Initialisation des Exceptions PDO  --
  $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $bdd->exec("SET CHARACTER SET utf8"); //Codage en UTF8
  // -- Préparation de la requête --
  $RequetePreparee = $bdd->prepare($requete_sql);
  // -- Liaison avec les paramètres --
  if (!empty($MAJ_Nom))
      $RequetePreparee->bindParam(':Nom', $Nom, PDO::PARAM_STR,30);
  if (!empty($MAJ_Prenom))
       $RequetePreparee->bindParam(':Prenom', $Prenom, PDO::PARAM_
       STR,30);
  if (!empty($MAJ_Age))
      $RequetePreparee->bindParam(':Age', $Age, PDO::PARAM_INT,3);
  // -- Modification de la table --
  $RequetePreparee->execute();
  echo "$Nom  $Prenom modifié".PHP_EOL    ;
```

```
        }
        catch(Exception $e)
        {echo $contexte.PHP_EOL;
         echo 'Erreur : '.$e->getMessage().PHP_EOL;
        }
       }
       else echo "Aucun champ n'a été modifié".PHP_EOL;
      }
      else echo "Aucune donnée modifiée".PHP_EOL;
     }
    }
   }
  }
}
// == Fonction quitter ==
function Quitter()
{echo "Au revoir !".PHP_EOL;}
// == Outil de sélection du choix pour la recherche ==
function choix_critere()
{ $tab_critere_retour=array();
  echo "Critère de recherche : ".PHP_EOL;;
  echo "    -a- Identifiant (ID)".PHP_EOL;
  echo "    -b- Nom".PHP_EOL;
  echo "    -c- Prénom".PHP_EOL;
  echo "    -d- Age".PHP_EOL;
  echo "Entrez le critère (a,b,c ou d) : ";
  fscanf(STDIN,"%s",$critere);
  $critere=$critere[0];
  switch ($critere)
  { case 'a' : echo "Identifiant : "; break;
    case 'b' : echo "Nom         : "; break;
    case 'c' : echo "Prénom      : "; break;
    case 'd' : echo "Age         : "; break;
    default  : echo "Choix erroné !".PHP_EOL; break;
  }
  if (($critere >='a') && ($critere <='d'))
  {$ValRech=fgets(STDIN);
   trim($ValRech);
   $tab_critere_retour[0]=$critere;
   $tab_critere_retour[1]=$ValRech;
  }
 return $tab_critere_retour;
}
// == Recherche sur critère ==
function Recherche_sur_critere($CritRech,$ValRech)
{global $TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_ADM,$MDP_ADM;
 $tab_personnes_retourne=array();
 switch($CritRech)
 {case 'a':$Nom_champ='ID'    ;$ValRech=intval($ValRech);break;
  case 'b':$Nom_champ='Nom'   ;$ValRech="%".normalisation_
  Nom($ValRech)."%";break;
```

```
 case 'c':$Nom_champ='PreNom';$ValRech="%".normalisation_
 Nom($ValRech)."%";break;
 case 'd':$Nom_champ='Age'   ;$ValRech=intval($ValRech);break;
}
try
{// -- Contexte pour le message d'erreur --
 $contexte="Problème de requête sur la table.";
 // -- Connexion de la base de données --
 $bdd = new PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
        $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
 // -- Initialisation des Exceptions PDO  --
 $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
 $bdd->exec("SET CHARACTER SET utf8"); // Codage en UTF8
 // -- Préparation de la requête --
 $requete_sql='SELECT * FROM '.$TABLEPERSONNES.' WHERE '.$Nom_champ.'
 LIKE :ValRech';
 $RequetePreparee = $bdd->prepare($requete_sql);
 // -- Liaison avec les paramètres --
 switch($CritRech)
 {case 'a': case 'd' : $RequetePreparee->bindParam(':ValRech',
 $ValRech, PDO::PARAM_INT, 10);break;
  case 'b': case 'c' : $RequetePreparee->bindParam(':ValRech',
  $ValRech, PDO::PARAM_STR, 50);break;
 }
 // -- Exécution de la requête préparée --
 $RequetePreparee->execute();
 // -- Retourne un tableau associatif --
 $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
 unset($tab_personnes_retourne);
 $tab_personnes_retourne=$RequetePreparee->fetchAll();
 sort($tab_personnes_retourne);
 $tab_personnes_retourne=array_unique($tab_personnes_retourne,SORT_
 REGULAR);
 }
catch(Exception $e)
{ echo $contexte.PHP_EOL;
  echo 'Erreur : '.$e->getMessage().PHP_EOL;
}
return $tab_personnes_retourne;
}
?>
```

Le fichier `MySQL_include_sprog_commun_shell.php` contient les fonctions outils. Les fonctions `supprime_accents()`, `normalisation_nom()`, `normalisation_numerique()` sont présentées au listing 10.20, leur contenu est remplacé par des « ... ».

Listing 10.39 – MySQL_include_sprog_commun_shell.php

```
<?php
// == Fonction d'affichage du tableau ==
function affichage_liste_personnes($texte,$tab_mixte)
{if (count($tab_mixte)==0)
        throw new Exception('Aucun élément à afficher.');
 // -- Affichage entête du tableau --
 reset($tab_mixte);
 $une_personne=current($tab_mixte);
 $liste_champs=array_keys($une_personne);
 echo "-----------------------------------------".PHP_EOL;
 echo " $texte".PHP_EOL;
 echo "-----------------------------------------".PHP_EOL;
 foreach($liste_champs as $NomChamp)
   echo "$NomChamp\t";
 echo PHP_EOL;
 echo "-----------------------------------------".PHP_EOL;
 // -- Boucle de traitement de chaque personne --
 foreach ($tab_mixte as $une_personne)
 {// -- On affiche le contenu des champs --
  foreach($liste_champs as $NomChamp)
    echo $une_personne[$NomChamp]."\t";
  echo PHP_EOL;
 }
 echo "-----------------------------------------".PHP_EOL;
}
// == Fonction outil de suppression des accents ==
function supprime_accent($chaine) { ... }
// == Fonction outil de normalisation des noms  ==
function normalisation_nom($chaine) { ... }
// == Fonction outil de normalisation des numériques  ==
function normalisation_numerique($numero) { ... }
?>
```

Le fichier `MySQL_include_param_dbb.php` contient les paramètres de connexion à la base de données.

Listing 10.40 – MySQL_include_param_dbb.php

```
<?php
// == Paramètres de connexion à la base de données ==
$TYPE_DBB="mysql";
$SERVEUR="localhost";
$BASEDD="CoursPHP";
$TABLEPERSONNES="personnes";
$LOGIN_ADM="personnesadm";
$MDP_ADM="xxxx";
?>
```