10-1

COMPLEMENTS MYSQL

PLAN

OBJECTIF

- Maîtriser les syntaxes de base du langage SQL et le mode transactionnel.

10-1.1 PRESENTATION

Ce chapitre complémentaire détaille les requêtes du langage SQL servant de support aux programmes PHP du chapitre 10. Il aborde également le mode transactionnel et en présente à la fois les syntaxes SQL et sa programmation PHP via PDO.

10-1.2 LE LANGAGE SQL

Cette section présente les syntaxes SQL en ligne de commandes, sous le moniteur MySQL, comme une suite de manipulations sans détailler chaque instruction. Un tutoriel complet du langage SQL est disponible à l'URL : http://sql.sh

Accès au serveur de Base de données

La syntaxe suivante présente l'accès au SGBD MySQL sur le poste local (localhost). Dans notre exemple, le texte ***mot_de_passe*** doit être remplacé par le mot de passe effectif. S'il est indiqué sur la ligne de commande (-pmot_de_passe). Un message prévient que cette méthode d'accès n'est pas sécurisée.

```
$ mysql --no-defaults -u root -pmot_de_passe -h localhost
Warning: Using a password on the command line interface can be
insecure.
...
```

La syntaxe suivante est à privilégier. Le mot de passe est saisi dans un second temps. L'écho de sa frappe n'apparaît pas au moment de la saisie.

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
```

Remarque

Les syntaxes sous le moniteur SQL (après l'invite « mysql> »), sont présentées en majuscules, en respect des règles d'usage, mais elles peuvent être saisies en minuscules.

Afficher toutes les bases de données

Une fois connecté, il est possible d'afficher toutes les bases de données.

```
mysql> SHOW DATABASES;
+------------------+
|Database          |
+------------------+
|information_schema|
|cdcol             |
|mysql             |
|performance_schema|
|phpmyadmin        |
|test              |
+------------------+
6 rows in set (0,00 sec)
```

Quitter le serveur de Base de données

Pout quitter le moniteur MySQL il suffit de saisir :

```
mysql> QUIT;
Bye
```

Gestion d'une base de données

Cette section présente la **création et la suppression** d'une **base de données**.

Création

La commande suivante crée la base « CoursPHP » avec un jeu de caractères utf8.

```
mysql> CREATE DATABASE CoursPHP CHARACTER SET 'utf8';
Query OK, 1 row affected (0,00 sec)
```

Suppression

La syntaxe suivante supprime la base de données « CoursPHP ».

```
mysql> DROP DATABASE CoursPHP;
Query OK, 0 rows affected (0,01 sec)
```

Gestion d'une table

Afin de simplifier les syntaxes de création, de suppression ou d'insertion de données dans une table, on peut indiquer la base de données à utiliser.

```
mysql> USE CoursPHP;
Database changed
```

Création

La syntaxe suivante crée la table « personnes » avec quatre colonnes, « ID » (int de 11 chiffres), « NOM » (varchar de 255 caractères), « Prenom » (varchar de 255 caractères), « Age » (int de 11 chiffres). Le moteur de stockage est InnoDB. La saisie est effectuée sur plusieurs lignes.

```
mysql> CREATE TABLE IF NOT EXISTS personnes (
    -> ID int(11) NOT NULL,
    -> Nom varchar(255) NOT NULL,
    -> Prenom varchar(255) NOT NULL,
    -> Age int(11) NOT NULL
    -> ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COMMENT='Table de personnes';
Query OK, 0 rows affected (0,00 sec)
```

La clef primaire est affectée sur le champ « ID » :

```
mysql> ALTER TABLE personnes
    -> ADD PRIMARY KEY (ID);
Query OK, 0 rows affected (0,04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

La clef primaire « ID » est auto-incrémenté et sa numérotation démarre à 1 :

```
mysql> ALTER TABLE personnes
    -> MODIFY ID int(11) NOT NULL
AUTO_INCREMENT,AUTO_INCREMENT=1;
Query OK, 0 rows affected (0,01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Affichage des tables

Voici la liste des tables présentes dans la base de données courante, « CoursPHP » :

```
mysql> SHOW TABLES;
+------------------+
|Tables_in_CoursPHP|
+------------------+
|personnes         |
+------------------+
1 row in set (0,00 sec)
```

Affichage de la structure d'une table

La syntaxe pour afficher la structure de la table « personnes » de la base « Cours-PHP » est :

```
mysql> DESCRIBE CoursPHP.personnes;
+------+------------+----+---+-------+--------------+
|Field |Type        |Null|Key|Default|Extra         |
+------+------------+----+---+-------+--------------+
|ID    |int(11)     |NO  |PRI|NULL   |auto_increment|
|Nom   |varchar(255)|NO  |   |NULL   |              |
|Prenom|varchar(255)|NO  |   |NULL   |              |
|Age   |int(11)     |NO  |   |NULL   |              |
+------+------------+----+---+-------+--------------+
4 rows in set (0,00 sec)
```

La syntaxe précédente indique explicitement la base de données et la table. On aurait pu également saisir :

```
mysql> USE CoursPHP;
mysql> DESCRIBE personnes;
```

Suppression complète de la table

Voici la syntaxe pour supprimer la table « personnes » :

```
mysql> DROP TABLE personnes;
Query OK, 0 rows affected (0,01 sec)
```

Vider la table de ses données

La syntaxe suivante vide la table « personnes » de ses données sans la supprimer :

```
mysql> TRUNCATE TABLE personnes;
Query OK, 0 rows affected (0,01 sec)
```

Gestion des données

Insertion de données

L'insertion de quatre personnes dans la table « personnes » se note :

```
mysql> INSERT INTO personnes (ID, Nom, Prenom, Age) VALUES
    -> (1, 'DUPONT', 'JEAN', 28),
    -> (2, 'MARTIN', 'PIERRE', 56),
    -> (3, 'DE-LA-FONTAINE', 'JEAN', 110),
    -> (4, 'DE-LA-RUE', 'JEAN-CHARLES', 45);
Query OK, 4 rows affected (0,01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Affichage

Voici l'affichage de tous les enregistrements contenus dans la table « personnes ».

```
mysql> SELECT * FROM personnes;
+--+--------------+------------+---+
|ID|Nom           |Prenom      |Age|
+--+--------------+------------+---+
| 1|DUPONT        |JEAN        | 28|
| 2|MARTIN        |PIERRE      | 56|
| 3|DE-LA-FONTAINE|JEAN        |110|
| 4|DE-LA-RUE     |JEAN-CHARLES| 45|
+--+--------------+------------+---+
4 rows in set (0,00 sec)
```

Modification

La syntaxe suivante modifie le prénom de MARTIN (ID=2), en « PIERRE-ANDRE » :

```
mysql> UPDATE CoursPHP.personnes SET Prenom = 'PIERRE-ANDRE'
WHERE personnes.ID = 2;
Query OK, 1 row affected (0,01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Suppression

La syntaxe suivante supprime la personne ayant l'identifiant numéro 3 :

```
mysql> DELETE FROM personnes WHERE ID=3;
Query OK, 1 row affected (0,00 sec)
```

L'affichage montre que cette personne est supprimée.

```
mysql> SELECT * FROM personnes;
+--+---------+-----------+---+
|ID|Nom      |Prenom     |Age|
+--+---------+-----------+---+
| 1|DUPONT   |JEAN       | 28|
| 2|MARTIN   |PIERRE-ANDRE| 56|
| 4|DE-LA-RUE|JEAN-CHARLES| 45|
+--+---------+-----------+---+
3 rows in set (0,00 sec)
```

Les critères de sélection

Il est possible d'affiner les requêtes SQL comme SELECT, UPDATE ou DELETE avec des critères de sélection tels que : WHERE, ORDER BY, LIMIT. Pour montrer l'usage des ces critères, nous avons insérez de nouvelles données dans la table « personnes » dont voici le contenu :

```
mysql> SELECT * FROM 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|
| 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|
+--+--------------+---------------+---+
16 rows in set (0,00 sec)
```

Le filtrage avec *WHERE*

Ce critère permet de filtrer sur une partie des données. Par exemple, on peut afficher les prénoms et noms des personnes ayant 25 ans :

```
mysql> SELECT Prenom, Nom FROM personnes WHERE Age=25;
+-----------+---------+
|Prenom     |Nom      |
+-----------+---------+
|JEAN-MICHEL|LERY     |
|FREDERIC   |JACQUENOD|
|LAURENCE   |DE-LA-RUE|
|ALBERT     |MARTIN   |
+-----------+---------+
4 rows in set (0,00 sec)
```

IL est également possible d'utiliser l'opérateur LIKE avec la condition WHERE pour cherche d'après un modèle particulier. La syntaxe suivante affiche la liste des personnes ayant un prénom commençant par JEAN. Le caractère % indique un nombre quelconque de caractères.

```
mysql> SELECT Prenom, Nom FROM personnes WHERE Prenom LIKE
'JEAN%';
+---------------+--------------+
|Prenom         |Nom           |
+---------------+--------------+
|JEAN           |DUPONT        |
|JEAN-CHRISTOPHE|JACQUENOD     |
|JEAN-MICHEL    |LERY          |
|JEAN-CHRISTOPHE|DE-LA-RUE     |
|JEAN-CHRISTOPHE|DUMOULIN      |
|JEAN           |DE-LA-FONTAINE|
|JEAN           |DUPONT        |
+---------------+--------------+
7 rows in set (0,00 sec)
```

De même l'opérateur BETWEEN avec la condition WHERE autorise un filtrage sur une plage de valeurs. La syntaxe suivante travaille sur la table « clients ». Elle affiche les clients ayant entre 1 et 2 enfants :

```
mysql> SELECT ID,Nom,Prenom,Etat_Civil,Nb_Enfants FROM clients
WHERE Nb_Enfants BETWEEN 1 AND 2;
+--+------------+---------------+-----------+----------+
|ID|Nom         |Prenom         |Etat_Civil |Nb_Enfants|
+--+------------+---------------+-----------+----------+
| 1|DUPONT      |JEAN           |Marié      |         2|
| 2|JACQUENOD   |JEAN-CHRISTOPHE|Marié      |         1|
| 3|MURCIAN     |CAROLE         |Célibataire|         1|
| 4|LERY        |JEAN-MICHEL    |Marié      |         2|
|10|DUMOULIN    |JEAN-CHRISTOPHE|Marié      |         2|
```

```
|11|LABONNE-JAYAT|OLIVIER        |Célibataire|         1|
|14|DE-LA-RUE    |LAURENCE       |Marié      |         1|
|15|DUPONT       |JEAN           |Veuf       |         2|
|16|MARTIN       |ALBERT         |Célibataire|         1|
+--+-------------+---------------+-----------+----------+
9 rows in set (0,00 sec)
```

Cette autre syntaxe affiche dans l'ordre, le prénom, le nom et l'âge pour toutes les personnes dont l'âge est supérieur à 25 ans :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25;
+---------------+-------------+---+
|Prenom         |Nom          |Age|
+---------------+-------------+---+
|JEAN           |DUPONT       | 28|
|JEAN-CHRISTOPHE|JACQUENOD    | 54|
|CAROLE         |MURCIAN      | 44|
|JEAN-CHRISTOPHE|DE-LA-RUE    | 27|
|PIERRE-DAVID   |MARTIN       | 27|
|PIERRE         |MARTIN       | 56|
|JEAN-CHRISTOPHE|DUMOULIN     | 54|
|OLIVIER        |LABONNE-JAYAT | 54|
|JEAN           |DE-LA-FONTAINE|110|
|SAMUEL         |LEVY         | 56|
|JEAN           |DUPONT       | 54|
+---------------+-------------+---+
11 rows in set (0,00 sec)
```

Il est possible de combiner plusieurs conditions. Cette syntaxe affiche le prénom, le nom et l'âge des personnes dont l'âge est supérieur à 25 ans **ET** ayant comme prénom JEAN :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 AND
Prenom="JEAN";
+------+-------------+---+
|Prenom|Nom          |Age|
+------+-------------+---+
|JEAN  |DUPONT       | 28|
|JEAN  |DE-LA-FONTAINE|110|
|JEAN  |DUPONT       | 54|
+------+-------------+---+
3 rows in set (0,00 sec)
```

Cette syntaxe affiche le prénom, le nom et l'âge des personnes dont l'âge est supérieur à 25 ans **OU** ayant comme prénom JEAN :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN";
+---------------+-------------+---+
```

```
|Prenom          |Nom           |Age|
+---------------+--------------+---+
|JEAN            |DUPONT        | 28|
|JEAN-CHRISTOPHE|JACQUENOD     | 54|
|CAROLE          |MURCIAN       | 44|
|JEAN-CHRISTOPHE|DE-LA-RUE     | 27|
|PIERRE-DAVID    |MARTIN        | 27|
|PIERRE          |MARTIN        | 56|
|JEAN-CHRISTOPHE|DUMOULIN      | 54|
|OLIVIER         |LABONNE-JAYAT | 54|
|JEAN            |DE-LA-FONTAINE|110|
|SAMUEL          |LEVY          | 56|
|JEAN            |DUPONT        | 54|
+---------------+--------------+---+
11 rows in set (0,00 sec)
```

Le tri avec *ORDER BY*

La syntaxe ORDER BY ordonne les résultats de la requête. Si on désire présenter le résultat de la requête précédente par ordre croisant de l'âge, la syntaxe devient :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN" ORDER BY Age;
+---------------+--------------+---+
|Prenom          |Nom           |Age|
+---------------+--------------+---+
|JEAN-CHRISTOPHE|DE-LA-RUE     | 27|
|PIERRE-DAVID    |MARTIN        | 27|
|JEAN            |DUPONT        | 28|
|CAROLE          |MURCIAN       | 44|
|JEAN-CHRISTOPHE|JACQUENOD     | 54|
|JEAN-CHRISTOPHE|DUMOULIN      | 54|
|OLIVIER         |LABONNE-JAYAT | 54|
|JEAN            |DUPONT        | 54|
|PIERRE          |MARTIN        | 56|
|SAMUEL          |LEVY          | 56|
|JEAN            |DE-LA-FONTAINE|110|
+---------------+--------------+---+
11 rows in set (0,00 sec)
```

Pour un affichage par ordre décroissant il suffit d'ajouter DESC à la fin de la syntaxe :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN" ORDER BY Age DESC;
+---------------+--------------+---+
|Prenom          |Nom           |Age|
+---------------+--------------+---+
|JEAN            |DE-LA-FONTAINE|110|
```

```
|PIERRE          |MARTIN         | 56|
|SAMUEL          |LEVY           | 56|
|JEAN-CHRISTOPHE|JACQUENOD      | 54|
|JEAN-CHRISTOPHE|DUMOULIN       | 54|
|OLIVIER         |LABONNE-JAYAT  | 54|
|JEAN            |DUPONT         | 54|
|CAROLE          |MURCIAN        | 44|
|JEAN            |DUPONT         | 28|
|JEAN-CHRISTOPHE|DE-LA-RUE      | 27|
|PIERRE-DAVID    |MARTIN         | 27|
+---------------+--------------+---+
11 rows in set (0,00 sec)
```

Pour un tri sur le nom :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN" ORDER BY Nom;
+---------------+--------------+---+
|Prenom          |Nom           |Age|
+---------------+--------------+---+
|JEAN            |DE-LA-FONTAINE|110|
|JEAN-CHRISTOPHE|DE-LA-RUE      | 27|
|JEAN-CHRISTOPHE|DUMOULIN       | 54|
|JEAN            |DUPONT         | 28|
|JEAN            |DUPONT         | 54|
|JEAN-CHRISTOPHE|JACQUENOD      | 54|
|OLIVIER         |LABONNE-JAYAT  | 54|
|SAMUEL          |LEVY           | 56|
|PIERRE-DAVID    |MARTIN         | 27|
|PIERRE          |MARTIN         | 56|
|CAROLE          |MURCIAN        | 44|
+---------------+--------------+---+
11 rows in set (0,00 sec)
```

Dans ce dernier cas, le choix de la table de codage des caractères (UTF8) et la sensibilité à la casse impacte le tri « alphabétique ».

La limitation avec *LIMIT*

La syntaxe LIMIT ne sélectionne qu'une partie des résultats de la requête. La syntaxe générale est :

```
LIMIT début, nb
```

Où `début` est le numéro de l'entrée dans le résultat (0 pour la première entrée), et `nb` le nombre d'entrées à sélectionner. Voici quelques exemples de syntaxe :

```
LIMIT 0,5 : les 5 premières entrée ;
```

LIMIT 5,3 : de la 6ème entrée à la 8ème entrée (3 entrées à partir de la 6ème)

La syntaxe suivante affiche les 5 premières lignes de la requête précédente :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN" ORDER BY Nom LIMIT 0,5;
+---------------+--------------+---+
|Prenom         |Nom           |Age|
+---------------+--------------+---+
|JEAN           |DE-LA-FONTAINE|110|
|JEAN-CHRISTOPHE|DE-LA-RUE     | 27|
|JEAN-CHRISTOPHE|DUMOULIN      | 54|
|JEAN           |DUPONT        | 28|
|JEAN           |DUPONT        | 54|
+---------------+--------------+---+
5 rows in set (0,00 sec)
```

Voici les lignes 6 à 8 :

```
mysql> SELECT Prenom, Nom, Age FROM personnes WHERE Age>25 OR
Prenom="JEAN" ORDER BY Nom LIMIT 5,3;
+---------------+-------------+---+
|Prenom         |Nom          |Age|
+---------------+-------------+---+
|JEAN-CHRISTOPHE|JACQUENOD    | 54|
|OLIVIER        |LABONNE-JAYAT| 54|
|SAMUEL         |LEVY         | 56|
+---------------+-------------+---+
3 rows in set (0,00 sec)
```

Le filtrage avec *HAVING*

Cette condition se comporte comme WHERE. La différence est que HAVING permet de filtrer en utilisant des fonctions comme SUM(), COUNT(), AVG(), MIN() ou MAX(). Elle s'utilise généralement sur des données regroupées par GROUP BY. Des exemples de syntaxes sont présentés avec ces fonctions.

Le regroupement avec *GROUP BY*

Il est possible de regrouper les résultats selon un champ avec GROUP BY. Cette syntaxe est utilisée avec les fonctions SQL d'agrégations comme AVG(), SUM(), ... Des exemples de syntaxes sont présentés avec ces fonctions.

Les fonctions SQL

Le langage SQL effectue des traitements sur les données via des fonctions. Elles sont spécifiques aux bases de données et donc très rapides. Nous n'en présentons

que quelques-unes, une liste exhaustive des fonctions SQL est disponible à l'URL http://sql.sh/fonctions.

Remarque

Pour des questions de performance, il faut privilégier une fonction SQL, quand elle existe, à une fonction ou un traitement équivalent en PHP.

Il y a plusieurs catégories de fonctions :

- *Les fonction d'agrégat* **:** Elles effectuent un traitement sur la totalité de la table. C'est par exemple la somme ou la moyenne d'un champ numérique.
- *Les fonctions scalaires* **:** Elles travaillent sur chaque entrée de la table. On y trouve :
  - ♦ Les fonctions sur les chaînes de caractères, comme la conversion en majuscules ou minuscules d'un champ texte ;
  - ♦ Les fonctions mathématiques telles que l'arrondi d'un champ numérique ;
  - ♦ Les fonctions de date et d'heure ;
  - ♦ Les fonctions de chiffrement ;
  - ♦ Diverses fonctions comme le transtypage CAST() ou la conversion CONVERT().

Pour cette section, nous utilisons la table « clients » (d'une banque) dont la structure est présentée sur la figure 10-1.1 et ses enregistrements sur la figure 10-1.2.

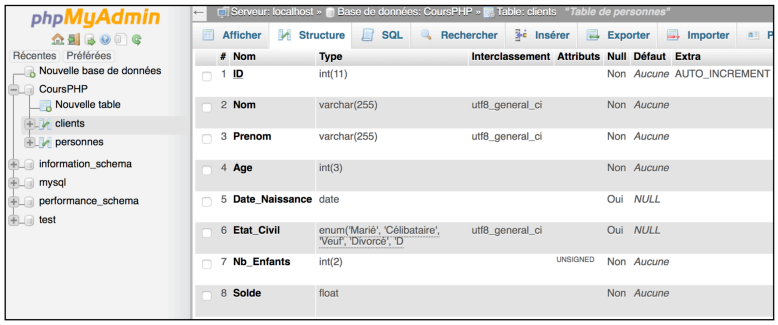


Figure 10-1.1

Structure de la table clients.

ID	Nom	Prenom	Age	Date_Naissance	Etat_Civil	Nb_Enfants	Solde
1	DUPONT	JEAN	27	1987-12-28	Marié	2	1200.5
2	JACQUENOD	JEAN-CHRISTOPHE	54	1961-02-10	Marié	1	-308.87
3	MURCIAN	CAROLE	44	1970-10-20	Célibataire	1	3548.98
4	LERY	JEAN-MICHEL	25	1989-05-07	Marié	2	-18.98
5	DE-LA-RUE	JEAN-CHRISTOPHE	23	1991-06-18	Divorcé	0	-27.44
6	MARTIN	PIERRE-DAVID	23	1991-08-22	Célibataire	0	206.21
7	MARTIN	PIERRE	56	1959-01-18	Veuf	3	1234.56
8	JACQUENOD	FREDERIC	25	1989-11-27	Marié	0	432.98
9	JACQUENOD	LAURENCE	24	1990-11-01	Marié	0	-203.18
10	DUMOULIN	JEAN-CHRISTOPHE	54	1960-08-22	Marié	2	-2186.86
11	LABONNE-JAYAT	OLIVIER	54	1960-09-23	Célibataire	1	-65.98
12	DE-LA-FONTAINE	JEAN	110	1905-01-22	Décédé	0	1825.54
13	LEVY	SAMUEL	56	1959-03-27	Divorcé	3	231.87
14	DE-LA-RUE	LAURENCE	25	1989-12-13	Marié	1	2135.98
15	DUPONT	JEAN	54	1960-10-15	Veuf	2	12314.9
16	MARTIN	ALBERT	25	1989-08-15	Célibataire	1	213.49

Figure 10-1.2
Enregistrements de la table clients.

Remarque
Il est illogique de conserver un champ « Age » dès lors qu'il y a un champ « Date_Naissance ». Cela ne peut que produire des incohérences de données, en plus de la redondance d'information. Ce champ est maintenu uniquement pour servir de support aux exemples suivants.

Les fonctions d'agrégat

AVG

La fonction AVG() calcule la **moyenne** sur un ensemble d'enregistrements. Elle fournit un résultat sous la forme d'un « champ virtuel » qui n'existe que durant la requête, et qu'il est préférable de nommer. Pour cet exemple nous utiliserons le nom « **solde_moyen** » :

```
mysql> SELECT AVG(Solde) AS solde_moyen FROM clients;
+------------------+
|solde_moyen       |
+------------------+
|1283.3543809652328|
+------------------+
1 row in set (0,00 sec)
```

La fonction ROUND(), présente le résultat à la deuxième décimale :

```
mysql> SELECT ROUND(AVG(Solde),2) AS solde_moyen FROM clients;
+-----------+
|solde_moyen|
+-----------+
```

```
|    1283.35|
+-----------+
1 row in set (0,00 sec)
```

Avec la syntaxe GROUP BY, il est possible de regrouper le calcul selon un des champs. Voici la syntaxe précédente présentée selon l'état civil des clients. L'état civil est également affiché pour connaître le champ utilisé pour le regroupement :

```
mysql> SELECT Etat_Civil,ROUND(AVG(Solde),2) AS solde_moyen
FROM clients GROUP BY Etat_Civil;
+------------+-----------+
|Etat_Civil  |solde_moyen|
+------------+-----------+
|Marié       |     150.22|
|Célibataire |     975.67|
|Veuf        |    6774.72|
|Divorcé     |     102.21|
|Décédé      |    1825.54|
+------------+-----------+
5 rows in set (0,00 sec)
```

Avec la condition HAVING, il est possible de filtrer le résultat. Voici la syntaxe précédente où seuls les soldes moyens supérieurs à 1000 sont affichés :

```
mysql> SELECT Etat_Civil,ROUND(AVG(Solde),2) AS solde_moyen
FROM clients GROUP BY Etat_Civil HAVING solde_moyen > 1000;
+----------+-----------+
|Etat_Civil|solde_moyen|
+----------+-----------+
|Veuf      |    6774.72|
|Décédé    |    1825.54|
+----------+-----------+
2 rows in set (0,00 sec)
```

COUNT

La fonction COUNT() calcule le **nombre** d'enregistrement dans une table. Voici le nombre total de clients :

```
mysql> SELECT COUNT(*) AS nbclients FROM clients;
+---------+
|nbclients|
+---------+
|       16|
+---------+
1 row in set (0,00 sec)
```

Voici le nombre total de clients mariés :

```
mysql> SELECT COUNT(*) AS nbmarié FROM clients WHERE
Etat_Civil='Marié';
```

```
+--------+
|nbmarié |
+--------+
|       7|
+--------+
1 row in set (0,00 sec)
```

La syntaxe GROUP BY, permet d'afficher le nombre de clients selon leur Age :

```
mysql> SELECT Age,COUNT(*) AS nbclients FROM clients GROUP BY
Age;
+---+---------+
|Age|nbclients|
+---+---------+
| 23|        2|
| 24|        1|
| 25|        4|
| 27|        1|
| 44|        1|
| 54|        4|
| 56|        2|
|110|        1|
+---+---------+
8 rows in set (0,00 sec)
```

Avec la condition HAVING, on peut filtrer ce résultat pour n'afficher que le nombre de clients supérieur à 1 :

```
mysql> SELECT Age,COUNT(*) AS nbclients FROM clients GROUP BY
Age HAVING COUNT(*)>1;
+---+---------+
|Age|nbclients|
+---+---------+
| 23|        2|
| 25|        4|
| 54|        4|
| 56|        2|
+---+---------+
4 rows in set (0,00 sec)
```

Cette syntaxe est identique à

```
mysql> SELECT Age,COUNT(*) AS nbclients FROM clients GROUP BY
Age HAVING nbclients>1;
```

MAX

La fonction MAX() calcule la valeur **maximale** dans une table. Voici le client ayant le plus grand solde :

```
mysql> SELECT Nom, ROUND(MAX(Solde),2) AS solde_max FROM
clients;
+------+---------+
|Nom   |solde_max|
+------+---------+
|DUPONT| 12314.87|
+------+---------+
1 row in set (0,00 sec)
```

MIN

La fonction MIN() calcule la valeur **minimale** dans une table. Voici le client ayant le plus petit solde :

```
mysql> SELECT Nom, ROUND(MIN(Solde),2) AS solde_min FROM
clients;
+------+---------+
|Nom   |solde_min|
+------+---------+
|DUPONT| -2186.86|
+------+---------+
1 row in set (0,00 sec)
```

SUM

La fonction SUM() calcule la **somme** d'un champ numérique. La syntaxe suivante affiche le solde total de tous les clients :

```
mysql> SELECT ROUND(SUM(Solde),2) AS solde_total FROM clients;
+-----------+
|solde_total|
+-----------+
|   20533.67|
+-----------+
1 row in set (0,00 sec)
```

Voici le solde total des clients décédés :

```
mysql> SELECT ROUND(SUM(Solde),2) AS solde_total FROM clients
WHERE Etat_Civil='Décédé';
+-----------+
|solde_total|
+-----------+
|    1825.54|
+-----------+
1 row in set (0,00 sec)
```

Voici le solde total des clients selon leur âge :

```
mysql> SELECT Age,ROUND(SUM(Solde),2) AS solde_total FROM
clients GROUP BY Age;
```

```
+---+-----------+
|Age|solde_total|
+---+-----------+
| 23|     178.77|
| 24|    -203.18|
| 25|    2763.47|
| 27|    1200.50|
| 44|    3548.98|
| 54|    9753.16|
| 56|    1466.43|
|110|    1825.54|
+---+-----------+
8 rows in set (0,00 sec)
```

Voici le solde total des clients, dépassant 1000, en fonction de leur âge :

```
mysql> SELECT Age,ROUND(SUM(Solde),2) AS solde_total FROM
clients GROUP BY Age HAVING solde_total>1000;
+---+-----------+
|Age|solde_total|
+---+-----------+
| 25|    2763.47|
| 27|    1200.50|
| 44|    3548.98|
| 54|    9753.16|
| 56|    1466.43|
|110|    1825.54|
+---+-----------+
6 rows in set (0,00 sec)
```

Quelques fonctions sur les chaînes de caractères

Les fonctions sur les chaînes de caractères sont nombreuses. Nous n'en présentons que quelques unes. Une présentation complète est disponible à l'URL http://sql.sh/fonctions/chaines-de-caracteres.

CONCAT

C'est la concaténation de chaînes de caractères. L'exemple suivant concatène le « prénom », un espace, et le « nom » dans un seul champ nommé « prenom_nom » :

```
mysql> SELECT ID, CONCAT(Prenom,' ',Nom) AS prenom_nom,
Date_Naissance FROM clients;
+--+-------------------------+--------------+
|ID|prenom_nom               |Date_Naissance|
+--+-------------------------+--------------+
| 1|JEAN DUPONT              |1987-12-28    |
| 2|JEAN-CHRISTOPHE JACQUENOD|1961-02-10    |
| 3|CAROLE MURCIAN           |1970-10-20    |
```

```
| 4|JEAN-MICHEL LERY          |1989-05-07    |
| 5|JEAN-CHRISTOPHE DE-LA-RUE|1991-06-18    |
| 6|PIERRE-DAVID MARTIN       |1991-08-22    |
| 7|PIERRE MARTIN             |1959-01-18    |
| 8|FREDERIC JACQUENOD        |1989-11-27    |
| 9|LAURENCE JACQUENOD        |1990-11-01    |
|10|JEAN-CHRISTOPHE DUMOULIN |1960-08-22    |
|11|OLIVIER LABONNE-JAYAT     |1960-09-23    |
|12|JEAN DE-LA-FONTAINE       |1905-01-22    |
|13|SAMUEL LEVY               |1959-03-27    |
|14|LAURENCE DE-LA-RUE        |1989-12-13    |
|15|JEAN DUPONT               |1960-10-15    |
|16|ALBERT MARTIN             |1989-08-15    |
+--+-------------------------+--------------+
16 rows in set (0,00 sec)
```

LENGTH

C'est la longueur d'une chaîne de caractères. L'exemple suivant affiche la taille du champ nom de tous les clients :

```
mysql> SELECT ID,Nom,LENGTH(Nom) AS Taille_Nom FROM clients;
+--+--------------+----------+
|ID|Nom           |Taille_Nom|
+--+--------------+----------+
| 1|DUPONT        |         6|
| 2|JACQUENOD     |         9|
| 3|MURCIAN       |         7|
| 4|LERY          |         4|
| 5|DE-LA-RUE     |         9|
| 6|MARTIN        |         6|
| 7|MARTIN        |         6|
| 8|JACQUENOD     |         9|
| 9|JACQUENOD     |         9|
|10|DUMOULIN      |         8|
|11|LABONNE-JAYAT |        13|
|12|DE-LA-FONTAINE|        14|
|13|LEVY          |         4|
|14|DE-LA-RUE     |         9|
|15|DUPONT        |         6|
|16|MARTIN        |         6|
+--+--------------+----------+
16 rows in set (0,00 sec)
```

Cet autre exemple affiche la taille du plus grand nom de la table clients :

```
mysql> SELECT MAX(LENGTH(Nom)) AS Taille_Max_Nom FROM clients;
+--------------+
|Taille_Max_Nom|
+--------------+
```

```
|            14|
+--------------+
1 row in set (0,01 sec)
```

REPLACE

Cette fonction remplace une chaîne par une autre. La syntaxe suivante montre le résultat du remplacement du texte 'DUPONT' par 'DURAND' dans la colonne « Nom ». Comme il s'agit d'un SELECT aucun changement n'est effectué.

```
mysql> SELECT ID,Nom,REPLACE(Nom,'DUPONT','DURAND') as
Nouveau_Nom FROM clients;
+--+--------------+--------------+
|ID|Nom           |Nouveau_Nom   |
+--+--------------+--------------+
| 1|DUPONT        |DURAND        |
| 2|JACQUENOD     |JACQUENOD     |
| 3|MURCIAN       |MURCIAN       |
| 4|LERY          |LERY          |
| 5|DE-LA-RUE     |DE-LA-RUE     |
| 6|MARTIN        |MARTIN        |
| 7|MARTIN        |MARTIN        |
| 8|JACQUENOD     |JACQUENOD     |
| 9|JACQUENOD     |JACQUENOD     |
|10|DUMOULIN      |DUMOULIN      |
|11|LABONNE-JAYAT |LABONNE-JAYAT |
|12|DE-LA-FONTAINE|DE-LA-FONTAINE|
|13|LEVY          |LEVY          |
|14|DE-LA-RUE     |DE-LA-RUE     |
|15|DUPONT        |DURAND        |
|16|MARTIN        |MARTIN        |
+--+--------------+--------------+
16 rows in set (0,00 sec)
```

L'exemple montre l'utilisation de REPLACE avec une requête UPDATE, pour mettre à jour une partie du prénom.

```
mysql> UPDATE clients SET
Prenom=REPLACE(Prenom,'PIERRE','PAUL') WHERE ID=6;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

La requête suivante montre le résultat de cette mise à jour du prénom de l'utilisateur ayant l'ID 6 qui a été changé de 'PIERRE-DAVID' en 'PAUL-DAVID'.

```
mysql> SELECT ID,Nom,Prenom FROM clients;
+--+--------------+---------------+
|ID|Nom           |Prenom         |
+--+--------------+---------------+
| 1|DUPONT        |JEAN           |
```

```
| 2|JACQUENOD     |JEAN-CHRISTOPHE|
| 3|MURCIAN       |CAROLE         |
| 4|LERY          |JEAN-MICHEL    |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|
| 6|MARTIN        |PAUL-DAVID     |
| 7|MARTIN        |PIERRE         |
| 8|JACQUENOD     |FREDERIC       |
| 9|JACQUENOD     |LAURENCE       |
|10|DUMOULIN      |JEAN-CHRISTOPHE|
|11|LABONNE-JAYAT |OLIVIER        |
|12|DE-LA-FONTAINE|JEAN           |
|13|LEVY          |SAMUEL         |
|14|DE-LA-RUE     |LAURENCE       |
|15|DUPONT        |JEAN           |
|16|MARTIN        |ALBERT         |
+--+------------------------------+
16 rows in set (0,00 sec)
```

SUBSTRING

La fonction SUBSTRING retourne une partie de la chaîne indiqué. Elle possède plusieurs syntaxes :

- SUBSTRING(chaine,debut) : Retourne la chaîne à partir de début ;
- SUBSTRING(chaine FROM debut) : Retourne la chaîne à partir de début ;
- SUBSTRING(chaine,debut,longueur) : Retourne la chaîne à partir de début sur longueur caractères ;
- SUBSTRING(chaine FROM debut FOR longueur) : Retourne la chaîne à partir de début sur longueur caractères ;

L'exemple suivant affiche les quatre premiers caractères du prénom dans une nouvelle colonne.

```
mysql> SELECT ID,Nom,Prenom,SUBSTRING(Prenom,1,4) as
Prem_4_Caract FROM clients;
+--+--------------+---------------+-------------+
|ID|Nom           |Prenom         |Prem_4_Caract|
+--+--------------+---------------+-------------+
| 1|DUPONT        |JEAN           |JEAN         |
| 2|JACQUENOD     |JEAN-CHRISTOPHE|JEAN         |
| 3|MURCIAN       |CAROLE         |CARO         |
| 4|LERY          |JEAN-MICHEL    |JEAN         |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|JEAN         |
| 6|MARTIN        |PAUL-DAVID     |PAUL         |
| 7|MARTIN        |PIERRE         |PIER         |
| 8|JACQUENOD     |FREDERIC       |FRED         |
| 9|JACQUENOD     |LAURENCE       |LAUR         |
|10|DUMOULIN      |JEAN-CHRISTOPHE|JEAN         |
|11|LABONNE-JAYAT |OLIVIER        |OLIV         |
|12|DE-LA-FONTAINE|JEAN           |JEAN         |
```

```
|13|LEVY          |SAMUEL         |SAMU         |
|14|DE-LA-RUE     |LAURENCE       |LAUR         |
|15|DUPONT        |JEAN           |JEAN         |
|16|MARTIN        |ALBERT         |ALBE         |
+--+--------------+---------------+-------------+
16 rows in set (0,00 sec)
```

LEFT

La fonction LEFT retourne les N caractères de gauche (premiers). L'affichage précèdent aurait pu être obtenu par la syntaxe :

```
mysql> SELECT ID,Nom,Prenom,LEFT(Prenom,4) as Prem_4_Caract
FROM clients;
```

RIGHT

La fonction RIGHT retourne les N caractères de droite (derniers). L'exemple suivant affiche les quatre derniers caractères du prénom dans une nouvelle colonne.

```
mysql> SELECT ID,Nom,Prenom,RIGHT(Prenom,4) as Dern_4_Caract
FROM clients;
+--+--------------+---------------+-------------+
|ID|Nom           |Prenom         |Dern_4_Caract|
+--+--------------+---------------+-------------+
| 1|DUPONT        |JEAN           |JEAN         |
| 2|JACQUENOD     |JEAN-CHRISTOPHE|OPHE         |
| 3|MURCIAN       |CAROLE         |ROLE         |
| 4|LERY          |JEAN-MICHEL    |CHEL         |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|OPHE         |
| 6|MARTIN        |PAUL-DAVID     |AVID         |
| 7|MARTIN        |PIERRE         |ERRE         |
| 8|JACQUENOD     |FREDERIC       |ERIC         |
| 9|JACQUENOD     |LAURENCE       |ENCE         |
|10|DUMOULIN      |JEAN-CHRISTOPHE|OPHE         |
|11|LABONNE-JAYAT |OLIVIER        |VIER         |
|12|DE-LA-FONTAINE|JEAN           |JEAN         |
|13|LEVY          |SAMUEL         |MUEL         |
|14|DE-LA-RUE     |LAURENCE       |ENCE         |
|15|DUPONT        |JEAN           |JEAN         |
|16|MARTIN        |ALBERT         |BERT         |
+--+--------------+---------------+-------------+
16 rows in set (0,00 sec)
```

REVERSE

La fonction REVERSE renverse l'ordre des caractères d'une chaîne. Voici un exemple d'inversion des lettres du prénom :

```
mysql> SELECT ID,Nom,Prenom,REVERSE(Prenom) as Prenom_retourné
FROM clients;
```

```
+--+--------------+---------------+----------------+
|ID|Nom           |Prenom         |Prenom_retourné |
+--+--------------+---------------+----------------+
| 1|DUPONT        |JEAN           |NAEJ            |
| 2|JACQUENOD     |JEAN-CHRISTOPHE|EHPOTSIRHC-NAEJ |
| 3|MURCIAN       |CAROLE         |ELORAC          |
| 4|LERY          |JEAN-MICHEL    |LEHCIM-NAEJ     |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|EHPOTSIRHC-NAEJ |
| 6|MARTIN        |PAUL-DAVID     |DIVAD-LUAP      |
| 7|MARTIN        |PIERRE         |ERREIP          |
| 8|JACQUENOD     |FREDERIC       |CIREDERF        |
| 9|JACQUENOD     |LAURENCE       |ECNERUAL        |
|10|DUMOULIN      |JEAN-CHRISTOPHE|EHPOTSIRHC-NAEJ |
|11|LABONNE-JAYAT |OLIVIER        |REIVILO         |
|12|DE-LA-FONTAINE|JEAN           |NAEJ            |
|13|LEVY          |SAMUEL         |LEUMAS          |
|14|DE-LA-RUE     |LAURENCE       |ECNERUAL        |
|15|DUPONT        |JEAN           |NAEJ            |
|16|MARTIN        |ALBERT         |TREBLA          |
+--+--------------+---------------+----------------+
16 rows in set (0,00 sec)
```

TRIM, LTRIM, RTRIM

La fonction TRIM supprime les caractères invisibles (espaces, tabulations, retour à la ligne) au début et en fin de chaîne. En voici un exemple :

```
mysql> SELECT ID,Nom,Prenom,TRIM(Prenom) as Prenom_nettoyé
FROM clients;
```

La fonction LTRIM applique ce traitement à gauche (début) de la chaîne. La fonction RTRIM applique ce traitement à droite (fin) de la chaîne.

LPAD, RPAD

La fonction LPAD complète une chaîne de caractère jusqu'à atteindre la taille demandée en ajoutant des caractères en début de chaîne (à gauche). En voici un exemple :

```
mysql> SELECT ID,Nom,Prenom,LPAD(Prenom,14,'_') as
Prenom_complété FROM clients;
+--+--------------+---------------+----------------+
|ID|Nom           |Prenom         |Prenom_complété |
+--+--------------+---------------+----------------+
| 1|DUPONT        |JEAN           |__________JEAN  |
| 2|JACQUENOD     |JEAN-CHRISTOPHE|JEAN-CHRISTOPH  |
| 3|MURCIAN       |CAROLE         |________CAROLE  |
| 4|LERY          |JEAN-MICHEL    |___JEAN-MICHEL  |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|JEAN-CHRISTOPH  |
| 6|MARTIN        |PAUL-DAVID     |____PAUL-DAVID  |
| 7|MARTIN        |PIERRE         |________PIERRE  |
```

```
| 8|JACQUENOD     |FREDERIC       |______FREDERIC   |
| 9|JACQUENOD     |LAURENCE       |______LAURENCE   |
|10|DUMOULIN      |JEAN-CHRISTOPHE|JEAN-CHRISTOPH   |
|11|LABONNE-JAYAT |OLIVIER        |_______OLIVIER   |
|12|DE-LA-FONTAINE|JEAN           |__________JEAN   |
|13|LEVY          |SAMUEL         |________SAMUEL   |
|14|DE-LA-RUE     |LAURENCE       |______LAURENCE   |
|15|DUPONT        |JEAN           |__________JEAN   |
|16|MARTIN        |ALBERT         |________ALBERT   |
+--+--------------+---------------+-----------------+
16 rows in set (0,00 sec)
```

La fonction RPAD effectue le même traitement en ajoutant le caractère de remplissage à droite.

LOWER, LCASE

Cette fonction convertit une chaîne en minuscules. LCASE est un alias de LOWER. En voici un exemple :

```
mysql> SELECT ID,LOWER(Nom),Prenom FROM clients;
+--+--------------+---------------+
|ID|LOWER(Nom)    |Prenom         |
+--+--------------+---------------+
| 1|dupont        |JEAN           |
| 2|jacquenod     |JEAN-CHRISTOPHE|
| 3|murcian       |CAROLE         |
| 4|lery          |JEAN-MICHEL    |
| 5|de-la-rue     |JEAN-CHRISTOPHE|
| 6|martin        |PAUL-DAVID     |
| 7|martin        |PIERRE         |
| 8|jacquenod     |FREDERIC       |
| 9|jacquenod     |LAURENCE       |
|10|dumoulin      |JEAN-CHRISTOPHE|
|11|labonne-jayat |OLIVIER        |
|12|de-la-fontaine|JEAN           |
|13|levy          |SAMUEL         |
|14|de-la-rue     |LAURENCE       |
|15|dupont        |JEAN           |
|16|martin        |ALBERT         |
+--+--------------+---------------+
16 rows in set (0,00 sec)
```

UPPER, UCASE

Cette fonction convertit une chaîne en majuscules. UCASE est un alias de UPPER. En voici un exemple :

```
mysql> SELECT ID,Nom,Prenom,UPPER(Etat_Civil) FROM clients;
+--+--------------+---------------+-----------------+
|ID|Nom           |Prenom         |UPPER(Etat_Civil)|
```

```
+--+--------------+---------------+-----------------+
| 1|DUPONT        |JEAN           |MARIÉ            |
| 2|JACQUENOD     |JEAN-CHRISTOPHE|MARIÉ            |
| 3|MURCIAN       |CAROLE         |CÉLIBATAIRE      |
| 4|LERY          |JEAN-MICHEL    |MARIÉ            |
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|DIVORCÉ          |
| 6|MARTIN        |PAUL-DAVID     |CÉLIBATAIRE      |
| 7|MARTIN        |PIERRE         |VEUF             |
| 8|JACQUENOD     |FREDERIC       |MARIÉ            |
| 9|JACQUENOD     |LAURENCE       |MARIÉ            |
|10|DUMOULIN      |JEAN-CHRISTOPHE|MARIÉ            |
|11|LABONNE-JAYAT |OLIVIER        |CÉLIBATAIRE      |
|12|DE-LA-FONTAINE|JEAN           |DÉCÉDÉ           |
|13|LEVY          |SAMUEL         |DIVORCÉ          |
|14|DE-LA-RUE     |LAURENCE       |MARIÉ            |
|15|DUPONT        |JEAN           |VEUF             |
|16|MARTIN        |ALBERT         |CÉLIBATAIRE      |
+--+--------------+---------------+-----------------+
16 rows in set (0,00 sec)
```

LOCATE, INSTR

La fonction LOCATE indique la position d'une sous-chaîne dans une chaîne. Cet exemple affiche la position du caractère 'C' dans le prénom :

```
mysql> SELECT ID,Nom,Prenom,LOCATE('C',Prenom) FROM clients;
+--+--------------+---------------+------------------+
|ID|Nom           |Prenom         |LOCATE('C',Prenom)|
+--+--------------+---------------+------------------+
| 1|DUPONT        |JEAN           |                 0|
| 2|JACQUENOD     |JEAN-CHRISTOPHE|                 6|
| 3|MURCIAN       |CAROLE         |                 1|
| 4|LERY          |JEAN-MICHEL    |                 8|
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|                 6|
| 6|MARTIN        |PAUL-DAVID     |                 0|
| 7|MARTIN        |PIERRE         |                 0|
| 8|JACQUENOD     |FREDERIC       |                 8|
| 9|JACQUENOD     |LAURENCE       |                 7|
|10|DUMOULIN      |JEAN-CHRISTOPHE|                 6|
|11|LABONNE-JAYAT |OLIVIER        |                 0|
|12|DE-LA-FONTAINE|JEAN           |                 0|
|13|LEVY          |SAMUEL         |                 0|
|14|DE-LA-RUE     |LAURENCE       |                 7|
|15|DUPONT        |JEAN           |                 0|
|16|MARTIN        |ALBERT         |                 0|
+--+--------------+---------------+------------------+
16 rows in set (0,00 sec)
```

La fonction INSTR est identique. Elle retourne la même information, mais les arguments sont inversés. Le résultat précédent peut être obtenu avec :

```
mysql> SELECT ID,Nom,Prenom,INSTR(Prenom,'C') FROM clients;
```

Les fonctions mathématiques

Il existe beaucoup de fonctions mathématiques comme CONV(), ABS(), POWER(), SQRT(), TRUNCATE(), ROUND() …, nous n'en présentons que deux.

TRUNCATE

Cette fonction tronque un nombre réel à la décimale indiquée. En voici un exemple :

```
mysql> SELECT ID,Nom,Prenom,TRUNCATE(Solde,0) AS Solde_Entier
FROM clients;
---+--------------+---------------+------------+
|ID|Nom           |Prenom         |Solde_Entier|
+--+--------------+---------------+------------+
| 1|DUPONT        |JEAN           |        1200|
| 2|JACQUENOD     |JEAN-CHRISTOPHE|        -308|
| 3|MURCIAN       |CAROLE         |        3548|
| 4|LERY          |JEAN-MICHEL    |         -18|
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|         -27|
| 6|MARTIN        |PAUL-DAVID     |         206|
| 7|MARTIN        |PIERRE         |        1234|
| 8|JACQUENOD     |FREDERIC       |         432|
| 9|JACQUENOD     |LAURENCE       |        -203|
|10|DUMOULIN      |JEAN-CHRISTOPHE|       -2186|
|11|LABONNE-JAYAT |OLIVIER        |         -65|
|12|DE-LA-FONTAINE|JEAN           |        1825|
|13|LEVY          |SAMUEL         |         231|
|14|DE-LA-RUE     |LAURENCE       |        2135|
|15|DUPONT        |JEAN           |       12314|
|16|MARTIN        |ALBERT         |         213|
+--+--------------+---------------+------------+
16 rows in set (0,00 sec)
```

ROUND

Cette fonction arrondit un nombre réel à la décimale indiquée. En voici un exemple :

```
mysql> SELECT ID,Nom,Prenom,ROUND(Solde,0) AS Solde_Entier
FROM clients;
+--+--------------+---------------+------------+
|ID|Nom           |Prenom         |Solde_Entier|
+--+--------------+---------------+------------+
| 1|DUPONT        |JEAN           |        1200|
| 2|JACQUENOD     |JEAN-CHRISTOPHE|        -309|
| 3|MURCIAN       |CAROLE         |        3549|
```

```
| 4|LERY           |JEAN-MICHEL    |         -19|
| 5|DE-LA-RUE      |JEAN-CHRISTOPHE|         -27|
| 6|MARTIN         |PAUL-DAVID     |         206|
| 7|MARTIN         |PIERRE         |        1235|
| 8|JACQUENOD      |FREDERIC       |         433|
| 9|JACQUENOD      |LAURENCE       |        -203|
|10|DUMOULIN       |JEAN-CHRISTOPHE|       -2187|
|11|LABONNE-JAYAT  |OLIVIER        |         -66|
|12|DE-LA-FONTAINE|JEAN           |        1826|
|13|LEVY           |SAMUEL         |         232|
|14|DE-LA-RUE      |LAURENCE       |        2136|
|15|DUPONT         |JEAN           |       12315|
|16|MARTIN         |ALBERT         |         213|
+--+--------------+---------------+------------+
16 rows in set (0,00 sec)
```

Les dates en SQL

Les types de dates et d'heures

Dans la structure de la table clients présentée précédemment, nous avons utilisé un champ « Date_Naissance » de type DATE. Il existe en SQL plusieurs types concernant les dates et heures. En voici une synthèse :

- DATE : La date est stockée au format AAAA-MM-JJ (Année-Jour-Mois) ;
- TIME : L'heure est stockée au format HH:MM:SS (Heures:Minutes:Secondes) ;
- DATETIME : La date et l'heure sont stockées au format AAAA-MM-JJ HH:MM:SS ;
- DATETIME : La date et l'heure sont stockées au format AAAAMMJJHHMMSS ;
- YEAR : L'année est stockée au format AAAA ;

Sélection des enregistrements selon une date

La requête suivante affiche tous les clients dont la date de naissance est postérieure au 1er janvier 1970 ;

```
mysql> SELECT ID,Nom,Prenom,Date_Naissance FROM clients WHERE
Date_Naissance >= '1970-01-01';
+--+---------+--------------+--------------+
|ID|Nom      |Prenom        |Date_Naissance|
+--+---------+--------------+--------------+
| 1|DUPONT   |JEAN          |1987-12-28    |
| 3|MURCIAN  |CAROLE        |1970-10-20    |
| 4|LERY     |JEAN-MICHEL   |1989-05-07    |
| 5|DE-LA-RUE|JEAN-CHRISTOPHE|1991-06-18    |
| 6|MARTIN   |PAUL-DAVID    |1991-08-22    |
| 8|JACQUENOD|FREDERIC      |1989-11-27    |
```

```
| 9|JACQUENOD|LAURENCE       |1990-11-01    |
|14|DE-LA-RUE|LAURENCE       |1989-12-13    |
|16|MARTIN   |ALBERT         |1989-08-15    |
+--+---------+---------------+--------------+
9 rows in set (0,00 sec)
```

Cette autre requête affiche les clients née entre le 1er janvier 1970 et le 31 décembre 1989 ;

```
mysql> SELECT ID,Nom,Prenom,Date_Naissance FROM clients WHERE
Date_Naissance BETWEEN '1970-01-01' AND '1989-12-31';
+--+---------+-----------+--------------+
|ID|Nom      |Prenom     |Date_Naissance|
+--+---------+-----------+--------------+
| 1|DUPONT   |JEAN       |1987-12-28    |
| 3|MURCIAN  |CAROLE     |1970-10-20    |
| 4|LERY     |JEAN-MICHEL|1989-05-07    |
| 8|JACQUENOD|FREDERIC   |1989-11-27    |
|14|DE-LA-RUE|LAURENCE   |1989-12-13    |
|16|MARTIN   |ALBERT     |1989-08-15    |
+--+---------+-----------+--------------+
6 rows in set (0,00 sec)
```

Les fonctions de dates et d'heures

Nous présentons dans cette section quelques fonctions de gestion des dates et heures en SQL. Une liste exhaustive est présentée à l'URL http://sql.sh/fonctions/date-heure

NOW, CURDATE, CURTIME

La fonction NOW() retourne la date actuelle au format AAAA-MM-JJ HH:MM:SS. La fonction CURDATE() retourne la date actuelle au format AAAA-MM-JJ. La fonction CURTIME() retourne la date actuelle au format HH:MM:SS. Un exemple d'utilisation de la fonction NOW() est présenté avec la fonction DATEDIFF().

DAY, MONTH, YEAR

Les fonctions DAY(), MONTH() et YEAR() retournent respectivement le jour, le mois et l'année d'une date. La requête suivante affiche l'année de naissance des clients :

```
mysql> SELECT ID,Nom,Prenom,YEAR(Date_Naissance) AS
Année_Naissance FROM clients;
+--+-------------+---------------+----------------+
|ID|Nom          |Prenom         |Année_Naissance |
+--+-------------+---------------+----------------+
| 1|DUPONT       |JEAN           |            1987|
| 2|JACQUENOD    |JEAN-CHRISTOPHE|            1961|
| 3|MURCIAN      |CAROLE         |            1970|
```

```
| 4|LERY           |JEAN-MICHEL    |            1989|
| 5|DE-LA-RUE      |JEAN-CHRISTOPHE|            1991|
| 6|MARTIN         |PAUL-DAVID     |            1991|
| 7|MARTIN         |PIERRE         |            1959|
| 8|JACQUENOD      |FREDERIC       |            1989|
| 9|JACQUENOD      |LAURENCE       |            1990|
|10|DUMOULIN       |JEAN-CHRISTOPHE|            1960|
|11|LABONNE-JAYAT  |OLIVIER        |            1960|
|12|DE-LA-FONTAINE|JEAN           |            1905|
|13|LEVY           |SAMUEL         |            1959|
|14|DE-LA-RUE      |LAURENCE       |            1989|
|15|DUPONT         |JEAN           |            1960|
|16|MARTIN         |ALBERT         |            1989|
+--+--------------+---------------+----------------+
16 rows in set (0,01 sec)
```

DATE_FORMAT

Cette fonction présente la date et l'heure selon le format indiqué. En voici un exemple :

```
mysql> SELECT
ID,Nom,Prenom,DATE_FORMAT(Date_Naissance,'%d/%m/%Y') AS
Naissance FROM clients;
+--+--------------+---------------+----------+
|ID|Nom           |Prenom         |Naissance |
+--+--------------+---------------+----------+
| 1|DUPONT        |JEAN           |28/12/1987|
| 2|JACQUENOD     |JEAN-CHRISTOPHE|10/02/1961|
| 3|MURCIAN       |CAROLE         |20/10/1970|
| 4|LERY          |JEAN-MICHEL    |07/05/1989|
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE|18/06/1991|
| 6|MARTIN        |PIERRE-DAVID   |22/08/1991|
| 7|MARTIN        |PIERRE         |18/01/1959|
| 8|JACQUENOD     |FREDERIC       |27/11/1989|
| 9|JACQUENOD     |LAURENCE       |01/11/1990|
|10|DUMOULIN      |JEAN-CHRISTOPHE|22/08/1960|
|11|LABONNE-JAYAT |OLIVIER        |23/09/1960|
|12|DE-LA-FONTAINE|JEAN           |22/01/1905|
|13|LEVY          |SAMUEL         |27/03/1959|
|14|DE-LA-RUE     |LAURENCE       |13/12/1989|
|15|DUPONT        |JEAN           |15/10/1960|
|16|MARTIN        |ALBERT         |15/08/1989|
+--+--------------+---------------+----------+
16 rows in set (0,00 sec)
```

Parmi les nombreux formats, le format « %d/%m/%Y » présente la date au format JJ/MM/AAAA. Pour l'heure, un format tel que « %Hh%imin%ssec » retournerait 23h54min34sec.

DATEDIFF

Cette fonction calcule le nombre de jours entre deux dates. Ainsi la syntaxe DATEDIFF(NOW(),Date_Naissance) donne l'âge de la personne en nombre de jours. Si on divise le résultat par 365, et qu'on prenne la partie entière, alors on obtient l'âge de la personne (en années). Voici cette syntaxe :

```
mysql> SELECT
ID,Nom,Prenom,Age,TRUNCATE((DATEDIFF(NOW(),Date_Naissance)/365
),0) AS Age_calculé FROM clients;
+--+--------------+---------------+---+------------+
|ID|Nom           |Prenom         |Age|Age_calculé |
+--+--------------+---------------+---+------------+
| 1|DUPONT        |JEAN           | 27|          27|
| 2|JACQUENOD     |JEAN-CHRISTOPHE| 54|          54|
| 3|MURCIAN       |CAROLE         | 44|          44|
| 4|LERY          |JEAN-MICHEL    | 25|          25|
| 5|DE-LA-RUE     |JEAN-CHRISTOPHE| 23|          23|
| 6|MARTIN        |PAUL-DAVID     | 23|          23|
| 7|MARTIN        |PIERRE         | 56|          56|
| 8|JACQUENOD     |FREDERIC       | 25|          25|
| 9|JACQUENOD     |LAURENCE       | 24|          24|
|10|DUMOULIN      |JEAN-CHRISTOPHE| 54|          54|
|11|LABONNE-JAYAT |OLIVIER        | 54|          54|
|12|DE-LA-FONTAINE|JEAN           |110|         110|
|13|LEVY          |SAMUEL         | 56|          56|
|14|DE-LA-RUE     |LAURENCE       | 25|          25|
|15|DUPONT        |JEAN           | 54|          54|
|16|MARTIN        |ALBERT         | 25|          25|
+--+--------------+---------------+---+------------+
16 rows in set (0,00 sec)
```

Remarque

Ce calcul montre qu'il est inutile de conserver une colonne « Age », puisqu'il peut être déduit de la date de naissance. C'est même une erreur, car l'âge change selon la date du moment, seule la date de naissance est immuable dans le temps. **Seule la date de naissance doit être présente dans la table clients.**

Les fonctions MySQL d'information

MySQL propose quelques fonctions retournant des informations sur les dernières opérations, sur les bases de données ou les tables.

Information sur MySQL, les utilisateurs et la base de données

VERSION

Cette fonction retourne une chaîne de caractère UTF8 indiquant la version de MySQL. En voici un exemple :

```
mysql> SELECT VERSION();
+---------+
|VERSION()|
+---------+
|5.6.21   |
+---------+
1 row in set (0,00 sec)
```

USER, SYSTEM_USER, SESSION_USER

Cette fonction retourne une chaîne de caractère UTF8 indiquant quel est l'utilisateur et le nom de l'ordinateur client utilisé pour la connexion. USER(), SYSTEM_USER() ou SESSION_USER() sont des synonymes. En voici un exemple :

```
mysql> SELECT USER();
+--------------+
|USER()        |
+--------------+
|root@localhost|
+--------------+
1 row in set (0,01 sec)
```

CURRENT_USER

Cette fonction retourne une chaîne de caractère UTF8 indiquant quels sont l'utilisateur et le nom de l'ordinateur que le serveur utilise pour identifier le client. La valeur peut être différente de USER().

SCHEMA, DATABASE

Cette fonction retourne une chaîne de caractère UTF8 indiquant la base de données courante ou NULL si aucune base de données n'est utilisée. SCHEMA() ou DATABASE() sont des synonymes. En voici un exemple :

```
mysql > SELECT DATABASE();
+----------+
|DATABASE()|
+----------+
|coursphp  |
+----------+
1 row in set (0,00 sec)
```

CONNECTION_ID

Cette fonction retourne un numéro entier, identifiant unique de connexion. En voici un exemple :

```
mysql> SELECT CONNECTION_ID();
+---------------+
```

```
|CONNECTION_ID()|
+---------------+
|              1|
+---------------+
1 row in set (0,00 sec)
```

BENCHMARK

Cette fonction exécute « nb » fois le traitement « expression ». Elle évalue la performance de MySQL. En voici un exemple :

```
mysql> SELECT BENCHMARK(1000000,ENCODE('bonjour','au
revoir'));
+------------------------------------------------+
|BENCHMARK(1000000,ENCODE('bonjour','au revoir'))|
+------------------------------------------------+
|                                               0|
+------------------------------------------------+
1 row in set (0,13 sec)
```

CHARSET

Cette fonction indique le jeu de caractères utilisé par l'argument. Appliquée sur un texte sans accents, elle affiche la table par défaut. En voici deux exemples :

```
mysql> SELECT CHARSET('bonjour');
+------------------+
|CHARSET('bonjour')|
+------------------+
|utf8              |
+------------------+
1 row in set (0,00 sec)

mysql> SELECT CHARSET(CONVERT('bonjour' USING latin1));
+----------------------------------------+
|CHARSET(CONVERT('bonjour' USING latin1))|
+----------------------------------------+
|latin1                                  |
+----------------------------------------+
1 row in set (0,00 sec)
```

COERCIBILITY

Cette fonction indique la coercibilité de la chaîne en argument. Cela définit quel jeu de caractères serait utilisé en cas de regroupement de deux résultats comme par exemple avec la clause UNION. La valeur de coercibilité la plus faible sera utilisée comme référence, et son jeu de caractères sera prioritaire pour la conversion du résultat final. Les valeurs retournées sont :

- 0 : Le jeu de caractères est défini explicitement, via une clause COLLATE ;

- 1 : Aucun jeu de caractères à utiliser en particulier. Les chaînes sont concaténées avec différents jeux de caractères ;
- 2 : Le jeu de caractères est implicite, il est défini par la valeur de la colonne ;
- 3 : Si l'argument est une constante système. Par exemple avec la fonction USER() ;
- 4 : Dans le cas d'une chaîne littérale ;
- 5 : À ignorer. Par exemple avec une valeur comme NULL.

  En voici des exemples :

```
mysql> SELECT COERCIBILITY('bonjour');
+-----------------------+
|COERCIBILITY('bonjour')|
+-----------------------+
|                      4|
+-----------------------+
1 row in set (0,00 sec)

mysql> SELECT COERCIBILITY(USER());
+--------------------+
|COERCIBILITY(USER())|
+--------------------+
|                   3|
+--------------------+
1 row in set (0,00 sec)

mysql> SELECT COERCIBILITY('bonjour' COLLATE utf8_general_ci);
+-----------------------------------------------+
|COERCIBILITY('bonjour' COLLATE utf8_general_ci)|
+-----------------------------------------------+
|                                              0|
+-----------------------------------------------+
1 row in set (0,00 sec)
```

COLLATION

Cette fonction indique le jeu de caractères (collation) utilisé pour la chaîne en argument. La clause SQL COLLATE redéfinit ponctuellement le jeu de caractères (collation) à utiliser par exemple pour une comparaison. En voici des exemples :

```
mysql> SELECT Nom COLLATE utf8_spanish_ci AS Nom1 FROM
personnes ORDER BY Nom1;
+--------------+
|Nom1          |
+--------------+
|DE-LA-FONTAINE|
|DE-LA-RUE     |
|DE-LA-RUE     |
|DUMONTEL      |
|DUMOULIN      |
```

```
|DUPONT         |
|DUPONT         |
|JACQUENOD      |
|JACQUENOD      |
|JACQUENOD      |
|KACZMA         |
|LABONNE-JAYAT  |
|LERY           |
|LEVY           |
|MARTIN         |
|MARTIN         |
|MARTIN         |
|MURCIAN        |
+---------------+
18 rows in set (0,01 sec)
mysql> SELECT COLLATION('bonjour');
+--------------------+
|COLLATION('bonjour')|
+--------------------+
|utf8_general_ci     |
+--------------------+
1 row in set (0,00 sec)

mysql> SELECT COLLATION(_latin1'bonjour');
+---------------------------+
|COLLATION(_latin1'bonjour')|
+---------------------------+
|latin1_swedish_ci          |
+---------------------------+
1 row in set (0,00 sec)
```

Information sur les dernières opérations

FOUND_ROWS

Cette fonction retourne un entier correspondant au nombre de lignes trouvées dans la requête SELECT précédente avec l'option SQL_CAL_FOUND_ROWS. En voici un exemple :

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM personnes WHERE Age >
40 LIMIT 10;
+--+--------------+---------------+---+
|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|
+--+--------------+---------------+---+
8 rows in set (0,00 sec)

mysql> SELECT FOUND_ROWS();
+-------------+
|FOUND_ROWS()|
+-------------+
|           8|
+-------------+
1 row in set (0,00 sec)
```

ROWS_COUNT

Cette fonction retourne un entier correspondant au nombre de lignes changées, supprimées ou insérées par la dernière instruction UPDATE, DELETE ou INSERT. En voici un exemple :

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-100 WHERE
Solde>200;
Query OK, 23 rows affected (0,00 sec)
Rows matched: 23  Changed: 23  Warnings: 0

mysql> SELECT ROW_COUNT();
+------------+
|ROW_COUNT()|
+------------+
|          23|
+------------+
1 row in set (0,00 sec)
```

LAST_INSERT_ID

Cette fonction retourne le numéro du premier indice AUTOINCREMENT utilisé lors de la dernière insertion de donnée. En voici un exemple :

```
mysql> SELECT * FROM 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|
| 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|
+--+--------------+---------------+---+
16 rows in set (0,00 sec)

mysql> INSERT INTO personnes (Nom,Prenom,Age) VALUES
('KACZMA','HELENE',52),('DUMONTEL','FRANCK',23);
Query OK, 2 rows affected (0,00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT LAST_INSERT_ID();
+----------------+
|LAST_INSERT_ID()|
+----------------+
|              17|
+----------------+
1 row in set (0,00 sec)

mysql> SELECT * FROM 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|
| 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|
|17|KACZMA        |HELENE         | 52|
|18|DUMONTEL      |FRANCK         | 23|
+--+--------------+---------------+---+
18 rows in set (0,00 sec)
```

Les jointures entre tables

Le langage SQL permet la mise en relation de tables via un champ commun à travers la *jointure*. Pour sa mise en œuvre, nous modifions la table des clients d'une banque, utilisée précédemment, afin que le solde des comptes bancaires soit calculé à partir d'une autre table de comptes des clients.

Les tables support

Voici la nouvelle structure des tables utilisées pour cette section. Deux tables sont créées : « clients_bancaires » et « comptes_bancaires ».

```
mysql> SHOW TABLES;
+------------------+
|Tables_in_coursphp|
+------------------+
|clients           |
|clients_bancaires |
|comptes_bancaires |
|personnes         |
+------------------+
4 rows in set (0,00 sec)
```

La table « clients_bancaires »

La table « clients_bancaires » contient la liste des clients de la banque. Elle est créée à partir de la table « clients » après avoir supprimé les colonnes « Age » et « Solde ». Sa structure est présentée à la figure 10-1.3 et ses enregistrements à la figure 10-1.4.

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	ID_Clt	int(11)		UNSIGNED	Non	*Aucune*	AUTO_INCREMENT
2	Nom	varchar(255)	utf8_general_ci		Non	*Aucune*	
3	Prenom	varchar(255)	utf8_general_ci		Non	*Aucune*	
4	Date_Naissance	date			Oui	*NULL*	
5	Etat_Civil	enum('Marié', 'Célibataire', 'Veuf', 'Divorcé', 'D	utf8_general_ci		Oui	*NULL*	
6	Nb_Enfants	int(2)		UNSIGNED	Non	*Aucune*	

Figure 10-1.3

Structure de la table clients_bancaires.

ID_Clt	Nom	Prenom	Date_Naissance	Etat_Civil	Nb_Enfants
1	DUPONT	JEAN	1987-12-28	Marié	2
2	JACQUENOD	JEAN-CHRISTOPHE	1961-02-10	Marié	1
3	MURCIAN	CAROLE	1970-10-20	Célibataire	1
4	LERY	JEAN-MICHEL	1989-05-07	Marié	2
5	DE-LA-RUE	JEAN-CHRISTOPHE	1991-06-18	Divorcé	0
6	MARTIN	PAUL-DAVID	1991-08-22	Célibataire	0
7	MARTIN	PIERRE	1959-01-18	Veuf	3
8	JACQUENOD	FREDERIC	1989-11-27	Marié	0
9	JACQUENOD	LAURENCE	1990-11-01	Marié	0
10	DUMOULIN	JEAN-CHRISTOPHE	1960-08-22	Marié	2
11	LABONNE-JAYAT	OLIVIER	1960-09-23	Célibataire	1
12	DE-LA-FONTAINE	JEAN	1905-01-22	Décédé	0
13	LEVY	SAMUEL	1959-03-27	Divorcé	3
14	DE-LA-RUE	LAURENCE	1989-12-13	Marié	1
15	DUPONT	JEAN	1960-10-15	Veuf	2
16	MARTIN	ALBERT	1989-08-15	Célibataire	1
17	ROUSSE	JACQUES	1990-11-05	Célibataire	0

Figure 10-1.4

Enregistrements de la table clients_bancaires.

La table « comptes_bancaires »

La table « comptes_bancaires » contient les champs suivants :

1. ID_Cpt : Un identifiant unique interne du compte ;
2. Agence : Le code de l'agence bancaire, sur 5 caractères alphanumériques ;
3. Numero : Le numéro du compte bancaire, sur 7 caractères alphanumériques ;
4. Type : le type du compte : liste de type comme Compte_Dépôts, Livret_A, …
5. Libelle : Le libellé du compte ;
6. ID_Clt : L'identifiant du client dans la table « clients_bancaires » ;
7. Solde : Le solde restant sur ce compte bancaire.

Remarque

Le numéro de compte devrait être unique. Cependant, pour une présentation qui différencie le solde actuel avec le solde potentiel, la carte bancaire à débit différée est présentée à part du compte, mais possède le même numéro de compte. L'unicité du numéro de compte n'est donc pas possible avec cette modélisation.

La figure 10-1.5 présente sa structure, et la figure 10-1.6, une partie de ses enregistrements.

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	ID_Cpt	int(11)		UNSIGNED	Non	*Aucune*	AUTO_INCREMENT
2	Agence	varchar(5)	utf8_general_ci		Non	*Aucune*	
3	Numero	varchar(7)	utf8_general_ci		Non	*Aucune*	
4	Type	enum('Compte_Dépôts', 'Carte_Différé', 'Livret_A',	utf8_general_ci		Oui	*NULL*	
5	Libelle	varchar(30)	utf8_general_ci		Non	*Aucune*	
6	ID_Clt	int(11)		UNSIGNED	Non	*Aucune*	
7	Solde	float			Non	*Aucune*	

Figure 10-1.5

Structure de la table comptes_bancaires.

ID_Cpt	Agence	Numero	Type	Libelle	ID_Clt	Solde
1	00602	165143P	Compte_Dépôts	Compte de dépôts	1	550.98
2	00602	165143P	Carte_Différé	Carte à débit différé	1	-115.8
3	00602	116476Q	Livret_A	Livret A	1	765.32
4	00523	025123R	Compte_Dépôts	Compte de dépôts	2	-140.17
5	00523	025123R	Carte_Différé	Carte à débit différé	2	-200
6	00523	790327V	Livret_Banque	Compte sur Livret	2	31.3
7	00602	154123P	Compte_Dépôts	Compte de dépôts	3	3185.08
8	00602	154123P	Carte_Différé	Carte à débit différé	3	-104.1
9	00602	102476Q	Livret_A	Livret A	3	120
10	00602	921029R	Livret_Banque	Compte sur Livret	3	50
11	00602	413621M	Livret_Jeune	Livret Jeune	3	298
12	00521	032154P	Compte_Dépôts	Compte de dépôts	4	-688.98
13	00521	139390R	Livret_Banque	Compte sur Livret	4	50
14	00521	321747M	Livret_Jeune	Livret Jeune	4	500
15	00521	002551B	Livret_Dév_Dur	Livret de Dév. Durable	4	120

Figure 10-1.6

Enregistrements de la table comptes_bancaires.

Voici l'ensemble de ses enregistrements. Certains libellés ont été modifiés ou tronqués pour une meilleure lisibilité.

```
mysql> SELECT * FROM comptes_bancaires;
+---+-----+-------+--------------+--------------+---+--------+
|ID_|Agenc|Numero |Type          |Libelle       |ID_|Solde   |
|Cpt|e    |       |              |              |Clt|        |
+---+-----+-------+--------------+--------------+---+--------+
|  1|00602|165143P|Compte_Dépôts |Compte de dépô|  1|  550.98|
|  2|00602|165143P|Carte_Différé |Carte débit di|  1|  -115.8|
|  3|00602|116476Q|Livret_A      |Livret A      |  1|  765.32|
|  4|00523|025123R|Compte_Dépôts |Compte de dépô|  2| -140.17|
|  5|00523|025123R|Carte_Différé |Carte débit di|  2|    -200|
|  6|00523|790327V|Livret_Banque |Compte sur Liv|  2|    31.3|
|  7|00602|154123P|Compte_Dépôts |Compte de dépô|  3| 3185.08|
```

```
|  8|00602|154123P|Carte_Différé |Carte débit di|  3|  -104.1|
|  9|00602|102476Q|Livret_A      |Livret A      |  3|     120|
| 10|00602|921029R|Livret_Banque |Compte sur Liv|  3|      50|
| 11|00602|413621M|Livret_Jeune  |Livret Jeune  |  3|     298|
| 12|00521|032154P|Compte_Dépôts |Compte de dépô|  4| -688.98|
| 13|00521|139390R|Livret_Banque |Compte sur Liv|  4|      50|
| 14|00521|321747M|Livret_Jeune  |Livret Jeune  |  4|     500|
| 15|00521|002551B|Livret_Dév_Dur|Livret Dév.Dur|  4|     120|
| 16|00523|123456J|Compte_Dépôts |Compte de dépô|  5|   94.68|
| 17|00523|123456J|Carte_Différé |Carte débit di|  5| -122.12|
| 18|00523|615243H|Compte_Dépôts |Compte de dépô|  6|  406.21|
| 19|00523|615243H|Carte_Différé |Carte débit di|  6|    -200|
| 20|00521|062332P|Compte_Dépôts |Compte de dépô|  7| 1790.22|
| 21|00521|062332P|Carte_Différé |Carte débit di|  7| -555.66|
| 22|00521|889261D|Compte_Dépôts |Compte de dépô|  8|  394.87|
| 23|00521|889261D|Carte_Différé |Carte débit di|  8| -552.87|
| 24|00521|009060K|Livret_A      |Livret A      |  8|  590.98|
| 25|00521|545823Z|Compte_Dépôts |Compte de dépô|  9| -679.08|
| 26|00521|545823Z|Carte_Différé |Carte débit di|  9| -276.21|
| 27|00521|104721W|Livret_A      |Livret A      |  9|     200|
| 28|00521|921116A|Livret_Banque |Compte sur Liv|  9|   52.11|
| 29|00521|415921B|Livret_Jeune  |Livret Jeune  |  9|     400|
| 30|00521|812005Q|Livret_Dév_Dur|Livret Dév.Dur|  9|     100|
| 31|00523|823452N|Compte_Dépôts |Compte de dépô| 10|-2186.86|
| 32|00523|823452N|Carte_Différé |Carte débit di| 10|       0|
| 33|00523|238245E|Compte_Dépôts |Compte de dépô| 11|  234.02|
| 34|00523|238245E|Carte_Différé |Carte débit di| 11|    -300|
| 35|00602|458263T|Compte_Dépôts |Compte de dépô| 12| 1825.54|
| 36|00523|904161A|Compte_Dépôts |Compte de dépô| 13|   12.09|
| 37|00523|904161A|Carte_Différé |Carte débit di| 13| -212.98|
| 38|00523|219071L|Livret_A      |Livret A      | 13|  432.76|
| 39|00521|045123P|Compte_Dépôts |Compte de dépô| 14|   275.7|
| 40|00521|045123P|Carte_Différé |Carte débit di| 14|  -104.1|
| 41|00521|014276Q|Livret_A      |Livret A      | 14| 1032.47|
| 42|00521|290129R|Livret_Banque |Compte sur Liv| 14|    31.3|
| 43|00521|146321M|Livret_Jeune  |Livret Jeune  | 14|  818.38|
| 44|00521|401002B|Livret_Dév_Dur|Livret Dév.Dur| 14|   82.23|
| 45|00523|987123P|Compte_Dépôts |Compte de dépô| 15|  4572.1|
| 46|00523|987123P|Carte_Différé |Carte débit di| 15|-2987.65|
| 47|00523|207275Q|Livret_A      |Livret A      | 15|    2500|
| 48|00523|297820R|Livret_Banque |Compte sur Liv| 15| 5628.34|
| 49|00523|245421M|Livret_Jeune  |Livret Jeune  | 15|    1600|
| 50|00523|502014B|Livret_Dév_Dur|Livret Dév.Dur| 15| 1002.11|
| 51|00602|004452N|Compte_Dépôts |Compte de dépô| 16|  363.49|
| 52|00602|004452N|Carte_Différé |Carte débit di| 16|    -150|
| 53|00602|084852A|Compte_Dépôts |Compte de dépô| 26|  665.29|
| 54|00602|084852A|Carte_Différé |Carte débit di| 26|    -320|
+---+-----+-------+--------------+--------------+---+--------+
```

```
54 rows in set (0,00 sec)
```

Relation entre les tables

La colonne « ID_Clt » de la table « comptes_bancaires » met en relation cette table avec la table « clients_bancaires » dans laquelle « ID_Clt » est la clef identifiant le propriétaire du compte (figure 10-1.7). Cet identifiant donne accès à toutes les informations du propriétaire, son nom, son prénom, sa date de naissance, dans la table « clients_bancaires ». La jointure met en relation ces deux tables et recherche des informations sur l'une ou sur l'autre.

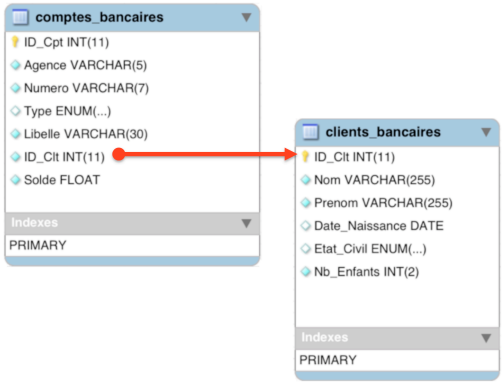


Figure 10-1.7

Relation entre les tables clients_bancaires et comptes_bancaires.

Les types de jointure

Il existe deux types de jointures :

- *Les jointures internes* : Elles ne sélectionnent que les données qui possèdent une correspondance entre les deux tables. Les éléments absents dans l'une des tables n'apparaissent pas dans le résultat de la requête ;
  C'est le cas de JACQUES ROUSSE ayant comme ID_Clt la valeur 17 dans la table « Clients_bancaires » et qui n'a aucun compte bancaire. Son identifiant n'apparaît pas dans la colonne ID_Clt de la table « Comptes_Bancaires ».
  De même les comptes bancaires ayant l'ID_Cpt N°53 et 54 ont comme propriétaire le client N°26 qui est absent de la table « Clients_bancaires ».
- *Les jointures externes* : Elles sélectionnent toutes les données, même celles qui n'ont aucune correspondance dans l'autre table.

Mise en œuvre de la jointure interne

Avec *WHERE*

La syntaxe suivante affiche pour chaque compte bancaire, le nom et le prénom du client ainsi que le libellé du compte bancaire.

Il est préférable d'identifier clairement la table de chaque champ. Ainsi le nom et le prénom du client se notent respectivement « clients_bancaires.Nom » et « clients_bancaires.Prenom », et le libellé du compte se note « comptes_bancaires.libelle ». De cette manière, il n'y a aucune ambiguïté sur la table à utiliser, même avec un même nom de champ utilisé dans les deux tables.

Le FROM est suivi de la liste des tables à utiliser. La clause WHERE indique les champs à mettre en correspondance.

Remarque

L'identifiant ID_Clt, peut se nommer différemment entre les deux tables puisque c'est la clause WHERE qui indique la relation à prendre en compte.

Voici la requête SQL et son résultat :

```
mysql> SELECT
clients_bancaires.Nom,clients_bancaires.Prenom,comptes_bancair
es.libelle,comptes_bancaires.Solde FROM clients_bancaires,
comptes_bancaires WHERE
clients_bancaires.ID_Clt=comptes_bancaires.ID_Clt;
+--------------+---------------+--------------+--------+
|Nom           |Prenom         |libelle       |Solde   |
+--------------+---------------+--------------+--------+
|DUPONT        |JEAN           |Compte de dépô|  550.98|
|DUPONT        |JEAN           |Carte débit di|  -115.8|
|DUPONT        |JEAN           |Livret A      |  765.32|
|JACQUENOD     |JEAN-CHRISTOPHE|Compte de dépô| -140.17|
|JACQUENOD     |JEAN-CHRISTOPHE|Carte débit di|    -200|
|JACQUENOD     |JEAN-CHRISTOPHE|Compte sur Liv|    31.3|
|MURCIAN       |CAROLE         |Compte de dépô| 3185.08|
|MURCIAN       |CAROLE         |Carte débit di|  -104.1|
|MURCIAN       |CAROLE         |Livret A      |     120|
|MURCIAN       |CAROLE         |Compte sur Liv|      50|
|MURCIAN       |CAROLE         |Livret Jeune  |     298|
|LERY          |JEAN-MICHEL    |Compte de dépô| -688.98|
|LERY          |JEAN-MICHEL    |Compte sur Liv|      50|
|LERY          |JEAN-MICHEL    |Livret Jeune  |     500|
|LERY          |JEAN-MICHEL    |Livret Dév.Dur|     120|
|DE-LA-RUE     |JEAN-CHRISTOPHE|Compte de dépô|   94.68|
|DE-LA-RUE     |JEAN-CHRISTOPHE|Carte débit di| -122.12|
|MARTIN        |PAUL-DAVID     |Compte de dépô|  406.21|
|MARTIN        |PAUL-DAVID     |Carte débit di|    -200|
|MARTIN        |PIERRE         |Compte de dépô| 1790.22|
|MARTIN        |PIERRE         |Carte débit di| -555.66|
|JACQUENOD     |FREDERIC       |Compte de dépô|  394.87|
|JACQUENOD     |FREDERIC       |Carte débit di| -552.87|
|JACQUENOD     |FREDERIC       |Livret A      |  590.98|
|JACQUENOD     |LAURENCE       |Compte de dépô| -679.08|
|JACQUENOD     |LAURENCE       |Carte débit di| -276.21|
```

```
|JACQUENOD     |LAURENCE       |Livret A      |     200|
|JACQUENOD     |LAURENCE       |Compte sur Liv|   52.11|
|JACQUENOD     |LAURENCE       |Livret Jeune  |     400|
|JACQUENOD     |LAURENCE       |Livret Dév.Dur|     100|
|DUMOULIN      |JEAN-CHRISTOPHE|Compte de dépô|-2186.86|
|DUMOULIN      |JEAN-CHRISTOPHE|Carte débit di|       0|
|LABONNE-JAYAT |OLIVIER        |Compte de dépô|  234.02|
|LABONNE-JAYAT |OLIVIER        |Carte débit di|    -300|
|DE-LA-FONTAINE|JEAN           |Compte de dépô| 1825.54|
|LEVY          |SAMUEL         |Compte de dépô|   12.09|
|LEVY          |SAMUEL         |Carte débit di| -212.98|
|LEVY          |SAMUEL         |Livret A      |  432.76|
|DE-LA-RUE     |LAURENCE       |Compte de dépô|   275.7|
|DE-LA-RUE     |LAURENCE       |Carte débit di|  -104.1|
|DE-LA-RUE     |LAURENCE       |Livret A      | 1032.47|
|DE-LA-RUE     |LAURENCE       |Compte sur Liv|    31.3|
|DE-LA-RUE     |LAURENCE       |Livret Jeune  |  818.38|
|DE-LA-RUE     |LAURENCE       |Livret Dév.Dur|   82.23|
|DUPONT        |JEAN           |Compte de dépô|  4572.1|
|DUPONT        |JEAN           |Carte débit di|-2987.65|
|DUPONT        |JEAN           |Livret A      |    2500|
|DUPONT        |JEAN           |Compte sur Liv| 5628.34|
|DUPONT        |JEAN           |Livret Jeune  |    1600|
|DUPONT        |JEAN           |Livret Dév.Dur| 1002.11|
|MARTIN        |ALBERT         |Compte de dépô|  363.49|
|MARTIN        |ALBERT         |Carte débit di|    -150|
+--------------+---------------+--------------+--------+
52 rows in set (0,00 sec)
```

Calculons maintenant le solde de chaque client. Travaillons d'abord sur la table « comptes_bancaires », en affichant le total du solde par numéro de client grâce aux fonctions SUM() et ROUND() associées à la clause GROUP BY.

```
mysql> SELECT ID_Clt,ROUND(SUM(Solde),2) AS Solde_Total FROM
comptes_bancaires GROUP BY ID_Clt;
+------+-----------+
|ID_Clt|Solde_Total|
+------+-----------+
|     1|    1200.50|
|     2|    -308.87|
|     3|    3548.98|
|     4|     -18.98|
|     5|     -27.44|
|     6|     206.21|
|     7|    1234.56|
|     8|     432.98|
|     9|    -203.18|
|    10|   -2186.86|
|    11|     -65.98|
```

```
|    12|    1825.54|
|    13|     231.87|
|    14|    2135.98|
|    15|   12314.90|
|    16|     213.49|
|    26|     345.29|
+------+-----------+
17 rows in set (0,00 sec)
```

Réécrivons ensuite cette requête en précisant le nom de la table devant le nom de chaque champ :

```
mysql> SELECT
comptes_bancaires.ID_Clt,ROUND(SUM(comptes_bancaires.Solde),2)
AS Solde_Total FROM comptes_bancaires GROUP BY
comptes_bancaires.ID_Clt;
```

Si la base de données n'a pas été sélectionnée auparavant (`USE CoursPHP;`), on l'indique explicitement :

```
mysql> SELECT
CoursPHP.comptes_bancaires.ID_Clt,ROUND(SUM(CoursPHP.comptes_b
ancaires.Solde),2) AS Solde_Total FROM
CoursPHP.comptes_bancaires GROUP BY
CoursPHP.comptes_bancaires.ID_Clt;
```

Ces deux syntaxes affichent le même résultat !

Une fois ce résultat obtenu, il suffit d'ajouter le nom et le prénom du client pour chaque identifiant ID_Clt, en indiquant de le chercher dans la table « clients_bancaires ». Cela s'obtient en indiquant le nom de la table devant le nom du champ et en ajoutant le nom de la table dans la liste de la clause FROM. La clause WHERE effectue la jointure en mettant en relation les champs « ID_Clt » des deux tables. La syntaxe devient :

```
mysql> SELECT
comptes_bancaires.ID_Clt,clients_bancaires.Nom,clients_bancair
es.Prenom,ROUND(SUM(comptes_bancaires.Solde),2) AS Solde_Total
FROM comptes_bancaires,clients_bancaires WHERE
comptes_bancaires.ID_Clt=clients_bancaires.ID_Clt GROUP BY
comptes_bancaires.ID_Clt;
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|     1|DUPONT        |JEAN           |    1200.50|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     3|MURCIAN       |CAROLE         |    3548.98|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
```

```
|     7|MARTIN         |PIERRE          |    1234.56|
|     8|JACQUENOD      |FREDERIC        |     432.98|
|     9|JACQUENOD      |LAURENCE        |    -203.18|
|    10|DUMOULIN       |JEAN-CHRISTOPHE|   -2186.86|
|    11|LABONNE-JAYAT  |OLIVIER         |     -65.98|
|    12|DE-LA-FONTAINE|JEAN            |    1825.54|
|    13|LEVY           |SAMUEL          |     231.87|
|    14|DE-LA-RUE      |LAURENCE        |    2135.98|
|    15|DUPONT         |JEAN            |   12314.90|
|    16|MARTIN         |ALBERT          |     213.49|
+------+--------------+---------------+-----------+
16 rows in set (0,00 sec)
```

Cette syntaxe peut être simplifiée par l'utilisation d'alias via la clause AS, comme « cb » pour représenter la table « comptes_bancaires », et « cl » pour indiquer la table « clients_bancaires » dans la clause FROM. Voici la récriture des syntaxes précédentes :

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

Le mot-clef AS étant facultatif il peut être supprimé. La syntaxe devient :

```
mysql> 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;
```

La syntaxe suivante affiche pour chaque personne le solde de son compte de dépôt et de sa carte bancaire différée.

```
mysql> SELECT cb.ID_Clt,cl.Nom,cl.Prenom,cb.Type,cb.Solde AS
Solde FROM comptes_bancaires AS cb,clients_bancaires AS cl
WHERE cb.ID_Clt=cl.ID_Clt AND (cb.Type="Compte_Dépôts" OR
cb.Type="Carte_Différé");
+------+--------------+---------------+-------------+--------+
|ID_Clt|Nom           |Prenom         |Type         |Solde   |
+------+--------------+---------------+-------------+--------+
|     1|DUPONT        |JEAN           |Compte_Dépôts|  550.98|
|     1|DUPONT        |JEAN           |Carte_Différé|  -115.8|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|Compte_Dépôts| -140.17|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|Carte_Différé|    -200|
|     3|MURCIAN       |CAROLE         |Compte_Dépôts| 3185.08|
|     3|MURCIAN       |CAROLE         |Carte_Différé|  -104.1|
|     4|LERY          |JEAN-MICHEL    |Compte_Dépôts| -688.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|Compte_Dépôts|   94.68|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|Carte_Différé| -122.12|
```

```
|     6|MARTIN        |PAUL-DAVID     |Compte_Dépôts|  406.21|
|     6|MARTIN        |PAUL-DAVID     |Carte_Différé|    -200|
|     7|MARTIN        |PIERRE         |Compte_Dépôts| 1790.22|
|     7|MARTIN        |PIERRE         |Carte_Différé| -555.66|
|     8|JACQUENOD     |FREDERIC       |Compte_Dépôts|  394.87|
|     8|JACQUENOD     |FREDERIC       |Carte_Différé| -552.87|
|     9|JACQUENOD     |LAURENCE       |Compte_Dépôts| -679.08|
|     9|JACQUENOD     |LAURENCE       |Carte_Différé| -276.21|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|Compte_Dépôts|-2186.86|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|Carte_Différé|       0|
|    11|LABONNE-JAYAT |OLIVIER        |Compte_Dépôts|  234.02|
|    11|LABONNE-JAYAT |OLIVIER        |Carte_Différé|    -300|
|    12|DE-LA-FONTAINE|JEAN           |Compte_Dépôts| 1825.54|
|    13|LEVY          |SAMUEL         |Compte_Dépôts|   12.09|
|    13|LEVY          |SAMUEL         |Carte_Différé| -212.98|
|    14|DE-LA-RUE     |LAURENCE       |Compte_Dépôts|   275.7|
|    14|DE-LA-RUE     |LAURENCE       |Carte_Différé|  -104.1|
|    15|DUPONT        |JEAN           |Compte_Dépôts|  4572.1|
|    15|DUPONT        |JEAN           |Carte_Différé|-2987.65|
|    16|MARTIN        |ALBERT         |Compte_Dépôts|  363.49|
|    16|MARTIN        |ALBERT         |Carte_Différé|    -150|
+------+--------------+---------------+-------------+--------+
30 rows in set (0,00 sec)
```

La syntaxe suivante cumule le solde du compte de dépôt et de la carte bancaire différée, soit le solde de fin de mois.

```
mysql> SELECT
cb.ID_Clt,cl.Nom,cl.Prenom,cb.Type,ROUND(SUM(cb.Solde),2) AS
Solde_Fin_Mois FROM comptes_bancaires AS cb,clients_bancaires
AS cl WHERE cb.ID_Clt=cl.ID_Clt AND (cb.Type="Compte_Dépôts"
OR cb.Type="Carte_Différé") GROUP BY cb.ID_Clt;
+-----+--------------+---------------+-------------+--------+
|ID_Clt|Nom          |Prenom         |Type   |Solde_Fin_Mois|
+-----+--------------+---------------+-------------+--------+
|    1|DUPONT        |JEAN           |Compte_Dépôts|  435.18|
|    2|JACQUENOD     |JEAN-CHRISTOPHE|Compte_Dépôts| -340.17|
|    3|MURCIAN       |CAROLE         |Compte_Dépôts| 3080.98|
|    4|LERY          |JEAN-MICHEL    |Compte_Dépôts| -688.98|
|    5|DE-LA-RUE     |JEAN-CHRISTOPHE|Compte_Dépôts|  -27.44|
|    6|MARTIN        |PAUL-DAVID     |Compte_Dépôts|  206.21|
|    7|MARTIN        |PIERRE         |Compte_Dépôts| 1234.56|
|    8|JACQUENOD     |FREDERIC       |Compte_Dépôts| -158.00|
|    9|JACQUENOD     |LAURENCE       |Compte_Dépôts| -955.29|
|   10|DUMOULIN      |JEAN-CHRISTOPHE|Compte_Dépôts|-2186.86|
|   11|LABONNE-JAYAT |OLIVIER        |Compte_Dépôts|  -65.98|
|   12|DE-LA-FONTAINE|JEAN           |Compte_Dépôts| 1825.54|
|   13|LEVY          |SAMUEL         |Compte_Dépôts| -200.89|
|   14|DE-LA-RUE     |LAURENCE       |Compte_Dépôts|  171.60|
```

```
|   15|DUPONT        |JEAN           |Compte_Dépôts| 1584.45|
|   16|MARTIN        |ALBERT         |Compte_Dépôts|  213.49|
+-----+--------------+---------------+-------------+--------+
16 rows in set (0,00 sec)
```

Avec *INNER JOIN*

La jointure précédente utilisait la clause WHERE. Cette clause a été utilisée pour faciliter la compréhension, puisque elle avait déjà été présentée, mais elle est devenue obsolète pour les jointures. On lui préfère la syntaxe JOIN plus explicite. La réécriture de :

```
mysql> 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 :

```
mysql> 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;
```

Cette syntaxe indique que les données sont récupérées à partir de la table « comptes_bancaires », et que la jointure interne (INNER JOIN) est effectuée avec la table « clients_bancaires ». La clause ON fait la liaison entre les deux tables. Voici son résultat :

```
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|     1|DUPONT        |JEAN           |    1200.50|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     3|MURCIAN       |CAROLE         |    3548.98|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
|     7|MARTIN        |PIERRE         |    1234.56|
|     8|JACQUENOD     |FREDERIC       |     432.98|
|     9|JACQUENOD     |LAURENCE       |    -203.18|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|   -2186.86|
|    11|LABONNE-JAYAT |OLIVIER        |     -65.98|
|    12|DE-LA-FONTAINE|JEAN           |    1825.54|
|    13|LEVY          |SAMUEL         |     231.87|
|    14|DE-LA-RUE     |LAURENCE       |    2135.98|
|    15|DUPONT        |JEAN           |   12314.90|
|    16|MARTIN        |ALBERT         |     213.49|
+------+--------------+---------------+-----------+
```

```
16 rows in set (0,00 sec)
```

Il est possible d'ajouter des clauses GROUP BY, ORDER BY, LIMIT, après la clause JOIN. Cette syntaxe affiche le résultat précédent par ordre croissant du Solde_Total.

```
mysql> 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 ORDER BY Solde_Total;
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|    10|DUMOULIN      |JEAN-CHRISTOPHE|   -2186.86|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     9|JACQUENOD     |LAURENCE       |    -203.18|
|    11|LABONNE-JAYAT |OLIVIER        |     -65.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
|    16|MARTIN        |ALBERT         |     213.49|
|    13|LEVY          |SAMUEL         |     231.87|
|     8|JACQUENOD     |FREDERIC       |     432.98|
|     1|DUPONT        |JEAN           |    1200.50|
|     7|MARTIN        |PIERRE         |    1234.56|
|    12|DE-LA-FONTAINE|JEAN           |    1825.54|
|    14|DE-LA-RUE     |LAURENCE       |    2135.98|
|     3|MURCIAN       |CAROLE         |    3548.98|
|    15|DUPONT        |JEAN           |   12314.90|
+------+--------------+---------------+-----------+
16 rows in set (0,00 sec)
```

De la même manière :

```
mysql> SELECT
cb.ID_Clt,cl.Nom,cl.Prenom,cb.Type,ROUND(SUM(cb.Solde),2) AS
Solde_Fin_Mois FROM comptes_bancaires AS cb,clients_bancaires
AS cl WHERE cb.ID_Clt=cl.ID_Clt AND (cb.Type="Compte_Dépôts"
OR cb.Type="Carte_Différé") GROUP BY cb.ID_Clt;
```

Devient :

```
mysql> SELECT
cb.ID_Clt,cl.Nom,cl.Prenom,cb.Type,ROUND(SUM(cb.Solde),2) AS
Solde_Fin_Mois FROM comptes_bancaires AS cb INNER JOIN
clients_bancaires AS cl ON cb.ID_Clt=cl.ID_Clt AND
(cb.Type="Compte_Dépôts" OR cb.Type="Carte_Différé") GROUP BY
cb.ID_Clt;
```

Et affiche :

```
+------+--------------+---------------+-------------+---------+
|ID_Clt|Nom           |Prenom         |Type   |Solde_Fin_Mois|
+------+--------------+---------------+-------------+---------+
|     1|DUPONT        |JEAN           |Compte_Dépôts|  435.18|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|Compte_Dépôts| -340.17|
|     3|MURCIAN       |CAROLE         |Compte_Dépôts| 3080.98|
|     4|LERY          |JEAN-MICHEL    |Compte_Dépôts| -688.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|Compte_Dépôts|  -27.44|
|     6|MARTIN        |PAUL-DAVID     |Compte_Dépôts|  206.21|
|     7|MARTIN        |PIERRE         |Compte_Dépôts| 1234.56|
|     8|JACQUENOD     |FREDERIC       |Compte_Dépôts| -158.00|
|     9|JACQUENOD     |LAURENCE       |Compte_Dépôts| -955.29|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|Compte_Dépôts|-2186.86|
|    11|LABONNE-JAYAT |OLIVIER        |Compte_Dépôts|  -65.98|
|    12|DE-LA-FONTAINE|JEAN           |Compte_Dépôts| 1825.54|
|    13|LEVY          |SAMUEL         |Compte_Dépôts| -200.89|
|    14|DE-LA-RUE     |LAURENCE       |Compte_Dépôts|  171.60|
|    15|DUPONT        |JEAN           |Compte_Dépôts| 1584.45|
|    16|MARTIN        |ALBERT         |Compte_Dépôts|  213.49|
+------+--------------+---------------+-------------+---------+
16 rows in set (0,00 sec)
```

Mise en œuvre de la jointure externe avec *LEFT JOIN* et *RIGHT JOIN*

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 située à 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 située à gauche.

Pour comprendre la différence entre la jointure interne et les deux jointures externes, gauches et droites, comparons les résultats des trois syntaxes :

Jointure interne : INNER JOIN

La syntaxe suivante affiche la somme des comptes avec une **jointure interne**. Seules les informations ayant une correspondance entre les deux tables, « comptes_bancaires » et « clients_bancaires », sont affichées.

Ainsi le compte ayant comme client le « ID_Clt » N°26 présent dans la table « comptes_bancaires » mais absent de « clients_bancaires » n'est pas affiché.

De la même manière, le client ayant comme « ID_Clt » le N°17 (JACQUES ROUSSE) dans la table « clients_bancaires » qui ne possède aucun compte dans « comptes_bancaires » n'est pas affiché.

```
mysql> 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;
```

```
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|     1|DUPONT        |JEAN           |    1200.50|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     3|MURCIAN       |CAROLE         |    3548.98|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
|     7|MARTIN        |PIERRE         |    1234.56|
|     8|JACQUENOD     |FREDERIC       |     432.98|
|     9|JACQUENOD     |LAURENCE       |    -203.18|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|   -2186.86|
|    11|LABONNE-JAYAT |OLIVIER        |     -65.98|
|    12|DE-LA-FONTAINE|JEAN           |    1825.54|
|    13|LEVY          |SAMUEL         |     231.87|
|    14|DE-LA-RUE     |LAURENCE       |    2135.98|
|    15|DUPONT        |JEAN           |   12314.90|
|    16|MARTIN        |ALBERT         |     213.49|
+------+--------------+---------------+-----------+
16 rows in set (0,00 sec)
```

Jointure externe gauche : LEFT JOIN

La syntaxe suivante affiche la somme des comptes avec une **jointure externe sur la table de gauche**, soit la table « comptes_bancaires ».Toutes les informations de la table de gauche « comptes_bancaires » avec ou sans correspondance dans la table de droite « clients_bancaires » sont affichées.

Ainsi le compte ayant comme « ID_Clt » le client N°26 dans la table située à gauche de JOIN, « comptes_bancaires », est affiché sans aucun Nom ni prénom, et un solde de 345,29 € alors que le client N°26 (ID_Clt) est absent de la table située à droite de JOIN, « clients_bancaires ».

```
mysql> 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;
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|     1|DUPONT        |JEAN           |    1200.50|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     3|MURCIAN       |CAROLE         |    3548.98|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
|     7|MARTIN        |PIERRE         |    1234.56|
|     8|JACQUENOD     |FREDERIC       |     432.98|
|     9|JACQUENOD     |LAURENCE       |    -203.18|
```

```
|    10|DUMOULIN      |JEAN-CHRISTOPHE|   -2186.86|
|    11|LABONNE-JAYAT |OLIVIER        |     -65.98|
|    12|DE-LA-FONTAINE|JEAN           |    1825.54|
|    13|LEVY          |SAMUEL         |     231.87|
|    14|DE-LA-RUE     |LAURENCE       |    2135.98|
|    15|DUPONT        |JEAN           |   12314.90|
|    16|MARTIN        |ALBERT         |     213.49|
|    26|NULL          |NULL           |     345.29|
+------+--------------+---------------+-----------+
17 rows in set (0,00 sec)
```

Jointure externe droite : RIGHT JOIN

La syntaxe suivante affiche la somme des comptes avec une **jointure externe sur la table de droite**, soit la table « clients_bancaires ».Toutes les informations de la table « clients_bancaires » avec ou sans correspondance dans la table de gauche « comptes_bancaires » sont affichées.

Ainsi le client JACQUES ROUSSE ayant pour « ID_Clt » le N°17 dans la table située à droite de JOIN, « clients_bancaires », est affiché avec un solde NULL puisqu'il est absent de la table située à gauche de JOIN, « comptes_bancaires ».

```
mysql> 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;
+------+--------------+---------------+-----------+
|ID_Clt|Nom           |Prenom         |Solde_Total|
+------+--------------+---------------+-----------+
|     1|DUPONT        |JEAN           |    1200.50|
|     2|JACQUENOD     |JEAN-CHRISTOPHE|    -308.87|
|     3|MURCIAN       |CAROLE         |    3548.98|
|     4|LERY          |JEAN-MICHEL    |     -18.98|
|     5|DE-LA-RUE     |JEAN-CHRISTOPHE|     -27.44|
|     6|MARTIN        |PAUL-DAVID     |     206.21|
|     7|MARTIN        |PIERRE         |    1234.56|
|     8|JACQUENOD     |FREDERIC       |     432.98|
|     9|JACQUENOD     |LAURENCE       |    -203.18|
|    10|DUMOULIN      |JEAN-CHRISTOPHE|   -2186.86|
|    11|LABONNE-JAYAT |OLIVIER        |     -65.98|
|    12|DE-LA-FONTAINE|JEAN           |    1825.54|
|    13|LEVY          |SAMUEL         |     231.87|
|    14|DE-LA-RUE     |LAURENCE       |    2135.98|
|    15|DUPONT        |JEAN           |   12314.90|
|    16|MARTIN        |ALBERT         |     213.49|
|    17|ROUSSE        |JACQUES        |       NULL|
+------+--------------+---------------+-----------+
17 rows in set (0,00 sec)
```

Sauvegarde de la base de données

Cette partie présente la sauvegarde d'une base de données. Au niveau du shell, après avoir quitter le moniteur MySQL si vous y étiez, saisissez la commande suivante, puis le mot de passe lorsqu'il est demandé :

```
$ mysqldump -u root -p --opt CoursPHP >
sauvegarde_CoursPHP.sql
Enter password: xxxx
```

Cela produit le fichier `sauvegarde_CoursPHP.sql` dans le répertoire courant :

```
$ ls -l sauvegarde_CoursPHP.sql
-rw-rw-r-- 1 lery lery 2071 mars  30 16:54
sauvegarde_CoursPHP.sql
```

Ce fichier contient toutes les syntaxes SQL recréant les différentes tables de la base de données « CoursPHP ». Certaines lignes sont remplacées par des « … ».

```
$ cat sauvegarde_CoursPHP.sql
-- MySQL dump 10.13  Distrib 5.6.21, for Linux (x86_64)
-- Host: localhost    Database: CoursPHP
-- ------------------------------------------------------
-- Server version5.6.21
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
--
-- Table structure for table `personnes`
--
DROP TABLE IF EXISTS `personnes`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `personnes` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(255) NOT NULL,
  `Prenom` varchar(255) NOT NULL,
```

```
  `Age` int(11) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8
COMMENT='Table de personnes';
...
```

Restauration de la base de données

La base de données « CoursPHP » n'est pas sauvegardée elle-même par la syntaxe précédente. Si elle a été supprimée il faut préalablement la recréer avant d'effectuer la restauration. La restauration peut être obtenue par la commande suivante :

```
$ mysql -u root -h localhost CoursPHP -p <
sauvegarde_CoursPHP.sql
Enter password: xxxx
```

Ou bien sous le moniteur MySQL :

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
mysql> USE CoursPHP;
Database changed
mysql> SOURCE sauvegarde_CoursPHP.sql;
Query OK, 0 rows affected (0,00 sec)
...
```

Les requêtes préparées

Principe

Durant une session SQL, il est courant d'effectuer plusieurs fois la même requête avec différentes valeurs. Le langage SQL permet de préparer à l'avance une requête et de lui affecter un nom. Elle devient réutilisable, et s'exécute 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 généralement des variables de l'utilisateur qui sont présentées dans la section suivante.

Les variables utilisateurs

Les variables SQL sont toutes précédées par le caractère @. Elles contiennent des valeurs comme des entiers, des réels, des chaînes de caractères. Seuls les lettres, chiffres ainsi que les caractères souligné, dollar et point sont autorisés pour le nom des variables. En SQL, le nom des variables n'est pas sensible à la casse.

Création et modification

L'instruction « SET » crée une variable SQL si elle n'existe pas, ou la modifie sinon. La variable n'existe que durant la session SQL. La syntaxe suivante crée les deux variables `@Nom` et `@Prenom`. Le signe d'affectation est le caractère « = » :

```
mysql> SET @Nom='MARTIN',@Prenom='JEAN';
Query OK, 0 rows affected (0,00 sec)
```

Il est également possible de créer ou d'affecter les variables avec la syntaxe SELECT. Le signe d'affectation est le caractère « := » :

```
mysql> SELECT @Nom:='MARTIN',@Prenom:='JEAN';
+--------------+---------------+
|@Nom:='MARTIN'|@Prenom:='JEAN'|
+--------------+---------------+
|MARTIN        |JEAN           |
+--------------+---------------+
1 row in set (0,00 sec)
```

L'affichage

L'affichage d'une variable s'obtient avec la syntaxe SELECT :

```
mysql> SELECT @Nom;
+------+
|@Nom  |
+------+
|MARTIN|
+------+
1 row in set (0,00 sec)
```

L'utilisation

L'exemple suivant montre l'utilisation de la variable `@Nom` dans un SELECT sur la table « personnes ». Cette variable sert de filtre dans une clause WHERE :

```
mysql> SELECT ID,Nom,Prenom,Age FROM personnes WHERE Nom=@Nom;
+--+------+------------+---+
|ID|Nom   |Prenom      |Age|
+--+------+------------+---+
| 6|MARTIN|PIERRE-DAVID| 27|
| 7|MARTIN|PIERRE      | 56|
|16|MARTIN|ALBERT      | 25|
+--+------+------------+---+
3 rows in set (0,00 sec)
```

Création d'une requête préparée

L'exemple précédent affiche toutes les personnes ayant comme nom MARTIN. Pour obtenir la liste des DUPONT il suffit de saisir :

```
mysql> SET @Nom='DUPONT';
Query OK, 0 rows affected (0,00 sec)

mysql> SELECT ID,Nom,Prenom,Age FROM personnes WHERE Nom=@Nom;
+--+------+------+---+
|ID|Nom   |Prenom|Age|
+--+------+------+---+
| 1|DUPONT|JEAN  | 28|
|15|DUPONT|JEAN  | 54|
+--+------+------+---+
2 rows in set (0,00 sec)
```

Afin d'éviter de répéter cette syntaxe, on peut préparer cette requête et lui affecter un nom grâce à la syntaxe PREPARE :

```
mysql> PREPARE selection_nom FROM 'SELECT ID,Nom,Prenom,Age
FROM personnes WHERE Nom=?';
Query OK, 0 rows affected (0,01 sec)
Statement prepared
```

Voici plusieurs remarques concernant cette syntaxe :

- Le nom de la requête `selection_nom`, n'est pas entre apostrophes ;
- Le texte de la requête est lui entre apostrophes ;
- Une seule requête peut être indiquée dans une requête préparée ;
- Le paramètre à utiliser au moment de l'exécution est représenté par le caractère « ? ». Il peut y avoir plusieurs paramètres ;
- Les paramètres ne peuvent contenir que des données.

La syntaxe suivante montre un autre exemple avec deux paramètres :

```
mysql> PREPARE selection_nom_prenom FROM 'SELECT
ID,Nom,Prenom,Age FROM personnes WHERE Nom=? AND Prenom=?';
Query OK, 0 rows affected (0,00 sec)
Statement prepared
```

La syntaxe suivante utilise deux paramètres et la clause LIKE :

```
mysql> PREPARE selection_nom_like_prenom FROM 'SELECT
ID,Nom,Prenom,Age FROM personnes WHERE Nom=? AND Prenom LIKE
?';
Query OK, 0 rows affected (0,00 sec)
Statement prepared
```

Exécution d'une requête préparée

L'exécution d'une requête préparée utilise la syntaxe EXECUTE et USING pour passer les valeurs aux paramètres. Voici l'exécution de `selection_nom` :

```
mysql> EXECUTE selection_nom USING @Nom;
+--+------+------+---+
|ID|Nom   |Prenom|Age|
+--+------+------+---+
| 1|DUPONT|JEAN  | 28|
|15|DUPONT|JEAN  | 54|
+--+------+------+---+
2 rows in set (0,00 sec)
```

Voici un autre exemple d'utilisation de cette requête préparée :

```
mysql> SET @Nom='MARTIN';
Query OK, 0 rows affected (0,00 sec)

mysql> EXECUTE selection_nom USING @Nom;
+--+------+------------+---+
|ID|Nom   |Prenom      |Age|
+--+------+------------+---+
| 6|MARTIN|PIERRE-DAVID| 27|
| 7|MARTIN|PIERRE      | 56|
|16|MARTIN|ALBERT      | 25|
+--+------+------------+---+
3 rows in set (0,00 sec)
```

Voici l'exécution de `selection_nom_prenom` sur un nom et un prénom :

```
mysql> SET @Prenom='PIERRE';
Query OK, 0 rows affected (0,00 sec)
mysql> EXECUTE selection_nom_prenom USING @Nom, @Prenom;
+--+------+------+---+
|ID|Nom   |Prenom|Age|
+--+------+------+---+
| 7|MARTIN|PIERRE| 56|
+--+------+------+---+
1 row in set (0,00 sec)
```

Voici l'exécution de `selection_nom_like_prenom` sur un nom et un prénom :

```
mysql> SET @Prenom='%PIERRE%';
Query OK, 0 rows affected (0,00 sec)

mysql> EXECUTE selection_nom_like_prenom USING @Nom, @Prenom;
+--+------+------------+---+
|ID|Nom   |Prenom      |Age|
+--+------+------------+---+
| 6|MARTIN|PIERRE-DAVID| 27|
| 7|MARTIN|PIERRE      | 56|
+--+------+------------+---+
2 rows in set (0,00 sec)
```

Suppression d'une requête préparée

La syntaxe DEALLOCATE PREPARE supprime une requête préparée.

```
mysql> DEALLOCATE PREPARE selection_nom_prenom;
Query OK, 0 rows affected (0,00 sec)
```

Après cette syntaxe, la requête préparée n'existe plus :

```
mysql> EXECUTE selection_nom_prenom USING @Nom, @Prenom;
ERROR 1243 (HY000): Unknown prepared statement handler
(selection_nom_prenom) given to EXECUTE
```

Avantages

Lors du lancement d'une requête, la base de données effectue deux traitement : l'analyse (syntaxique) de la requête, puis son exécution. Si la même requête est lancée plusieurs fois, son analyse est effectuée à chaque fois.

Avec la requête préparée, cette analyse ne se fait qu'une seule fois, lors de sa préparation. Chaque lancement ultérieur ne fait que l'exécution, ce qui optimise le temps de traitement.

Le second avantage est d'ordre sécuritaire. Dans le cadre d'une session SQL durant laquelle l'utilisateur saisie lui même les données de la requête directement à la console, la sécurité est assurée par cet utilisateur qui contrôle la nature des données. Cela est tout à fait différent lorsque que l'accès à la base se fait via un programme PHP. Dans ce cas, l'utilisateur du site web peut très bien entrer, à la place de données attendues comme un nom, une requête SQL, et ainsi avoir un accès direct à la base de données et potentiellement récupérer des données sensibles comme des mots de passe. Ce type de piratage se nomme **injection SQL**.

Avec les requêtes préparées, les paramètres sont identifiés comme tels et ne peuvent pas être assimilés à des requêtes, ce qui protège contre l'injection SQL.

Le mode transactionnel

Problématique initiale

La gestion des données dans des tables de bases de données peut nécessiter plusieurs requêtes.

Si à chaque requête mettant à jour les données (UPDATE ou INSERT) la base reste « cohérente » alors cela ne pose aucun problème, même s'il y a une erreur. En effet, à chaque requête on peut détecter l'erreur de mise à jour et corriger le problème.

Par contre, si la mise à jour est plus « complexe », la cohérence des données peut être obtenue après plusieurs requêtes. Ainsi, si la première requête est correctement effectuée mais pas la deuxième, alors les données restes incohérentes. Dans ce cas il faut pouvoir détecter l'erreur et revenir en arrière de cet ensemble de requêtes, à l'état d'origine.

C'est le cas d'un virement bancaire entre les comptes de deux clients, qui est constitué de deux actions : Le débit du premier compte bancaire ; Le crédit du second compte bancaire. Il est impératif que ces actions soient effectuées toutes les deux pour que le virement soit effectif.

Si le débit du premier compte est effectué et que le crédit du second compte échoue, alors la somme a bien été débitée du compte initial mais n'a pas été créditée sur le compte final. Le solde cumulé de ces deux comptes n'est plus le même.

Il faut détecter l'erreur de la *transaction*, le virement, constituée par deux actions complémentaires, le débit et le crédit, et corriger l'erreur afin de conserver l'équilibre des deux comptes !

De plus, il est impératif que, pendant les modifications au sein d'une transaction, aucun autre utilisateur ne puisse venir s'intercaler et modifier les tables sur lesquelles travaille la transaction. Une transaction gère le verrouillage des tables accédées. Dès la première modification effectuée à l'intérieur d'une transaction, toute modification par un autre utilisateur sera bloquée. Ainsi, si un autre utilisateur tente de modifier un élément de la table, sa modification reste en attente, elle ne sera prise en compte qu'à la fin de la transaction en cours de traitement.

Un traitement « global » regroupant plusieurs requêtes est une transaction. Elle doit être vue, appliquée, ou annulée, comme si c'était une seule requête.

Il existe deux moyens de mettre en œuvre le mode transactionnel :

- via la désactivation de la validation automatique pendant la session de travail : autocommit ;
- via la création d'une transaction particulière et ponctuelle : START TRANSACTION;

Une caractéristique du moteur de stockage

Selon le moteur de stockage des données, le mode transactionnel sera disponible ou non. Ainsi le moteur Mylsam, performant et ayant un index FULL-TEXT, ne supporte ni les clefs étrangères, ni les transactions. Le moteur InnoDB, performant dans l'intégrité des données, gère les clefs étrangères et les transactions. C'est le moteur qui a été utilisé lors de la création des différentes tables.

Gestion de la validation automatique via *autocommit*

Principe

Par défaut le moteur InnoDB valide automatiquement toutes les transactions, et considère chaque requête comme une transaction. Cette validation automatique est définie par le mode « autocommit ».

Affichage de l'état

La syntaxe suivante affiche son état :

```
mysql> SELECT @@autocommit;
+-------------+
|@@autocommit|
+-------------+
|           1|
+-------------+
1 row in set (0,00 sec)
```

Quand ce mode est actif (valeur 1), chaque mise à jour de la table (INSERT, UPDATE, …), est réellement effectuée et écrite sur le disque.

Modification de l'état

Si on souhaite faire des modifications sur les tables et à tout moment pouvoir les valider ou revenir en arrière, il faut désactiver ce mode, comme suit :

```
mysql> SET autocommit = 0;
Query OK, 0 rows affected (0,00 sec)
```

Une fois « autocommit » désactivé, la session de travail possède en permanence une transaction ouverte, ce qui n'est pas sans conséquence. C'est-à-dire que, si on veut rendre permanentes les modifications, il faudra les valider via l'instruction COMMIT pour provoquer leur écriture sur disque. Si on veut les annuler, il faut utiliser l'instruction ROLLBACK pour les supprimer de la mémoire.

Les instructions COMMIT ou ROLLBACK terminent la transaction courante, et une nouvelle transaction est à nouveau ouverte, tant que le mode « autocommit » reste désactivé. La syntaxe suivante revient au mode de validation automatique.

```
mysql> SET autocommit = 1;
Query OK, 0 rows affected (0,00 sec)
```

Inconvénients

La gestion directe de « autocommit » pour effectuer des transactions pose un certain nombre de problèmes :

8. Tant que les modifications ne sont pas validées via COMMIT, rien n'est écrit sur disque dur, tout reste dans la mémoire de la session de travail. L'affichage « laisse croire » à l'utilisateur que les traitements sont effectués, mais ce n'est que la vision de la mémoire de *sa* session de travail. En effet, si un autre utilisa-

teur se connecte en même temps et consulte la table, il ne verra que les informations stockées sur disque, donc aucune des modifications du premier utilisateur !

9. Dès la première modification à l'intérieur d'une transaction, le verrouillage des tables accédées est effectif. Dans le cas présent, la désactivation de « autocommit » implique qu'une transaction est ouverte en permanence. Ainsi les autres utilisateurs peuvent se retrouver bloqués très longtemps, empêchant le fonctionnement « normal » concurrentiel de la base de données.

Ce mode de gestion des transactions via « autocommit » n'est pas recommandé, pour les raisons évoquées précédemment. Il est préférable d'ouvrir une transaction quand on en a besoin, avec START TRANSACTION. Elle se terminera définitivement avec COMMIT ou ROLLBACK.

Exemples d'utilisation

Le premier exemple présente le fonctionnement du mode « autocommit », et le fait qu'une transaction reste ouverte en permanence. Le second exemple montre que tant que le mode « autocommit » est désactivé, les modifications ne sont pas écrites sur disque, les autres utilisateurs ne les voient pas.

Exemple de fonctionnement

Cet exemple présente « autocommit », et l'ouverture permanente d'une transaction. On se connecte et on sélectionne la base de données « CoursPHP »

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> USE CoursPHP;
Reading table information for completion of table and ...
Database changed
```

On affiche l'état des comptes N°1 et N°7, de la table « comptes_bancaires »

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+-----+--------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|3185.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On désactive le mode « autocommit »

```
mysql> SET autocommit = 0;
Query OK, 0 rows affected (0,00 sec)
```

Dorénavant, il faut valider ou annuler explicitement les requêtes pour les écrire ou non sur le disque. On débite 200 € du compte N°7, et on crédite 200 € au compte N°1 :

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-200 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> UPDATE comptes_bancaires SET Solde=Solde+200 WHERE
Id_Cpt=1;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

L'état des deux comptes montre que les modifications sont gardées en mémoire.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+------------+------+-------+
|ID_Cpt|Agence|Numero |Type        |ID_Clt|Solde  |
+------+------+-------+------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+------------+------+-------+
2 rows in set (0,00 sec)
```

On annule les traitements précédents.

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0,01 sec)
```

L'état des deux comptes montre que rien n'a été pris en compte, les valeurs d'origines sont affichées :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+------------+------+-------+
|ID_Cpt|Agence|Numero |Type        |ID_Clt|Solde  |
+------+------+-------+------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|3185.08|
+------+------+--------+----------+------+-------+
2 rows in set (0,00 sec)
```

À nouveau, on débite 200 € du compte N°7, et on crédite 200 € au compte N°1 :

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-200 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE comptes_bancaires SET Solde=Solde+200 WHERE
Id_Cpt=1;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

On affiche l'état des deux comptes, les modifications sont mémorisées.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On valide les traitements précédents.

```
mysql> COMMIT;
Query OK, 0 rows affected (0,01 sec)
```

L'affichage des deux comptes montre les mêmes données :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

Une annulation puis un affichage montre que rien ne change, la validation précédente a écrit les modifications sur le disque.

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0,00 sec)
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
```

```
2 rows in set (0,00 sec)
```

Une nouvelle fois, on débite 200 € du compte N°7 :

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-200 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

On affiche les deux comptes :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2785.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On annule les traitements:

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0,00 sec)
```

On affiche les deux comptes. Le débit précédent du compte N°7 est bien annulé, ce qui démontre qu'une transaction est ouverte en permanence tant que « auto-commit » est désactivée !

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On réactive le mode « autocommit ». Désormais, chaque modification est écrite sur disque.

```
mysql> SET autocommit = 1;
Query OK, 0 rows affected (0,00 sec)
```

Impact pour les autres utilisateurs

Les syntaxes suivantes montrent que tant que le mode « autocommit » est désactivé, et qu'aucune instruction COMMIT n'est exécutée, les modifications ne sont pas écrites sur disque, les autres utilisateurs ne les voient pas. Pour présenter l'impact sur les autres utilisateurs nous utilisons deux comptes en parallèle, l'administrateur « root » et le compte « clientsconsult ».

On se connecte comme **administrateur** et on sélectionne la base de données « CoursPHP » :

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> USE CoursPHP;
Reading table information for completion of table and ...
Database changed
```

On désactive le mode « autocommit »

```
mysql> SET autocommit = 0;
Query OK, 0 rows affected (0,00 sec)
```

On affiche l'état des comptes bancaires N°1 et N°7.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2785.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On débite 100 € du compte N°7, et on crédite 100 € au compte N°1 :

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-100 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> UPDATE comptes_bancaires SET Solde=Solde+100 WHERE
Id_Cpt=1;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

On affiche l'état des comptes bancaires N°1 et N°7. On voit ces deux modifications, qui ne sont faites qu'en mémoire de la session de l'administrateur.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
```

```
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 850.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2685.08|
+------+------+-------+-------------+------+-------+
```

On ouvre en parallèle une autre session avec le compte **clientsconsult** et on sélectionne la base de données « CoursPHP ».

```
$ mysql --no-defaults -u clientsconsult -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> USE CoursPHP;
Reading table information for completion of table and ...
Database changed
```

On affiche l'état des comptes bancaires N°1 et N°7. On ne voit aucune des modifications effectuées par l'administrateur car elles ne sont pas écrites sur le disque.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2785.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On revient à la session ouverte par **l'administrateur**, et on exécute l'instruction COMMIT :

```
mysql> COMMIT;
Query OK, 0 rows affected (0,01 sec)
```

Les modifications des comptes bancaires N°1 et N°7 sont écrites sur disque :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 850.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2685.08|
+------+------+-------+-------------+------+-------+
```

```
2 rows in set (0,00 sec)
```

On revient à la session ouverte par **clientsconsult**, et on affiche à nouveau l'état des comptes bancaires N°1 et N°7. On voit les modifications de l'administrateur.

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 850.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2685.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

Sur la session de **l'administrateur** on remet la validation automatique, et on ferme la session.

```
mysql> SET autocommit = 1;
Query OK, 0 rows affected (0,00 sec)

mysql> QUIT;
Bye
```

Sur la session de **clientsconsult** on ferme la session.

```
mysql> QUIT;
Bye
```

Utilisation d'une transaction spécifique avec *START TRANSACTION*

Principe

L'utilisation d'une transaction spécifique, évite de maintenir une transaction ouverte en permanence durant toute la session de travail, ce qui est préjudiciable pour les autres utilisateurs. Le processus d'écriture sur disque des données avec une transaction spécifique, est le suivant :

- **Avant la transaction** la **validation est automatique** : chaque modification est écrite sur disque.
- **Pendant la transaction** la validation n'est plus automatique : L'instruction COMMIT effectue la validation de l'ensemble des instructions, et provoque l'écriture simultanée des modifications. L'instruction ROLLBACK annule les modifications et ne provoque aucune écriture ;
- **Après la transaction** la **validation redevient automatique**.

Il est recommandé que chaque requête opérant des modifications les applique réellement sur disque. Il faut que le mode « autocommit » reste activé par défaut ! Il

faut utiliser une transaction spécifique quand l'écriture simultanée sur disque d'un ensemble de modifications est nécessaire.

Les requêtes :

START TRANSACTION

La syntaxe suivante démarre une nouvelle transaction :

```
START TRANSACTION;
```

- Elle désactive « autocommit » jusqu'à la saisie du COMMIT ou ROLLBACK ;
- Elle active le mécanisme de verrouillage « LOCK » de la table, empêchant toute modification par une autre personne, dès la première modification à l'intérieur de la transaction.

Après cette syntaxe on saisit la série de requête qui modifie la table (UPDATE, INSERT, …). Cette syntaxe possède deux alias : BEGIN et BEGIN WORK.

COMMIT

La syntaxe COMMIT termine la transaction en validant les traitements. Les modifications sont écrites sur disque. Le verrouillage de la table est levé après la validation. Sa syntaxe est :

```
COMMIT;
```

ROLLBACK

La syntaxe ROLLBACK termine la transaction en annulant les traitements. Aucune modification n'est écrite sur disque. Le verrouillage de la table est levé après l'annulation. Sa syntaxe est :

```
ROLLBACK;
```

Si une erreur est détectée sur une des requêtes de la transaction, on peut, via ROLLBACK, revenir en arrière à l'état initial avant le début de la transaction.

Rappel

Chaque modification est visible durant la transaction pour l'utilisateur qui l'effectue mais elle est invisible pour les autres tant qu'elle elle n'est pas écrite sur le disque.

Exemples

Exemple d'annulation

On connecte l'**administrateur** et on sélectionne la base de données « CoursPHP »

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> USE CoursPHP;
Reading table information for completion of table and ...
```

```
Database changed
```

Nous souhaitons faire un virement de 200 € du compte N°7 vers le compte N°1. Voici l'état de ces deux comptes :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|3185.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On démarre une transaction :

```
mysql> START TRANSACTION;
```

On retire 200 € du compte N°7. La modification est effectuée en mémoire.

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-200 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On ajoute 200 € du compte N°100, au lieu du compte N°1. Comme le compte 100 n'existe pas, aucune modification n'est effectuée. La requête donne un résultat et l'affichage montre la modification.

```
mysql> UPDATE comptes_bancaires SET Solde=Solde+200 WHERE
Id_Cpt=100;
Query OK, 0 rows affected (0,00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
```

```
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On annule toutes les modifications précédentes. L'état initial est restauré :

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0,00 sec)
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|3185.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
mysql> QUIT;
Bye
```

Exemple de validation

L'exemple suivant reprend les mêmes opérations, mais sans l'erreur de compte (100 au lieu de 1). C'est le bon compte qui est crédité. On se connecte comme **administrateur** et on sélectionne la base de données « CoursPHP »

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> USE CoursPHP;
Reading table information for completion of table and ...
Database changed
```

Nous voulons faire un virement de 200 € du compte N°7 vers le compte N°1. Voici l'état de ces deux comptes :

```
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|3185.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On démarre une transaction :

```
mysql> START TRANSACTION;
```

On retire 200 € du compte N°7. La modification est effectuée en mémoire.

```
mysql> UPDATE comptes_bancaires SET Solde=Solde-200 WHERE
Id_Cpt=7;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 550.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On ajoute 200 € au compte N°1. La modification est effectuée.

```
mysql> UPDATE comptes_bancaires SET Solde=Solde+200 WHERE
Id_Cpt=1;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
2 rows in set (0,00 sec)
```

On finalise la transaction. Les données sont écrites sur disque :

```
mysql> COMMIT;
Query OK, 0 rows affected (0,00 sec)
mysql> SELECT ID_Cpt,Agence,Numero,Type,ID_Clt,Solde FROM
comptes_bancaires WHERE Type="Compte_Dépôts" AND (ID_Cpt=1 OR
ID_Cpt=7);
+------+------+-------+-------------+------+-------+
|ID_Cpt|Agence|Numero |Type         |ID_Clt|Solde  |
+------+------+-------+-------------+------+-------+
|     1| 00602|165143P|Compte_Dépôts|     1| 750.98|
|     7| 00602|154123P|Compte_Dépôts|     3|2985.08|
+------+------+-------+-------------+------+-------+
```

```
2 rows in set (0,00 sec)
```

La gestion des utilisateurs

Cette section présente la **gestion des utilisateurs** en langage SQL.

Principe

MySQL gére les utilisateurs via une **identification** de la forme : **dupont@ordinateur.fr**. Le fonctionnement de cet identifiant, ainsi que les privilèges qui lui sont associés ont été présentés au chapitre 10 dans la section phpMyAdmin.

Affichage des utilisateurs existants

La table mysql.user

Pour gérer les utilisateurs il faut se connecter en tant que « root ».

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
```

Les utilisateurs sont gérés dans la table « mysql.user ». La liste est obtenue par la syntaxe suivante :

```
mysql> SELECT User, Host, Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                           |
|    |localhost|                                           |
|    |linux    |                                           |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-------------------------------------------+
5 rows in set (0,00 sec)
```

L'utilisateur anonyme

L'affichage précédent montre deux lignes dont la colonne « User » est vide, et la colonne « Host » indique « localhost » et « linux ». C'est l'utilisateur « anonyme » qui est créé dès l'installation de MySQL, en même temps que la base « test ». N'importe quel utilisateur qui n'est pas enregistré (aucun login) et qui n'a donc pas de mot de passe, peut se connecter à la base « test » et à toutes les bases dont le nom commence par « test ». En voici un exemple.

On se connecte sans login ni mot de passe (connexion anonyme) :

```
$ mysql --no-defaults -h localhost
Welcome to the MySQL monitor.  Commands end with ; or \g...
```

On tente d'accéder à la base « CoursPHP », l'accès est refusé.

```
mysql> USE CoursPHP;
ERROR 1044 (42000): Access denied for user ''@'localhost' to
database 'CoursPHP'
```

On tente d'accéder à la base « test », l'accès est accepté.

```
mysql> USE test;
Reading table information for completion of table and ...
Database changed
```

On voit toutes les tables de la base « test » :

```
mysql> SHOW TABLES;
+--------------+
|Tables_in_test|
+--------------+
|materiel      |
+--------------+
1 row in set (0,01 sec)
```

On voit tous les enregistrements de la table « materiel » :

```
mysql> SELECT * FROM materiel ;
+--+-------+-----+
|ID|Libelle|Prix |
+--+-------+-----+
| 1|pelle  | 12.5|
| 2|marteau|10.26|
+--+-------+-----+
2 rows in set (0,01 sec)
mysql> QUIT;
Bye
```

Création d'un compte utilisateur *CREATE USER*

Il existe deux méthodes pour créer des utilisateurs sous MySQL et leur affecter des privilèges :

1. Utiliser les instructions de gestion des utilisateurs telles que CREATE USER et GRANT ;
2. Manipuler directement les tables de privilèges avec les instructions INSERT, UPDATE, DELETE ;

IL est préférable d'utiliser les instructions dédiées à la gestion des utilisateurs, car elles effectuent des contrôles de cohérence lors de la création, ce qui n'est pas le cas avec l'accès direct aux tables ce qui reste très dangereux.

Dans cet exemple nous créons deux nouveaux comptes d'utilisateur :

- personnesadm@% : pour un accès depuis n'importe quel poste de connexion ;
- personnesadm@local : pour un accès local (depuis le serveur MySQL) ;

Le mot de passe est défini en même temps via la syntaxe « IDENTIFIED BY ». Les « xxxx » doivent être remplacées par le mot de passe réel.

```
mysql> CREATE USER personnesadm@'%' IDENTIFIED BY 'xxxx' ;
Query OK, 0 rows affected (0,27 sec)
mysql> CREATE USER personnesadm@localhost IDENTIFIED BY 'xxxx'
;
Query OK, 0 rows affected (0,00 sec)
```

Le mot de passe aurait pu être affecté ou modifié séparément. Ainsi la syntaxe de la création du compte « personnesadm@localhost » peut également s'écrire :

```
mysql> CREATE USER personnesadm@localhost ;
mysql> SET PASSWORD FOR personnesadm@localhost =
PASSWORD('xxx') ;
```

La syntaxe suivante vérifie que les deux comptes ont été créés :

```
mysql> SELECT User, Host, Password FROM mysql.user;
+------------+---------+-------------------------------------+
|User        |Host     |Password                             |
+------------+---------+-------------------------------------+
|root        |localhost|*9C4FE4A10F01988F50D685C3F9515570...|
|root        |linux    |                                     |
|            |localhost|                                     |
|            |linux    |                                     |
|pma         |localhost|*AC4D94A19F01998F50D68AC3F951A862...|
|personnesadm|localhost|*70828A978420F0614DEBA7174BF38083...|
|personnesadm|%        |*70828A978420F0614DEBA7174BF38083...|
+------------+---------+-------------------------------------+
7 rows in set (0,00 sec)
```

Gestion des privilèges

Affichage des privilèges *SHOW GRANTS*

La syntaxe suivante affiche les privilèges de l'utilisateur « pma@locahost » :

```
mysql> SHOW GRANTS FOR pma@localhost;
+-----------------------------------------------------------+
|Grants for pma@localhost                                   |
```

```
+-----------------------------------------------------------+
|GRANT USAGE ON *.* TO 'pma'@'localhost' IDENTIFIED BY ...  |
|GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE ON `phpmy ...|
|GRANT SELECT (Host, Create_priv, Shutdown_priv, Delete_p...|
|GRANT SELECT (Table_priv, Column_priv, Table_name, Db, ... |
|GRANT SELECT ON `mysql`.`host` TO 'pma'@'localhost'        |
|GRANT SELECT ON `mysql`.`db` TO 'pma'@'localhost'          |
+-----------------------------------------------------------+
6 rows in set (0,00 sec)
```

Ajout de privilèges *GRANT*

L'ajout de privilèges utilise la syntaxe SQL, GRANT.

Pour le compte personnesadm@%
La syntaxe suivante affecte le privilège de **consulter** la table « personnes » (SELECT) pour cet utilisateur depuis n'importe quel poste de travail.

```
mysql> GRANT SELECT ON CoursPHP.personnes TO personnesadm@'%';
Query OK, 0 rows affected (0,00 sec)
```

La syntaxe suivante vérifie les privilèges du compte de cet utilisateur :

```
mysql> SHOW GRANTS FOR personnesadm@'%';
+-----------------------------------------------------------+
|Grants for personnesadm@%                                  |
+-----------------------------------------------------------+
|GRANT USAGE ON *.* TO 'personnesadm'@'%' IDENTIFIED BY ...  |
|GRANT SELECT ON `CoursPHP`.`personnes` TO 'personnesadm'@'%'|
+-----------------------------------------------------------+
2 rows in set (0,00 sec)
```

Pour le compte personnesadm@localhost
Nous affectons le privilège de **modifier et supprimer** la table « personnes » pour cet utilisateur depuis le serveur.

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON CoursPHP.personnes
TO personnesadm@localhost;
Query OK, 0 rows affected (0,00 sec)
```

La syntaxe suivante affiche les privilèges de cet utilisateur :

```
mysql> SHOW GRANTS FOR personnesadm@localhost;
+-----------------------------------------------------------+
|Grants for personnesadm@localhost                          |
+-----------------------------------------------------------+
|GRANT USAGE ON *.* TO 'personnesadm'@'localhost' IDENTIFI...|
|GRANT SELECT, INSERT, UPDATE, DELETE ON `CoursPHP`.`perso...|
+-----------------------------------------------------------+
2 rows in set (0,00 sec)
```

Variations syntaxiques

Si on désire affecter le privilège SELECT sur toutes les tables de toutes les bases de données au compte « personnesadm@localhost », il faut saisir la syntaxe :

```
mysql> GRANT SELECT ON *.* FROM personnesadm@localhost;
```

On peut également appliquer cette syntaxe à plusieurs comptes :

```
mysql> GRANT SELECT ON *.* FROM personnesadm@localhost,
personnes@'%';
```

Si on désire affecter tous les privilèges sur la table « personnes », il faut saisir la syntaxe :

```
mysql> GRANT ALL PRIVILEGES ON CoursPHP.personnes FROM
personnesadm@localhost;
```

Enfin, il est possible de créer, d'affecter les privilèges et un mot de passe en une seule syntaxe. Ainsi pour le compte d'utilisateur « personnesadm@localhost », pour le créer et lui affecter les privilèges SELECT, INSERT, UPDATE, DELETE il suffit de saisir :

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON CoursPHP.personnes
TO personnesadm@localhost IDENTIFIED BY 'xxxx';
Query OK, 0 rows affected (0,00 sec)
```

Retrait de privilèges *REVOKE*

Sur une table particulière

Le retrait de privilèges correspond à la syntaxe SQL, REVOKE. La syntaxe suivante retire le privilège « DELETE » à « **personnesadm@localhost** » sur la table « personnes ».

```
mysql> REVOKE DELETE ON CoursPHP.personnes FROM
personnesadm@localhost;
Query OK, 0 rows affected (0,00 sec)
```

Le privilège DELETE a disparu de la liste pour « personnesadm@localhost ».

```
mysql> SHOW GRANTS FOR personnesadm@localhost;
+-----------------------------------------------------------+
|Grants for personnesadm@localhost                          |
+-----------------------------------------------------------+
|GRANT USAGE ON *.* TO 'personnesadm'@'localhost' IDENTIFI...|
|GRANT SELECT, INSERT, UPDATE ON `CoursPHP`.`personnes` TO...|
+-----------------------------------------------------------+
2 rows in set (0,00 sec)
```

Variations syntaxiques

Si on désire supprimer DELETE sur toutes les tables de toutes les bases de données, il faut saisir la syntaxe :

```
mysql> REVOKE DELETE ON *.* FROM personnesadm@localhost;
```

Si on désire supprimer tous les privilèges sur toutes les tables de toutes les bases de données, il faut saisir la syntaxe :

```
mysql> REVOKE ALL PRIVILEGES ON *.* FROM
personnesadm@localhost;
```

On peut également appliquer cette syntaxe à plusieurs comptes : :

```
mysql> REVOKE ALL PRIVILEGES ON *.* FROM
personnesadm@localhost, personnesadm@'%';
```

Gestion des paramètres de connexion

Comme cela à été présenté au chapitre 10 dans la section phpMyAdmin, chaque utilisateur possède des paramètres de connexion comme le **nombre maximal de connexions autorisées**. Ces paramètres sont définis dans la table « mysql.user ».

Problématique

La problématique et les conséquences de la mise en œuvre d'une limitation ou non des paramètres de connexion a été abordée avec phpMyAdmin.

Affichage des paramètres

La syntaxe suivante affiche les paramètres de connexion :

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

```
mysql> SELECT
max_questions,max_updates,max_connections,max_user_connections
FROM mysql.user WHERE USER='clientsconsult';
+-------------+-----------+---------------+------------------+
|max_questions|max_updates|max_connections|max_user_connec...|
+-------------+-----------+---------------+------------------+
|            0|          0|              0|                 0|
+-------------+-----------+---------------+------------------+
1 row in set (0,00 sec)
```

Modification des paramètres

La syntaxe suivante modifie les paramètres globaux de connexion pour l'utilisateur « clientsconsult » avec comme valeur :

- max_questions (MAX_QUERIES_PER_HOUR) = 20 ;
- max_updates (MAX_UPDATES_PER_HOUR) = 10 ;
- max_connections (MAX_CONNECTIONS_PER_HOUR) = 5 ;
- max_user_connections (MAX_USER_CONNECTIONS) = 15.

```
mysql> GRANT ALL ON *.* TO 'clientsconsult'@'%' WITH
MAX_QUERIES_PER_HOUR 20 MAX_UPDATES_PER_HOUR 10
MAX_CONNECTIONS_PER_HOUR 5 MAX_USER_CONNECTIONS 15;
Query OK, 0 rows affected (0,00 sec)
```

L'affichage confirme la modification :

```
mysql> SELECT
max_questions,max_updates,max_connections,max_user_connections
FROM mysql.user WHERE USER='clientsconsult';
+-------------+-----------+---------------+------------------+
|max_questions|max_updates|max_connections|max_user_connec...|
+-------------+-----------+---------------+------------------+
|           20|         10|              5|                15|
+-------------+-----------+---------------+------------------+
1 row in set (0,00 sec)
```

La syntaxe suivante limite les connexions à la base de données « CoursPHP » lors de la création du compte « clientsconsult » via la syntaxe GRANT.

```
mysql> GRANT ALL ON CoursPHP.* TO 'clientsconsult'@'%' WITH
MAX_QUERIES_PER_HOUR 20 MAX_UPDATES_PER_HOUR 10
MAX_CONNECTIONS_PER_HOUR 5 MAX_USER_CONNECTIONS 15;
```

La suppression de la limitation revient à affecter ces paramètres avec la valeur 0 qui indique qu'il n'y a aucune limite définie. La syntaxe devient :

```
mysql> GRANT ALL ON *.* TO 'clientsconsult'@'%' WITH
MAX_QUERIES_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

Remarque

La limitation du nombre de connexions dépend du paramètre MAX_USER_CONNECTIONS de l'utilisateur mais aussi de la valeur de la variable système max_user_connections.

Renommer un compte utilisateur *RENAME USER*

Il est possible de renommer un compte utilisateur via la syntaxe RENAME USER. La première syntaxe montre la liste des utilisateurs :

```
mysql> SELECT User, Host, Password FROM mysql.user;
+------------+---------+-------------------------------------+
|User        |Host     |Password                             |
+------------+---------+-------------------------------------+
|root        |localhost|*9C4FE4A10F01988F50D685C3F95155705...|
|root        |linux    |                                     |
|            |localhost|                                     |
|            |linux    |                                     |
|pma         |localhost|*AC4D94A19F01998F50D68AC3F951A8625...|
|personnesadm|localhost|*70828A978420F0614DEBA7174BF380835...|
|personnesadm|%        |*70828A978420F0614DEBA7174BF380835...|
+------------+---------+-------------------------------------+
7 rows in set (0,00 sec)
```

La syntaxe suivante renomme le compte personnesadm@'%' en personnesconsult@'%':

```
mysql> RENAME USER personnesadm@'%' TO personnesconsult@'%';
Query OK, 0 rows affected (0,00 sec)
```

Le compte a bien été renommé :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----------------+---------+-----------------------------------+
|User            |Host     |Password                           |
+----------------+---------+-----------------------------------+
|root            |localhost|*9C4FE4A10F01988F50D685C3F9515...|
|root            |linux    |                                   |
|                |localhost|                                   |
|                |linux    |                                   |
|pma             |localhost|*AC4D94A19F01998F50D68AC3F951A...|
|personnesadm    |localhost|*70828A978420F0614DEBA7174BF38...|
|personnesconsult|%        |*70828A978420F0614DEBA7174BF38...|
+----------------+---------+-----------------------------------+
7 rows in set (0,00 sec)
```

Suppression d'un compte utilisateur DROP USER

La syntaxe pour supprimer un compte d'utilisateur est DROP USER. Voici comment supprimer les deux comptes créés précédemment. La première syntaxe montre la liste des utilisateurs :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----------------+---------+-----------------------------------+
|User            |Host     |Password                           |
+----------------+---------+-----------------------------------+
|root            |localhost|*9C4FE4A10F01988F50D685C3F9515...|
|root            |linux    |                                   |
|                |localhost|                                   |
```

```
|                |linux    |                                   |
|pma             |localhost|*AC4D94A19F01998F50D68AC3F951A...|
|personnesadm    |localhost|*70828A978420F0614DEBA7174BF38...|
|personnesconsult|%        |*70828A978420F0614DEBA7174BF38...|
+----------------+---------+-----------------------------------+
7 rows in set (0,00 sec)
```

Suppression du compte « personnesadm@localhost » :

```
mysql> DROP USER personnesadm@localhost;
Query OK, 0 rows affected (0,00 sec)
```

Suppression du compte « personnesconsult@% » :

```
mysql> DROP USER personnesconsult@'%';
Query OK, 0 rows affected (0,00 sec)
```

L'affichage de la liste des utilisateurs confirme la suppression :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                           |
|    |localhost|                                           |
|    |linux    |                                           |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-------------------------------------------+
5 rows in set (0,00 sec)
```

10-1.3 SECURISATION DE MYSQL

Sécurisation des comptes

Dans cette partie nous présentons les points de vigilance et les préconisations pour sécuriser les accès au serveur MySQL.

Le compte root

Mot de passe

À l'installation de MySQL, un identifiant « root » est créé pour administrer le serveur. Par défaut il ne possède aucun mot de passe. **Il est impératif d'affecter un mot de passe sur le compte « root ».** Cet administrateur possède deux comptes, selon le type d'accès.

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                           |
|    |localhost|                                           |
|    |linux    |                                           |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-------------------------------------------+
5 rows in set (0,00 sec)
```

Il faut modifier le mot de passe du compte de « localhost », mais aussi des autres comptes ayant une adresse IP d'un poste distant (ce qui n'est pas le cas de linux dans l'exemple précédent). Cela peut se faire via phpMyAdmin, ou bien via la syntaxe SQL :

```
mysql> SET PASSWORD FOR
root@localhost=PASSWORD('nouveau_motdepasse');
```

Accès à distance

Pour ce compte, il faut vérifier que l'accès à partir de n'importe quel poste de connexion n'est pas ouvert. Cela se fait en tentant de se connecter en tant que « root » à partir d'une poste distant.

Dans l'exemple suivant l'accès distant est refusé. Le serveur MySQL possède l'adresse IP 10.211.55.2. La syntaxe suivante tente d'établir la connexion, en tant que « root », vers ce serveur à partir d'un poste distant 10.211.55.2 et échoue :

```
$ mysql --no-defaults -u root -h 10.211.55.16 -p
Enter password: xxxx
ERROR 1130 (HY000): Host '10.211.55.2' is not allowed to
connect to this MySQL server
```

Si le compte « root » est ouvert en accès à partir de l'extérieur, il faut vérifier que cela est limité au **seul poste de travail habituel et personnel de la personne qui administre le serveur MySQL**. Dans notre exemple il s'agit de l'ordinateur « pdtadm.cnam.fr ».

```
mysql> SELECT User,Host,Password FROM mysql.user;
```

```
+----+--------------+---------------------------------------+
|User|Host          |Password                               |
+----+--------------+---------------------------------------+
|root|localhost     |*9C4FE4A10F01988F50D685C3F9515570588...|
|root|linux         |                                       |
|root|pdtadm.cnam.fr|*9C4FE4A10F01988F50D685C3F9515570588...|
|    |localhost     |                                       |
|    |linux         |                                       |
|pma |localhost     |*AC4D94A19F01998F50D68AC3F951A862588...|
|root|localhost     |*9C4FE4A10F01988F50D685C3F9515570588...|
+----+--------------+---------------------------------------+
7 rows in set (0,01 sec)
```

Le compte anonyme

L'administrateur devra se poser la question du bien fondé de laisser l'accès à son serveur via un compte anonyme sans mot de passe. Que n'importe qui puisse accéder au serveur MySQL et ait tous les droits sur une base « test » peut paraître « anormal ». Si tel est le cas, il faudra supprimer les comptes anonymes.

On affiche la liste des comptes :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                           |
|    |localhost|                                           |
|    |linux    |                                           |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-------------------------------------------+
5 rows in set (0,01 sec)
```

On supprime les deux comptes anonymes (sans login) :

```
mysql> DROP USER ''@localhost;
Query OK, 0 rows affected (0,00 sec)
mysql> DROP USER ''@linux;
Query OK, 0 rows affected (0,00 sec)
```

Les comptes anonymes sont bien supprimés :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                           |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
```

```
+----+---------+-----------------------------------------+
3 rows in set (0,00 sec)
```

Si un compte anonyme existe pour une connexion depuis n'importe quel poste (le caractère '%' apparaît dans la colonne « Host »), sa suppression se note :

```
mysql> DROP USER ''@'%';
Query OK, 0 rows affected (0,00 sec)
```

La base de test

De la même manière, l'administrateur devra se poser la question du bien fondé de conserver une base « test », ou de toute base dont le nom commence par « test_ ». Pour supprimer ces bases de données il suffit de saisir :

```
mysql> DROP DATABASE test;
mysql> DELETE FROM mysql.db WHERE Db='test' OR Db='test\_%';
```

Script de sécurisation

MySQL propose un script shell de sécurisation `mysql_secure_installation`, qui effectue les tâches précédentes. Sous Linux, pour l'exécuter il faut saisir :

```
$ /opt/lampp/bin/mysql_secure_installation
```

Puis il suffit de répondre aux questions posées. On affiche l'état des comptes et des bases de données du serveur avant de lancer le script de sécurisation :

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-----------------------------------------+
|User|Host     |Password                                 |
+----+---------+-----------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
|root|linux    |                                         |
|    |localhost|                                         |
|    |linux    |                                         |
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-----------------------------------------+
5 rows in set (0,00 sec)
mysql> SHOW databases;
+------------------+
|Database          |
+------------------+
|information_schema|
|CoursPHP          |
|cdcol             |
```

```
|mysql              |
|performance_schema|
|phpmyadmin         |
|test               |
+-------------------+
7 rows in set (0,00 sec)
mysql> QUIT;
Bye
```

On exécute le script `mysql_secure_installation`.

```
$ /opt/lampp/bin/mysql_secure_installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL
MySQL SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP
CAREFULLY!
In order to log into MySQL to secure it, we'll need the
current password for the root user.  If you've just installed
MySQL, and you haven't set the root password yet, the password
will be blank, so you should just press enter here.
Enter current password for root (enter for none): xxxx
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the
MySQL root user without the proper authorisation.
You already have a root password set, so you can safely answer
'n'.
Change the root password? [Y/n] Y
New password: xxxx
Re-enter new password: xxxx
Password updated successfully!
Reloading privilege tables..... Success!
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have a
user account created for them.  This is intended only for
testing, and to make the installation
go a bit smoother.  You should remove them before moving into
a production environment.
Remove anonymous users? [Y/n] Y
 ... Success!
Normally, root should only be allowed to connect from
'localhost'.  This ensures that someone cannot guess at the
root password from the network.
Disallow root login remotely? [Y/n] Y
 ... Success!
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing, and
should be removed before moving into a production environment.
Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
```

```
 - Removing privileges on test database...
 ... Success!
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.
Reload privilege tables now? [Y/n] Y
 ... Success!
All done!  If you've completed all of the above steps, your
MySQL installation should now be secure. Thanks for using
MySQL!
Cleaning up...
```

Attention de bien vérifier qu'il n'y a aucune erreur ! Il se peut que lors de la suppression de la base « test », l'erreur suivante se produise.

```
Remove test database and access to it? [Y/n] Y
 - Dropping test database...
ERROR 1010 (HY000) at line 1: Error dropping database (can't
rmdir './test/', errno: 17)
 ... Failed!  Not critical, keep moving...
 - Removing privileges on test database...
 ... Success!
```

Cela provient du fait que la tentative de suppression du répertoire « test », via la commande « rmdir ./test/ », échoue. Ce répertoire contient les données de la base « test » et doit être totalement vide, ce qui n'est pas le cas. En fait il contient encore un fichier (vide) dont le nom est « NOTEMPTY » :

```
$ sudo ls -al /opt/lampp/var/mysql/test/NOTEMPTY
-rw-r--r-- 1 mysql mysql 0 juin 26 2013
/opt/lampp/var/mysql/test/NOTEMPTY
```

Il suffit supprimer ce fichier :

```
$ sudo rm /opt/lampp/var/mysql/test/NOTEMPTY
```

Puis relancer l'exécution du script `mysql_secure_installation`, pour supprimer l'erreur. L'affichage des comptes utilisateurs et des bases de données confirme la sécurisation :

```
$ mysql --no-defaults -u root -h localhost -p
Enter password: xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g...
```

Les comptes anonymes ou n'ayant aucun mot de passe ont disparus :

```
mysql> SELECT User,Host,Password FROM mysql.user;
+----+---------+-------------------------------------------+
|User|Host     |Password                                   |
+----+---------+-------------------------------------------+
|root|localhost|*9C4FE4A10F01988F50D685C3F9515570588FEFDF|
```

```
|pma |localhost|*AC4D94A19F01998F50D68AC3F951A862588AEFA8|
+----+---------+-----------------------------------------+
2 rows in set (0,01 sec)
```

La base « test » a disparue :

```
mysql> SHOW databases;
+------------------+
|Database          |
+------------------+
|information_schema|
|CoursPHP          |
|cdcol             |
|mysql             |
|performance_schema|
|phpmyadmin        |
+------------------+
6 rows in set (0,00 sec)
mysql> QUIT;
Bye
```

Sécurisation réseau

Si votre serveur MySQL est accédé uniquement par votre site web, alors il faut sécuriser son accès réseau, du reste du monde. Il faut **restreindre l'accès du serveur MySQL au seul serveur web qui accède à la base de données**.

La figure 10-1.8 présente cette architecture type de sécurisation des serveurs via les matériels réseaux (routeurs par exemple). Les serveurs « physiques » hébergeant Apache et MySQL sont distincts.

Le serveur Apache est dans la DMZ (Zone Démilitarisée), ce qui donne un accès complet au site web de l'entreprise depuis Internet.

Le serveur MySQL est dans l'intranet, inaccessible depuis Internet. Seul le serveur physique Apache, peut accéder au serveur physique MySQL, via le port de communication réseau (3306 par exemple pour le service MySQL). Ce contrôle d'accès peut être mis en œuvre via un routeur.

Ainsi aucun ordinateur extérieur à l'entreprise ne peut accéder directement au serveur MySQL. L'administrateur devra se connecter à partir d'un poste de l'intranet, ou bien utiliser un VPN (Réseau Privé Virtuel) installé sur son poste personnel extérieur à l'entreprise.

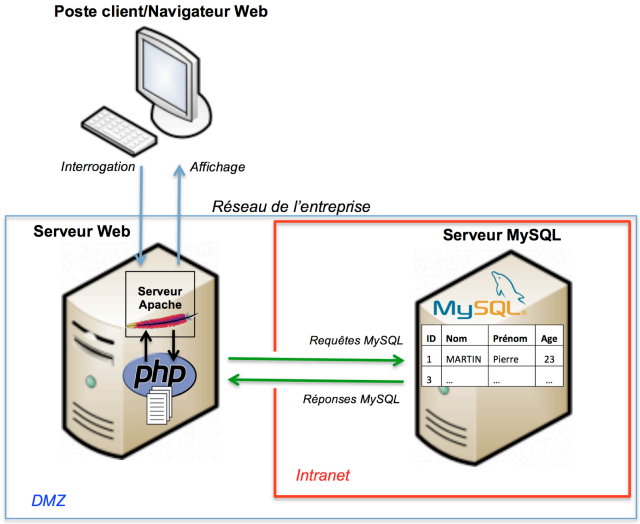


Figure 10-1.8
Sécurisation réseau du serveur MySQL.

10-1.4 PDO – PHP Data Objects – Complement

Présentation

PDO est un outil complet donnant accès, à partir d'un programme PHP, à n'importe quel type de base de données, comme MysQL, PostgeSQL ou Oracle. Il a été présenté au chapitre 10.

Programmes PHP avec filtrage et fonctions SQL

Les différentes requêtes ont été présentées au chapitre 10. Cette section présente des exemples de programmes utilisant le filtrage avec la clause WHERE et des fonctions SQL.

Le filtrage

Les programmes suivants, téléchargeables sur le site de l'éditeur, présentent les versions shell et web pour une clause `WHERE` avec un `ORDER BY` sur l'âge.

- `MySQL_PDO_query_fetch_where_order_by_personnes_shell.php`
- `MySQL_PDO_query_fetch_where_order_by_personnes_web.php`

Les programmes suivants, téléchargeables sur le site de l'éditeur, présentent les versions shell et web pour une clause `WHERE` sur l'âge ou le prénom, avec un `ORDER BY` sur le nom et une `LIMIT` à 5.

- `MySQL_PDO_query_fetch_where_order_by_limit_personnes_shell.php`
- `MySQL_PDO_query_fetch_where_order_by_limit_personnes_web.php`

Les fonctions d'agrégat

Cette section présente des exemples d'appels de **fonctions d'agrégat**. Les programmes suivants sont les versions shell et web de l'appel de la fonction `AVG()` avec une clause `GROUP BY`.

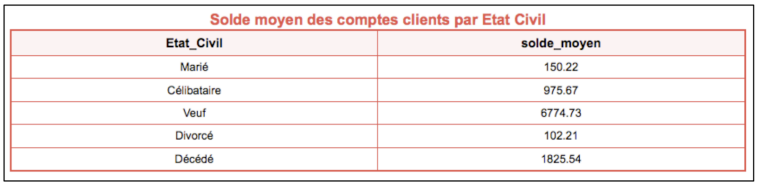
- `MySQL_PDO_query_fetch_round_avg_group_by_clients_shell.php`
- `MySQL_PDO_query_fetch_round_avg_group_by_clients_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_round_avg_group_by_clients_shell.php`.

Listing 10-1.1 : Exécution de MySQL_PDO_query_fetch_round_avg_group_by_clients_shell.php

```
$ php
MySQL_PDO_query_fetch_round_avg_group_by_clients_shell.php
------------------------------
 Solde moyen des comptes clients par Etat Civil
------------------------------
Etat_Civil    solde_moyen
------------------------------
Marié          150.22
Célibataire    975.67
Veuf           6774.73
Divorcé        102.21
Décédé         1825.54
```

La figure 10-1.9 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_round_avg_group_by_clients_web.php`.



Solde moyen des comptes clients par Etat Civil

Etat_Civil	solde_moyen
Marié	150.22
Célibataire	975.67
Veuf	6774.73
Divorcé	102.21
Décédé	1825.54

Figure 10-1.9
Affichage web query-fetch-round-avg.

Les programmes suivants présentent les versions shell et web pour les fonctions `ROUND()` et `SUM()` avec des clauses `GROUP BY` et `HAVING`.

- `MySQL_PDO_query_fetch_round_sum_group_by_having_clients_shell.php`
- `MySQL_PDO_query_fetch_round_sum_group_by_having_clients_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_round_sum_group_by_having_clients_shell.php`. L'affichage de certaines colonnes a été tronqué.

Listing 10-1.2 : Exécution de MySQL_PDO_query_fetch_round_sum_group_by_having_clients_shell.php

```
$ php
MySQL_PDO_query_fetch_round_sum_group_by_having_clients_shell.
php
------------------------------
 Liste des clients
------------------------------
ID   Nom    Prenom  Age  Date_Naissance Etat_Civil Nb_En Solde
------------------------------
1  DUPONT         JEAN        27  1987-12-28 Marié 2  1200.5
2  JACQUENOD      JEAN-CHRIST 54  1961-02-10 Marié 1  -308.87
3  MURCIAN        CAROLE      44  1970-10-20 Célib 1  3548.98
4  LERY           JEAN-MICHEL 25  1989-05-07 Marié 2  -18.98
5  DE-LA-RUE      JEAN-CHRIST 23  1991-06-18 Divor 0  -27.44
6  MARTIN         PAUL-DAVID  23  1991-08-22 Célib 0  206.21
7  MARTIN         PIERRE      56  1959-01-18 Veuf  3  1234.56
8  JACQUENOD      FREDERIC    25  1989-11-27 Marié 0  432.98
9  JACQUENOD      LAURENCE    24  1990-11-01 Marié 0  -203.18
10 DUMOULIN       JEAN-CHRIST 54  1960-08-22 Marié 2  -2186.86
11 LABONNE-JAYAT  OLIVIER     54  1960-09-23 Célib 1  -65.98
12 DE-LA-FONTAINE JEAN        110 1905-01-22 Décéd 0  1825.54
13 LEVY           SAMUEL      56  1959-03-27 Divor 3  231.87
14 DE-LA-RUE      LAURENCE    25  1989-12-13 Marié 1  2135.98
15 DUPONT         JEAN        54  1960-10-15 Veuf  2  12314.9
16 MARTIN         ALBERT      25  1989-08-15 Célib 1  213.49

Liste des clients ayant un solde total dépassant (ex: 1000) :
1000
------------------------------
 Solde total des comptes clients par âge
------------------------------
Age   solde_total
------------------------------
25   2763.47
27   1200.50
44   3548.98
54   9753.19
56   1466.43
110  1825.54
```

La figure 10-1.10 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_round_sum_group_by_having_clients_web.php`.

Le premier écran (1) affiche les clients et le formulaire de saisie du filtrage. Le second (2) affiche le résultat du traitement.

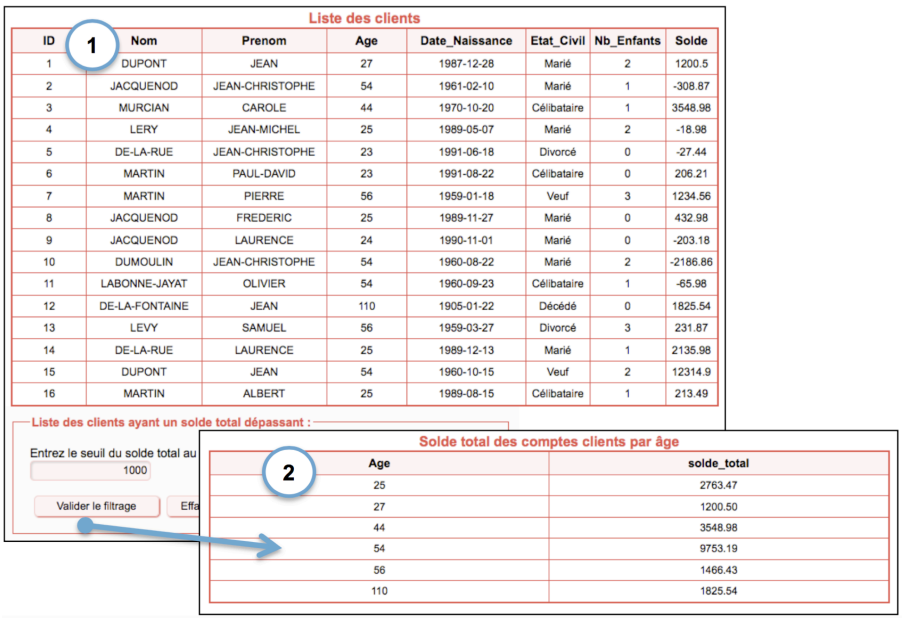


Figure 10-1.10
Affichage web query-fetch-round-sum-groupby.

Les fonctions sur les chaînes de caractères

Cette section présente des exemples de **fonctions sur les chaînes de caractères**. Les programmes suivants sont les versions shell et web de l'appel des fonctions `CONCAT()` et `LOWER()` sur le Prénom et le Nom.

- `MySQL_PDO_query_fetch_concat_lower_clients_shell.php`
- `MySQL_PDO_query_fetch_concat_lower_clients_web.php`

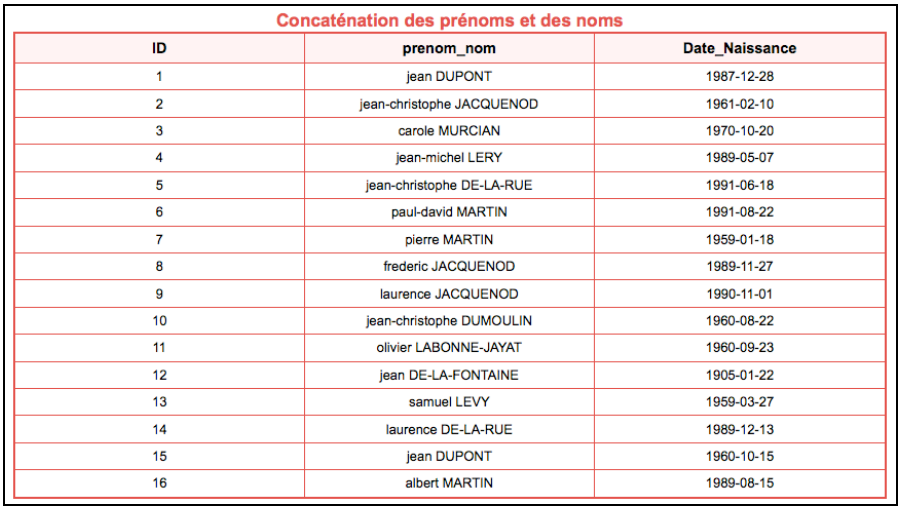
Voici l'exécution du programme `MySQL_PDO_query_fetch_concat_lower_clients_shell.php`.

Listing 10-1.3 : Exécution de MySQL_PDO_query_fetch_concat_lower_clients_shell.php

```
----------------------------------------
 Concaténation des prénoms et des noms
----------------------------------------
ID   prenom_nom              Date_Naissance
----------------------------------------
1   jean DUPONT                 1987-12-28
```

```
2   jean-christophe JACQUENOD   1961-02-10
3   carole MURCIAN              1970-10-20
4   jean-michel LERY            1989-05-07
5   jean-christophe DE-LA-RUE   1991-06-18
6   paul-david MARTIN           1991-08-22
7   pierre MARTIN               1959-01-18
8   frederic JACQUENOD          1989-11-27
9   laurence JACQUENOD          1990-11-01
10  jean-christophe DUMOULIN    1960-08-22
11  olivier LABONNE-JAYAT       1960-09-23
12  jean DE-LA-FONTAINE         1905-01-22
13  samuel LEVY                 1959-03-27
14  laurence DE-LA-RUE          1989-12-13
15  jean DUPONT                 1960-10-15
16  albert MARTIN               1989-08-15
```

La figure 10-1.11 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_concat_lower_clients_web.php`.



Concaténation des prénoms et des noms

ID	prenom_nom	Date_Naissance
1	jean DUPONT	1987-12-28
2	jean-christophe JACQUENOD	1961-02-10
3	carole MURCIAN	1970-10-20
4	jean-michel LERY	1989-05-07
5	jean-christophe DE-LA-RUE	1991-06-18
6	paul-david MARTIN	1991-08-22
7	pierre MARTIN	1959-01-18
8	frederic JACQUENOD	1989-11-27
9	laurence JACQUENOD	1990-11-01
10	jean-christophe DUMOULIN	1960-08-22
11	olivier LABONNE-JAYAT	1960-09-23
12	jean DE-LA-FONTAINE	1905-01-22
13	samuel LEVY	1959-03-27
14	laurence DE-LA-RUE	1989-12-13
15	jean DUPONT	1960-10-15
16	albert MARTIN	1989-08-15

Figure 10-1.11
Affichage web query-fetch-concat-lower.

Les fonctions mathématiques

Cette section présente des exemples de **fonctions mathématiques**. Les programmes suivants sont les versions shell et web de l'appel de la fonction `TRUNCATE()` sur le solde.

- `MySQL_PDO_query_fetch_truncate_clients_shell.php`

- `MySQL_PDO_query_fetch_truncate_clients_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_truncate_clients_shell.php`.

Listing 10-1.4 : Exécution de MySQL_PDO_query_fetch_truncate_clients_shell.php

```
$ php MySQL_PDO_query_fetch_truncate_clients_shell.php
----------------------------------------
 Solde entier
----------------------------------------
ID   Nom             Prenom    Solde_Entier
----------------------------------------
1    DUPONT          JEAN              1200
2    JACQUENOD       JEAN-CHRISTOPHE   -308
3    MURCIAN         CAROLE            3548
4    LERY            JEAN-MICHEL       -18
5    DE-LA-RUE       JEAN-CHRISTOPHE   -27
6    MARTIN          PAUL-DAVID        206
7    MARTIN          PIERRE            1234
8    JACQUENOD       FREDERIC          432
9    JACQUENOD       LAURENCE          -203
10   DUMOULIN        JEAN-CHRISTOPHE   -2186
11   LABONNE-JAYAT   OLIVIER           -65
12   DE-LA-FONTAINE  JEAN              1825
13   LEVY            SAMUEL            231
14   DE-LA-RUE       LAURENCE          2135
15   DUPONT          JEAN              12314
16   MARTIN          ALBERT            213
```

La figure 10-1.12 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_truncate_clients_web.php`.

Solde entier

ID	Nom	Prenom	Solde_Entier
1	DUPONT	JEAN	1200
2	JACQUENOD	JEAN-CHRISTOPHE	-308
3	MURCIAN	CAROLE	3548
4	LERY	JEAN-MICHEL	-18
5	DE-LA-RUE	JEAN-CHRISTOPHE	-27
6	MARTIN	PAUL-DAVID	206
7	MARTIN	PIERRE	1234
8	JACQUENOD	FREDERIC	432
9	JACQUENOD	LAURENCE	-203
10	DUMOULIN	JEAN-CHRISTOPHE	-2186
11	LABONNE-JAYAT	OLIVIER	-65
12	DE-LA-FONTAINE	JEAN	1825
13	LEVY	SAMUEL	231
14	DE-LA-RUE	LAURENCE	2135
15	DUPONT	JEAN	12314
16	MARTIN	ALBERT	213

Figure 10-1.12
Affichage web query-fetch-truncate.

Les fonctions de dates et d'heures

Cette section présente des exemples de **fonctions de dates et d'heures**. Les programmes suivants sont les versions shell et web du filtrage selon la date de naissance.

- `MySQL_PDO_query_fetch_filtre_date_naissance_clients_shell.php`
- `MySQL_PDO_query_fetch_filtre_date_naissance_clients1_web.php`
- `MySQL_PDO_query_fetch_filtre_date_naissance_clients1b_web.php`
- `MySQL_PDO_query_fetch_filtre_date_naissance_clients2_web.php`

Ces programmes demandent de saisir une date au format JJ/MM/AAAA. Il faut vérifier la validité de cette date puis la convertir au format AAAA-MM-JJ spécifique à la base de données.

Deux fonctions ont été écrites pour la validation et la conversion :

- `ValidationDate()` : Cette fonction admet deux paramètres : la date, et le format de cette date. Elle retourne un booléen qui indique si la date est valide et si elle est conforme au format indiqué. Par exemple :
  - ♦ ValidationDate('2012-02-28', 'Y-m-d')); est VRAI
  - ♦ ValidationDate('30/02/2012', 'd/m/Y')) ; est FAUX
  - ♦ ValidationDate('2012-02-28T12:12:12+02:00', 'Y-m-d\TH:i:sP')); est VRAI
  - ♦ ValidationDate('Tue, 28 Feb 2012 12:12:12 +0200', 'D, d M Y H:i:s O')); est VRAI
  - ♦ ValidationDate('14:50', 'H:i')); est VRAI
  - ♦ ValidationDate('14:77', 'H:i')); est FAUX

- `ConversionDate ()` : Cette fonction admet trois paramètres : la date, le format de départ, le format cible. Elle retourne la date indiquée, convertie dans le format cible. Le format de départ indique comment lire la date fournie en argument. Par exemple :
  - ♦ ConversionDate('2012-02-28','Y-m-d','d/m/Y'); retourne 28/02/2012
  - ♦ ConversionDate('2012-02-30','Y-m-d','d/m/Y'); retourne 01/03/2012
  - ♦ ConversionDate('28/02/2012', 'd/m/Y', 'Y-m-d'); retourne 2012-02-28
  - ♦ ConversionDate('30/02/2012', 'd/m/Y', 'Y-m-d'); retourne 2012-03-01

Ces deux fonctions utilisent la méthode `createFromFormat()` de la classe d'objet `DateTime` et la méthode `date_default_timezone_set()`. Il est nécessaire de définir le « timezone » pour le bon fonctionnement de cette méthode avec l'instruction :

```
date_default_timezone_set("Europe/Paris");
```

Ces deux fonctions sont stockées dans `MySQL_include_sprog_commun_shell.php` et `MySQL_include_sprog_commun_web.php`.

Ces deux fichiers contiennent les sous-programmes utilisés respectivement dans les versions shell et web des programmes. Il sont inclus au début de chaque programme par l'instruction :

```
include '../../INCLUDE/MySQL_include_sprog_commun_shell.php';
```

Voici ces deux fonctions PHP :

```
// -- Fonctions de vérification et de conversion des dates --
date_default_timezone_set("Europe/Paris");
// -- validation d'un format de date --
function ValidationDate($date_dep, $format = 'Y-m-d H:i:s')
{ $date_cree=DateTime::createFromFormat($format, $date_dep);
  return ($date_cree && ($date_cree->format($format) ==
$date_dep));
}
// -- Conversion d'un format de date --
function
ConversionDate($date_dep,$format_dep='d/m/Y',$format_cible='Y-
m-d')
{$date_cree=DateTime::createFromFormat($format_dep,$date_dep);
 $date_cible=$date_cree->format($format_cible);
 return $date_cible;
}
```

Le programme `MySQL_PDO_query_fetch_filtre_date_naissance_clients_shell.php` utilise la fonction SQL `DATE_FORMAT` pour transformer la date du format AAAA-MM-JJ au format JJ/MM/AAAA.

Voici la ligne de ce programme qui effectue cette requête :

```
// -- Exécution de la requête --
 $reponse = $bdd->query('SELECT
ID,Nom,Prenom,DATE_FORMAT(Date_Naissance,\'%d/%m/%Y\') As
Date_Naissance FROM clients');
```

Voici un exemple d'exécution :

Listing 10-1.5 : Exécution de MySQL_PDO_query_fetch_filtre_date_naissance_clients_shell.php

```
$ php
MySQL_PDO_query_fetch_filtre_date_naissance_clients_shell.php
-----------------------------------------
 Liste des clients
-----------------------------------------
ID   Nom           Prenom      Date_Naissance
-----------------------------------------
1   DUPONT         JEAN            28/12/1987
2   JACQUENOD      JEAN-CHRISTOPHE 10/02/1961
3   MURCIAN        CAROLE          20/10/1970
4   LERY           JEAN-MICHEL     07/05/1989
5   DE-LA-RUE      JEAN-CHRISTOPHE 18/06/1991
6   MARTIN         PAUL-DAVID      22/08/1991
7   MARTIN         PIERRE          18/01/1959
8   JACQUENOD      FREDERIC        27/11/1989
9   JACQUENOD      LAURENCE        01/11/1990
10  DUMOULIN       JEAN-CHRISTOPHE 22/08/1960
11  LABONNE-JAYAT  OLIVIER         23/09/1960
12  DE-LA-FONTAINE JEAN            22/01/1905
13  LEVY           SAMUEL          27/03/1959
14  DE-LA-RUE      LAURENCE        13/12/1989
15  DUPONT         JEAN            15/10/1960
16  MARTIN         ALBERT          15/08/1989

Afficher la liste des clients dont la date de naissance est
supérieure à (ex: 01/01/1970) : 01/01/1970
-----------------------------------------
 Liste des clients ayant une date de naissance >= 01/01/1970
-----------------------------------------
ID   Nom           Prenom      Date_Naissance
-----------------------------------------
1   DUPONT         JEAN            28/12/1987
3   MURCIAN        CAROLE          20/10/1970
4   LERY           JEAN-MICHEL     07/05/1989
5   DE-LA-RUE      JEAN-CHRISTOPHE 18/06/1991
6   MARTIN         PAUL-DAVID      22/08/1991
8   JACQUENOD      FREDERIC        27/11/1989
```

```
9   JACQUENOD      LAURENCE        01/11/1990
14  DE-LA-RUE      LAURENCE        13/12/1989
16  MARTIN         ALBERT          15/08/1989
```

Le programme `MySQL_PDO_query_fetch_filtre_date_naissance_clients1_web.php` utilise un formulaire HTML5 pour contrôler la saisie du format de la date. Voici les lignes de ce programme concernant le formulaire :

```
<form
action="MySQL_PDO_query_fetch_filtre_date_naissance_clients1_w
eb.php" method="post">
<fieldset>
<legend>Liste des clients dont la date de naissance est
sup&eacute;rieure &agrave; :</legend><br/>
Entrez la date de naissance à partir de laquelle les clients
seront affichés à (ex: 01/01/1970) : <input type="text"
name="DateNaissance" size="10" maxlength="10" pattern="[0-
9]{2}/[0-9]{2}/[0-9]{4}" /><br/><br/>
<input type="submit" name="valider" value="Valider le
filtrage" />
<input type="reset" value="Effacer le formulaire" />
</fieldset>
</form>
```

La figure 10-1.13 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_filtre_date_naissance_clients1_web.php`.

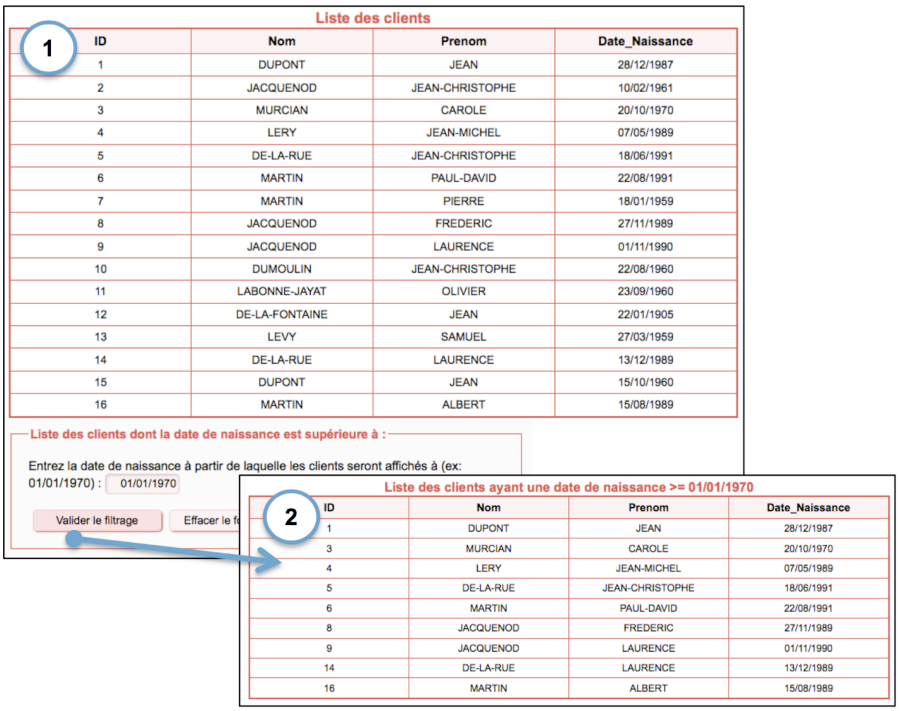


Figure 10-1.13
Affichage web query-fetch-filtre date-1.

HTML5 autorise le nouveau type « date » en lieu et place de « text ». Ce type « date » permet la saisie d'une date dans un calendrier. La syntaxe suivante a été intégrée dans le programme `MySQL_PDO_query_fetch_filtre_date_naissance_clients1b_web.php`.

```
Entrez la date de naissance à partir de laquelle les clients
seront affichés à (ex: 01/01/1970) : <input type="date"
name="DateNaissance" size="10" maxlength="10" /><br/><br/>
```

Malheureusement, au moment de la rédaction de cet ouvrage, le type « date » n'est pas supporté par l'ensemble des navigateurs. Firefox le reconnaît comme type « text », et ne propose aucun calendrier de saisie. Chrome propose un calendrier de saisie, mais le format envoyé par défaut via le formulaire est noté en anglo-saxon, ce qui provoque un échec du filtrage du programme PHP tel qu'il est écrit.

Le programme `MySQL_PDO_query_fetch_filtre_date_naissance_clients2_web.php` utilise « datepicker » de la bibliothèque JQuery pour permettre la saisie de la date dans un calendrier. Cette saisie est opérationnelle quelque soit le navigateur. Voici les lignes de syntaxes qui implémentent cette saisie, avec un calendrier en français :

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Affichage de la table clients</title>
    <link href="../../CSS/MySQL.css" rel="stylesheet"
type="text/css" />
  <link rel="stylesheet"
href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-
ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.4/jquery-
ui.js"></script>
  <script>
  $.datepicker.regional['fr'] = {
    closeText: 'Fermer',
    prevText: 'Précédent',
    nextText: 'Suivant',
    currentText: 'Aujourd\'hui',
    monthNames:
['Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Ao
ût','Septembre','Octobre','Novembre','Décembre'],
    monthNamesShort:
['Janv.','Févr.','Mars','Avril','Mai','Juin','Juil.','Août','S
ept.','Oct.','Nov.','Déc.'],
    dayNames:
['Dimanche','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Sam
edi'],
    dayNamesShort:
['Dim.','Lun.','Mar.','Mer.','Jeu.','Ven.','Sam.'],
    dayNamesMin: ['D','L','M','M','J','V','S'],
    weekHeader: 'Sem.',
    dateFormat: 'dd/mm/yy',
    firstDay: 1,
    isRTL: false,
    showMonthAfterYear: false,
    yearSuffix: ''
  };
  $.datepicker.setDefaults($.datepicker.regional['fr']);
  $(function() {
    $( "#datepicker" ).datepicker();
  });
  </script>
 </head>
 <body>
   ...

     ------------------------------------------------
```

```
     -- formulaire de saisie du critère de filtrage --
     ------------------------------------------------
     -->
     <form
action="MySQL_PDO_query_fetch_filtre_date_naissance_clients2_w
eb.php" method="post">
      <fieldset>
      <legend>Liste des clients dont la date de naissance est
sup&eacute;rieure &agrave; :</legend><br/>
      Entrez la date de naissance à partir de laquelle les
clients seront affichés à : <input type="text"
class="datepick" name="DateNaissance"><br/><br/>
      <input type="submit" name="valider" value="Valider le
filtrage" />
      <input type="reset" value="Effacer le formulaire" />
      </fieldset>
     </form>
     <script type="text/javascript">
     $(document).ready(function() {
        $('.datepick').datepicker({ dateFormat: "dd/mm/yy"});
     });
     </script>
...
```

La figure 10-1.14 présente l'écran de saisie. Le format de la date envoyée est bien en français JJ/MM/AAAA, le filtrage fonctionne parfaitement.

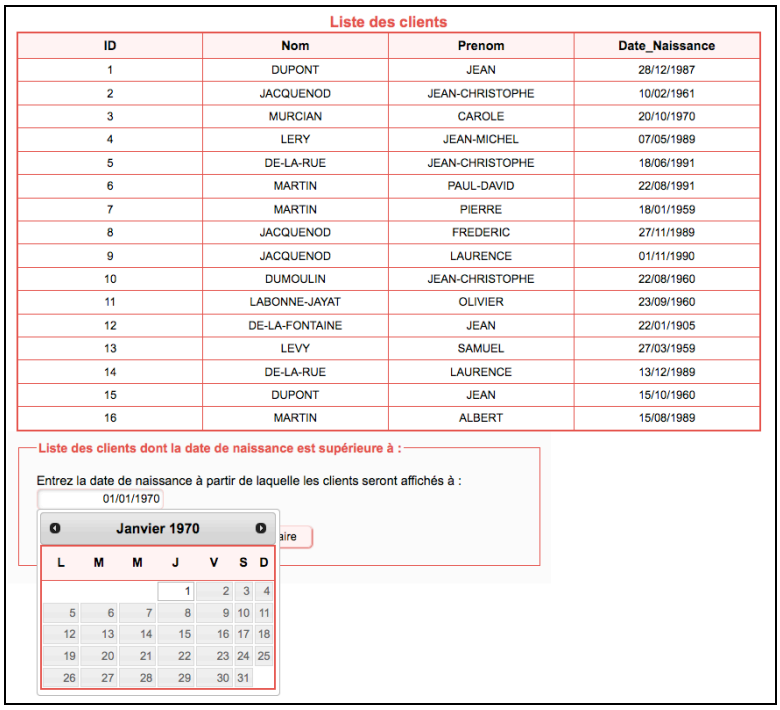


Figure 10-1.14
Affichage web query-fetch-filtre date-2.

Les programmes suivants présentent les versions shell et web effectuant le calcul de l'âge à partir de la date de naissance.

- `MySQL_PDO_query_fetch_calcul_age_clients_shell.php`
- `MySQL_PDO_query_fetch_calcul_age_clients_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_calcul_age_clients_shell.php`.

Listing 10-1.6 : Exécution de MySQL_PDO_query_fetch_calcul_age_clients_shell.php

```
$ php MySQL_PDO_query_fetch_calcul_age_clients_shell.php
-------------------------------------------------
 Calcul de l'âge
-------------------------------------------------
ID   Nom           Prenom           Age   Age_calculé
-------------------------------------------------
1    DUPONT        JEAN              27   27
2    JACQUENOD     JEAN-CHRISTOPHE   54   54
3    MURCIAN       CAROLE            44   44
4    LERY          JEAN-MICHEL       25   25
5    DE-LA-RUE     JEAN-CHRISTOPHE   23   23
```

```
6   MARTIN          PAUL-DAVID          23   23
7   MARTIN          PIERRE              56   56
8   JACQUENOD       FREDERIC            25   25
9   JACQUENOD       LAURENCE            24   24
10  DUMOULIN        JEAN-CHRISTOPHE     54   54
11  LABONNE-JAYAT   OLIVIER             54   54
12  DE-LA-FONTAINE  JEAN                110   110
13  LEVY            SAMUEL              56   56
14  DE-LA-RUE       LAURENCE            25   25
15  DUPONT          JEAN                54   54
16  MARTIN          ALBERT              25   25
```

la figure 10-1.15 présent le résultat de l'exécution du programme `MySQL_PDO_query_fetch_calcul_age_clients_web.php`.

Calcul de l'âge

ID	Nom	Prenom	Age	Age_calculé
1	DUPONT	JEAN	27	27
2	JACQUENOD	JEAN-CHRISTOPHE	54	54
3	MURCIAN	CAROLE	44	44
4	LERY	JEAN-MICHEL	25	25
5	DE-LA-RUE	JEAN-CHRISTOPHE	23	23
6	MARTIN	PAUL-DAVID	23	23
7	MARTIN	PIERRE	56	56
8	JACQUENOD	FREDERIC	25	25
9	JACQUENOD	LAURENCE	24	24
10	DUMOULIN	JEAN-CHRISTOPHE	54	54
11	LABONNE-JAYAT	OLIVIER	54	54
12	DE-LA-FONTAINE	JEAN	110	110
13	LEVY	SAMUEL	56	56
14	DE-LA-RUE	LAURENCE	25	25
15	DUPONT	JEAN	54	54
16	MARTIN	ALBERT	25	25

Figure 10-1.15
Affichage web query-fetch-filtre calcul âge.

Les jointures internes

Cette section présente des exemples de **jointure interne**. 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 ;

Les programmes suivants sont les versions shell et web d'une jointure interne avec la clause `WHERE`.

- `MySQL_PDO_query_fetch_jointure_interne_where1_shell.php`
- `MySQL_PDO_query_fetch_jointure_interne_where1_web.php`

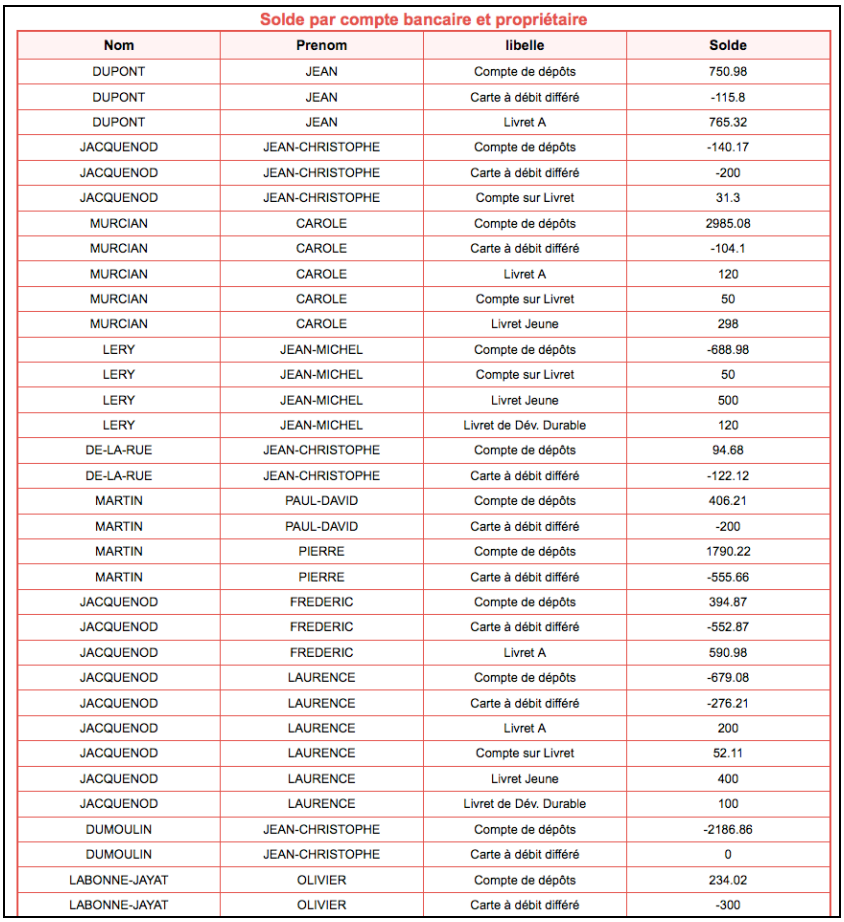
Voici l'exécution du programme `MySQL_PDO_query_fetch_jointure_interne_where1_shell.php`.

Listing 10-1.7 : 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              Carte débit di   -115.8
DUPONT         JEAN              Livret A         765.32
JACQUENOD      JEAN-CHRISTOPHE   Compte de dépô   -140.17
JACQUENOD      JEAN-CHRISTOPHE   Carte débit di   -200
JACQUENOD      JEAN-CHRISTOPHE   Compte sur Liv   31.3
MURCIAN        CAROLE            Compte de dépô   2985.08
MURCIAN        CAROLE            Carte débit di   -104.1
MURCIAN        CAROLE            Livret A         120
MURCIAN        CAROLE            Compte sur Liv   50
MURCIAN        CAROLE            Livret Jeune     298
LERY           JEAN-MICHEL       Compte de dépô   -688.98
LERY           JEAN-MICHEL       Compte sur Liv   50
LERY           JEAN-MICHEL       Livret Jeune     500
LERY           JEAN-MICHEL       Livret Dév.Dur   120
DE-LA-RUE      JEAN-CHRISTOPHE   Compte de dépô   94.68
DE-LA-RUE      JEAN-CHRISTOPHE   Carte débit di   -122.12
MARTIN         PAUL-DAVID        Compte de dépô   406.21
MARTIN         PAUL-DAVID        Carte débit di   -200
MARTIN         PIERRE            Compte de dépô   1790.22
MARTIN         PIERRE            Carte débit di   -555.66
JACQUENOD      FREDERIC          Compte de dépô   394.87
JACQUENOD      FREDERIC          Carte débit di   -552.87
JACQUENOD      FREDERIC          Livret A         590.98
JACQUENOD      LAURENCE          Compte de dépô   -679.08
JACQUENOD      LAURENCE          Carte débit di   -276.21
JACQUENOD      LAURENCE          Livret A         200
JACQUENOD      LAURENCE          Compte sur Liv   52.11
JACQUENOD      LAURENCE          Livret Jeune     400
JACQUENOD      LAURENCE          Livret Dév.Dur   100
DUMOULIN       JEAN-CHRISTOPHE   Compte de dépô   -2186.86
DUMOULIN       JEAN-CHRISTOPHE   Carte débit di   0
LABONNE-JAYAT  OLIVIER           Compte de dépô   234.02
LABONNE-JAYAT  OLIVIER           Carte débit di   -300
DE-LA-FONTAINE JEAN              Compte de dépô   1825.54
LEVY           SAMUEL            Compte de dépô   12.09
LEVY           SAMUEL            Carte débit di   -212.98
```

```
LEVY            SAMUEL          Livret A          432.76
DE-LA-RUE       LAURENCE        Compte de dépô    275.7
DE-LA-RUE       LAURENCE        Carte débit di    -104.1
DE-LA-RUE       LAURENCE        Livret A          1032.47
DE-LA-RUE       LAURENCE        Compte sur Liv    31.3
DE-LA-RUE       LAURENCE        Livret Jeune      818.38
DE-LA-RUE       LAURENCE        Livret Dév.Dur    82.23
DUPONT          JEAN            Compte de dépô    4572.1
DUPONT          JEAN            Carte débit di    -2987.65
DUPONT          JEAN            Livret A          2500
DUPONT          JEAN            Compte sur Liv    5628.34
DUPONT          JEAN            Livret Jeune      1600
DUPONT          JEAN            Livret Dév.Dur    1002.11
MARTIN          ALBERT          Compte de dépô    363.49
MARTIN          ALBERT          Carte débit di    -150
```

La figure 10-1.16 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_jointure_interne_where1_web.php`. Seules les premières lignes de l'affichage sont présentées.



Solde par compte bancaire et propriétaire

Nom	Prenom	libelle	Solde
DUPONT	JEAN	Compte de dépôts	750.98
DUPONT	JEAN	Carte à débit différé	-115.8
DUPONT	JEAN	Livret A	765.32
JACQUENOD	JEAN-CHRISTOPHE	Compte de dépôts	-140.17
JACQUENOD	JEAN-CHRISTOPHE	Carte à débit différé	-200
JACQUENOD	JEAN-CHRISTOPHE	Compte sur Livret	31.3
MURCIAN	CAROLE	Compte de dépôts	2985.08
MURCIAN	CAROLE	Carte à débit différé	-104.1
MURCIAN	CAROLE	Livret A	120
MURCIAN	CAROLE	Compte sur Livret	50
MURCIAN	CAROLE	Livret Jeune	298
LERY	JEAN-MICHEL	Compte de dépôts	-688.98
LERY	JEAN-MICHEL	Compte sur Livret	50
LERY	JEAN-MICHEL	Livret Jeune	500
LERY	JEAN-MICHEL	Livret de Dév. Durable	120
DE-LA-RUE	JEAN-CHRISTOPHE	Compte de dépôts	94.68
DE-LA-RUE	JEAN-CHRISTOPHE	Carte à débit différé	-122.12
MARTIN	PAUL-DAVID	Compte de dépôts	406.21
MARTIN	PAUL-DAVID	Carte à débit différé	-200
MARTIN	PIERRE	Compte de dépôts	1790.22
MARTIN	PIERRE	Carte à débit différé	-555.66
JACQUENOD	FREDERIC	Compte de dépôts	394.87
JACQUENOD	FREDERIC	Carte à débit différé	-552.87
JACQUENOD	FREDERIC	Livret A	590.98
JACQUENOD	LAURENCE	Compte de dépôts	-679.08
JACQUENOD	LAURENCE	Carte à débit différé	-276.21
JACQUENOD	LAURENCE	Livret A	200
JACQUENOD	LAURENCE	Compte sur Livret	52.11
JACQUENOD	LAURENCE	Livret Jeune	400
JACQUENOD	LAURENCE	Livret de Dév. Durable	100
DUMOULIN	JEAN-CHRISTOPHE	Compte de dépôts	-2186.86
DUMOULIN	JEAN-CHRISTOPHE	Carte à débit différé	0
LABONNE-JAYAT	OLIVIER	Compte de dépôts	234.02
LABONNE-JAYAT	OLIVIER	Carte à débit différé	-300

Figure 10-1.16
Affichage web query-fetch-jointure interne where-1.

Les programmes suivants présentent les versions shell et web effectuant la jointure interne avec la clause `WHERE` pour afficher le solde total de chaque client.

- `MySQL_PDO_query_fetch_jointure_interne_where2_shell.php`
- `MySQL_PDO_query_fetch_jointure_interne_where2_web.php`

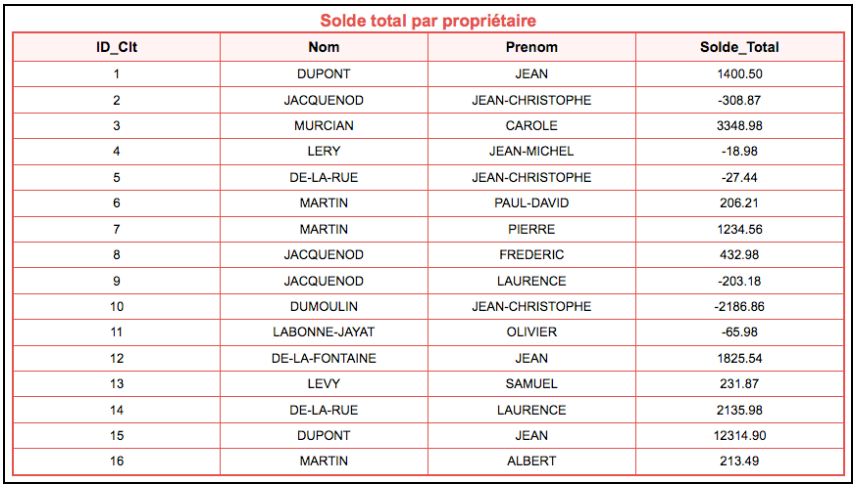
Voici l'exécution du programme `MySQL_PDO_query_fetch_jointure_interne_where2_shell.php`.

Listing 10-1.8 : Exécution de MySQL_PDO_query_fetch_jointure_interne_where2_shell.php

```
$ php MySQL_PDO_query_fetch_jointure_interne_where2_shell.php
-----------------------------------------------
 Solde total par propriétaire
```

```
-----------------------------------------------
ID_Clt Nom             Prenom         Solde_Total
-----------------------------------------------
1      DUPONT          JEAN                1400.50
2      JACQUENOD       JEAN-CHRISTOPHE     -308.87
3      MURCIAN         CAROLE              3348.98
4      LERY            JEAN-MICHEL         -18.98
5      DE-LA-RUE       JEAN-CHRISTOPHE     -27.44
6      MARTIN          PAUL-DAVID          206.21
7      MARTIN          PIERRE              1234.56
8      JACQUENOD       FREDERIC            432.98
9      JACQUENOD       LAURENCE            -203.18
10     DUMOULIN        JEAN-CHRISTOPHE     -2186.86
11     LABONNE-JAYAT   OLIVIER             -65.98
12     DE-LA-FONTAINE  JEAN                1825.54
13     LEVY            SAMUEL              231.87
14     DE-LA-RUE       LAURENCE            2135.98
15     DUPONT          JEAN                12314.90
16     MARTIN          ALBERT              213.49
```

La figure 10-1.17 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_jointure_interne_where2_web.php`.



Solde total par propriétaire

ID_Clt	Nom	Prenom	Solde_Total
1	DUPONT	JEAN	1400.50
2	JACQUENOD	JEAN-CHRISTOPHE	-308.87
3	MURCIAN	CAROLE	3348.98
4	LERY	JEAN-MICHEL	-18.98
5	DE-LA-RUE	JEAN-CHRISTOPHE	-27.44
6	MARTIN	PAUL-DAVID	206.21
7	MARTIN	PIERRE	1234.56
8	JACQUENOD	FREDERIC	432.98
9	JACQUENOD	LAURENCE	-203.18
10	DUMOULIN	JEAN-CHRISTOPHE	-2186.86
11	LABONNE-JAYAT	OLIVIER	-65.98
12	DE-LA-FONTAINE	JEAN	1825.54
13	LEVY	SAMUEL	231.87
14	DE-LA-RUE	LAURENCE	2135.98
15	DUPONT	JEAN	12314.90
16	MARTIN	ALBERT	213.49

Figure 10-1.17

Affichage web query-fetch-jointure interne where-2.

La jointure des syntaxes précédentes utilisait la clause `WHERE` afin de faciliter la compréhension, puisque cette clause avait déjà été présentée. La jointure avec la clause `WHERE` est devenue obsolète, même si elle fonctionne parfaitement. 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 programmes suivants présentent les versions shell et web effectuant la jointure interne avec la clause `INNER JOIN` pour afficher le solde total de chaque client.

- `MySQL_PDO_query_fetch_jointure_interne_inner_join_shell.php`
- `MySQL_PDO_query_fetch_jointure_interne_inner_join_web.php`

Ils affichent le même résultat que les deux programmes précédents.

Les jointures externes

Cette section présente des exemples de **jointure externe**. Les programmes suivants sont les versions shell et web effectuant la jointure externe avec la clause `LEFT JOIN` pour afficher le solde total de chaque compte, y compris ceux qui n'ont pas de propriétaire connu dans la table des clients.

- `MySQL_PDO_query_fetch_jointure_externe_left_join_shell.php`
- `MySQL_PDO_query_fetch_jointure_externe_left_join_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_jointure_externe_left_join_shell.php`.

Le compte ayant le propriétaire le client N°26 apparaît, alors qu'il est inconnu dans la table des clients.

Listing 10-1.9 : Exécution de MySQL_PDO_query_fetch_jointure_externe_left_join_shell.php

```
$ php
MySQL_PDO_query_fetch_jointure_externe_left_join_shell.php
-----------------------------------------
 Solde total par propriétaire, pour tous les clients
-----------------------------------------
ID_Clt   Nom       Prenom        Solde_Total
-----------------------------------------
1   DUPONT         JEAN              1400.50
2   JACQUENOD      JEAN-CHRISTOPHE   -308.87
3   MURCIAN        CAROLE            3348.98
4   LERY           JEAN-MICHEL       -18.98
5   DE-LA-RUE      JEAN-CHRISTOPHE   -27.44
6   MARTIN         PAUL-DAVID        206.21
7   MARTIN         PIERRE            1234.56
```

```
8   JACQUENOD       FREDERIC            432.98
9   JACQUENOD       LAURENCE            -203.18
10  DUMOULIN        JEAN-CHRISTOPHE     -2186.86
11  LABONNE-JAYAT   OLIVIER             -65.98
12  DE-LA-FONTAINE  JEAN                1825.54
13  LEVY            SAMUEL              231.87
14  DE-LA-RUE       LAURENCE            2135.98
15  DUPONT          JEAN                12314.90
16  MARTIN          ALBERT              213.49
26                                      345.29
```

La figure 10-1.18 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_jointure_externe_left_join_web.php`.

Solde total par propriétaire, pour tous les comptes

ID_Clt	Nom	Prenom	Solde_Total
1	DUPONT	JEAN	1400.50
2	JACQUENOD	JEAN-CHRISTOPHE	-308.87
3	MURCIAN	CAROLE	3348.98
4	LERY	JEAN-MICHEL	-18.98
5	DE-LA-RUE	JEAN-CHRISTOPHE	-27.44
6	MARTIN	PAUL-DAVID	206.21
7	MARTIN	PIERRE	1234.56
8	JACQUENOD	FREDERIC	432.98
9	JACQUENOD	LAURENCE	-203.18
10	DUMOULIN	JEAN-CHRISTOPHE	-2186.86
11	LABONNE-JAYAT	OLIVIER	-65.98
12	DE-LA-FONTAINE	JEAN	1825.54
13	LEVY	SAMUEL	231.87
14	DE-LA-RUE	LAURENCE	2135.98
15	DUPONT	JEAN	12314.90
16	MARTIN	ALBERT	213.49
26			345.29

Figure 10-1.18
Affichage web query-fetch-jointure externe left join.

Les programmes suivants sont les versions shell et web effectuant la jointure externe avec la clause `RIGHT JOIN` pour afficher le solde total de chaque client, y compris ceux qui n'ont pas aucun compte.

- `MySQL_PDO_query_fetch_jointure_externe_right_join_shell.php`
- `MySQL_PDO_query_fetch_jointure_externe_right_join_web.php`

Voici l'exécution du programme `MySQL_PDO_query_fetch_jointure_externe_right_join_shell.php`.

Le client N°17, JACQUES ROUSSE, apparaît alors qu'il n'a aucun compte dans la table des comptes bancaires.

MySQL_PDO_query_fetch_jointure_externe_right_join_shell.php

```
$ php
MySQL_PDO_query_fetch_jointure_externe_right_join_shell.php
```

```
-----------------------------------------
 Solde total par client, pour tous les clients
-----------------------------------------
ID_Clt   Nom          Prenom         Solde_Total
-----------------------------------------
1   DUPONT          JEAN               1400.50
2   JACQUENOD       JEAN-CHRISTOPHE    -308.87
3   MURCIAN         CAROLE             3348.98
4   LERY            JEAN-MICHEL        -18.98
5   DE-LA-RUE       JEAN-CHRISTOPHE    -27.44
6   MARTIN          PAUL-DAVID         206.21
7   MARTIN          PIERRE             1234.56
8   JACQUENOD       FREDERIC           432.98
9   JACQUENOD       LAURENCE           -203.18
10  DUMOULIN        JEAN-CHRISTOPHE    -2186.86
11  LABONNE-JAYAT   OLIVIER            -65.98
12  DE-LA-FONTAINE  JEAN               1825.54
13  LEVY            SAMUEL             231.87
14  DE-LA-RUE       LAURENCE           2135.98
15  DUPONT          JEAN               12314.90
16  MARTIN          ALBERT             213.49
17  ROUSSE          JACQUES
```

La figure 10-1.19 présente le résultat de l'exécution du programme `MySQL_PDO_query_fetch_jointure_externe_right_join_web.php`.

Solde total par client, pour tous les clients

ID_Clt	Nom	Prenom	Solde_Total
1	DUPONT	JEAN	1400.50
2	JACQUENOD	JEAN-CHRISTOPHE	-308.87
3	MURCIAN	CAROLE	3348.98
4	LERY	JEAN-MICHEL	-18.98
5	DE-LA-RUE	JEAN-CHRISTOPHE	-27.44
6	MARTIN	PAUL-DAVID	206.21
7	MARTIN	PIERRE	1234.56
8	JACQUENOD	FREDERIC	432.98
9	JACQUENOD	LAURENCE	-203.18
10	DUMOULIN	JEAN-CHRISTOPHE	-2186.86
11	LABONNE-JAYAT	OLIVIER	-65.98
12	DE-LA-FONTAINE	JEAN	1825.54
13	LEVY	SAMUEL	231.87
14	DE-LA-RUE	LAURENCE	2135.98
15	DUPONT	JEAN	12314.90
16	MARTIN	ALBERT	213.49
17	ROUSSE	JACQUES	

Figure 10-1.19
Affichage web query-fetch-jointure externe right join.

Le mode transactionnel avec MySQL et PDO

Principe

Les transactions MySQL **sécurisent l'exécution d'un groupe de requêtes** en revenant à l'état d'origine en cas de problème sur une des requêtes du groupe. En cas de succès de l'ensemble des requêtes, la validation de la transaction applique les changements de manière définitive. De plus, un mécanisme de verrouillage interdit la modification par un processus tierce, des éléments en cours de traitement par la transaction.

C'est l'exemple du traitement d'un virement bancaire, composé de deux opérations, le débit du compte initial et le crédit du compte cible, qui doit être validé ou annulé globalement.

L'annulation de la transaction et le retour à l'état d'origine se nomme « **roolback** ». La validation finale correspond à l'action de « **commit** ».

Le mode transactionnel est supporté par le moteur InnoDB de MySQL.

Les fonctions

Les méthodes de la classe PDO qui mettent en œuvre le mode transactionnel sont :

- `beginTransaction()` : cette méthode démarre une nouvelle transaction. Elle désactive le mode « autocommit ». Dès la désactivation de « autocommit », toutes les modifications sont gardées en mémoire, rien n'est réellement appliqué sur les tables (pas d'écriture sur disque). Elle retourne TRUE en cas de succès et FALSE si une erreur survient.
- `commit()` : cette méthode termine la transaction en validant les modifications. Les données sont écrites sur disque. Elle remet la connexion en « autocommit ». Cette méthode retourne TRUE en cas de succès et FALSE en cas d'erreur. Une exception PDOExeption est lancée en cas d'erreur, par exemple si aucune transaction n'est active.
- `rollback()` : cette méthode termine la transaction en annulant les modifications. Rien n'est écrit sur disque. Elle remet la connexion en « autocommit ». Elle retourne TRUE en cas de succès et FALSE en cas d'erreur. Une exception PDOExeption est lancée en cas d'erreur, par exemple si aucune transaction n'est active.

- PDO ne vérifie le support du mode transactionnel qu'au niveau du pilote qui accède à la base. Si certaines conditions particulières empêchent le fonctionnement des transactions, beginTransaction retournera tout de même TRUE sans erreur, si le démarrage de la transaction est accepté. Cela se produira par exemple quand le moteur MyISAM est utilisé pour des tables. À la fin du programme PHP, toute transaction ouverte par beginTransaction() qui n'a pas été fermée par commit() ou rollback() sera annulée automatiquement (roolback).

Les syntaxes

Nous présentons deux variations syntaxiques « simplifiées » de ces trois fonctions :

- Avec utilisation de la méthode `query()` pour effectuer les modifications. Cette fonction ne sécurise pas la requête ;
- Puis avec la méthode `prepare()`, qui sécurise la requête via les requêtes préparées.

Avec des requêtes standards

Nous utilisons la table « comptes_bancaires » pour effectuer un virement entre deux comptes. Dans cet exemple le compte ayant l'ID N°4 est débité de 100 €, et le compte ayant l'ID N°7 est crédité de 100 €.

```
$MtVirt=100 ;
$NumCptDebit=4 ;
$NumCptCredit=7 ;
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));
 // -- Initialisation des Exceptions PDO pour prepare --
 $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
 // -- Définition du codage en UTF8 --
 $bdd->exec("SET CHARACTER SET utf8");
 // -- On débute la transaction --
 $bdd->beginTransaction();
 //-- On débite le compte --
 $requete_debit='UPDATE comptes_bancaires SET Solde=Solde-
'.$MtVirt.' WHERE Id_Cpt='.$NumCptDebit;
 $reponse = $bdd->query($requete_debit);
 // -- Fermeture de la requête --
 $reponse->closeCursor();
 //-- On crédite le compte --
 $requete_credit='UPDATE comptes_bancaires SET
Solde=Solde+'.$MtVirt.' WHERE Id_Cpt='.$NumCptCredit;
 $reponse = $bdd->query($requete_credit);
 // -- Fermeture de la requête --
 $reponse->closeCursor();
 // -- Si aucune erreur, on valide la transaction --
 $reponse = $bdd->commit();
}
catch(Exception $e) {
 // -- On annule la transaction --
 $bdd->rollback();
 // -- On affiche un message d'erreur --
 echo 'Problème sur le virement - Transaction
annulée'.PHP_EOL;
 echo $e->getMessage().PHP_EOL;
}
```

Avec des requêtes préparées

Voici le même virement avec les requêtes préparées :

```
$MtVirt       = 100 ;
$NumCptDebit  =   4 ;
$NumCptCredit =   7 ;
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));
 // -- Initialisation des Exceptions PDO pour prepare --
 $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
 // -- Définition du codage en UTF8 --
 $bdd->exec("SET CHARACTER SET utf8");
 // -- On gère le virement  dans une transaction SQL --
 // -- On débute la transaction --
 $bdd->beginTransaction();
 // -- On débite le compte --
 // -- Préparation de la requête --
 $requete_sql='UPDATE comptes_bancaires SET
Solde=Solde+:MontantVir WHERE Id_Cpt=:NumCptOperation';
 $reponse = $bdd->prepare($requete_sql);
 // -- Liaison avec les paramètres --
 $reponse->bindParam(':MontantVir', $MontantVir);
 $reponse->bindParam(':NumCptOperation', $NumCptOperation,
PDO::PARAM_INT);
 // -- Affectation des valeurs pour les paramètres --
 $MontantVir      = -$MtVirt     ;
 $NumCptOperation = $NumCptDebit ;
 // -- Exécution de la requête --
 $reponse->execute();
 // -- Fermeture de la requête --
 $reponse->closeCursor();
 // -- On crédite le compte --
 // -- La requête est déjà préparée, elle ne change pas --
 // -- Paramètres déjà liés (bind) pas de changement --
 // -- Affectation des valeurs pour les paramètres --
 $MontantVir      = +$MtVirt      ;
 $NumCptOperation = $NumCptCredit ;
 // -- Exécution de la requête --
 $reponse->execute();
 // -- Fermeture de la requête --
 $reponse->closeCursor();
 // -- Si aucune erreur, on valide la transaction --
 $reponse = $bdd->commit();
}
catch(Exception $e) {
// -- On annule la transaction --
```

```
 $bdd->rollback();
 // -- On affiche un message d'erreur --
 echo 'Problème sur le virement - Transaction
annulée'.PHP_EOL;
 echo $e->getMessage().PHP_EOL;
}
```

Exemples

En shell

Le programme `MySQL_PDO_transaction_secure_prepare_shell.php` met en œuvre un virement entre deux comptes bancaires. Il utilise les tables `comptes_bancaires` et `clients_bancaires` avec une jointure interne pour présenter la liste des comptes avec le nom et le prénom de leur propriétaire. Il effectue un virement entre deux comptes sélectionnés dans la liste des comptes de dépôts.

Ce programme utilise les fonctions suivantes :

- `Affiche_Etat_Comptes()` : cette procédure affiche la liste de tous les comptes de dépôts de la table `comptes_bancaires`. Elle effectue la jointure interne entre les tables `comptes_bancaires` et `clients_bancaires` pour afficher les noms et les prénoms des propriétaires des comptes. Elle appelle `Affichage_Liste_Comptes()` pour la présentation ;
- `Saisie_Numero_Compte_Valide()` : cette fonction boucle sur la saisie d'un numéro de compte valide. Elle est utilisée pour saisir les numéros des deux comptes : le compte à débiter et le compte à créditer ;
- **`Virement()`** : cette fonction effectue le virement et utilise le mode transactionnel de MySQL.
- `Info_Compte()` : cette fonction récupère les informations d'un seul compte bancaire dans la table `comptes_bancaires`.
- `Affichage_Liste_Comptes()` : cette procédure « outil » affiche l'état des comptes du tableau passé en argument. Elle est appelée par `Affiche_Etat_Comptes()`, mais également deux fois, avant le virement pour présenter les deux comptes sur lesquels le virement est effectué, et après celui-ci pour confirmer le traitement.

Listing 10-1.10 : Programme MySQL_PDO_transaction_secure_prepare_shell.php

```
<?php
 include '../../INCLUDE/MySQL_include_param_dbb.php';
 $ERR_TRAIT=false;
 // -- On affiche l'état des comptes AVANT la transaction --
 $Tab_Tous_les_Comptes=Affiche_Etat_Comptes("Etat des comptes
AVANT le virement");
 // -- On récupère la colonne des ID_Cpt --
```

```
 if (!$ERR_TRAIT)
 {

$Tab_Colonne_IDCpt=array_column($Tab_Tous_les_Comptes,'ID_Cpt'
);
  // -- Initialisation des infos sur les comptes --
  $Infos_Cpt_Debit =array();
  $Infos_Cpt_Credit=array();
  // -- Saisie du numéro de compte à débiter --
  $Num_Cpt_Debit=Saisie_Numero_Compte_Valide('Debit');
  // -- Saisie du numéro de compte à créditer --
  $Num_Cpt_Credit=Saisie_Numero_Compte_Valide('Credit');
  // -- Saisie du montant du virement --
  echo "Montant du débit            : ";
  fscanf(STDIN,"%s",$Montant_Virement_saisi)    ;
  // -- Post traitement du montant du virement --

$Montant_Virement_saisi=str_replace(",",".",$Montant_Virement_
saisi);
  $Montant_Virement=floatval($Montant_Virement_saisi);

$Montant_Virement_formate=number_format($Montant_Virement,2,",
"," ")." €";
  // -- Affichage avant confirmation --
  $Tab_deux_comptes[0]=$Infos_Cpt_Debit;
  $Tab_deux_comptes[1]=$Infos_Cpt_Credit;
  Affichage_Liste_Comptes("Résumé : Virement de
$Montant_Virement_formate, du compte $Num_Cpt_Debit -> le
compte $Num_Cpt_Credit",$Tab_deux_comptes);
  // -- Demande de confirmation --
  echo "Confirmez le virement (o/n) : ";
  fscanf(STDIN,"%s",$ConfirmationVirement);
  if ($ConfirmationVirement == "o")
  {
   // == On gère le virement dans une transaction SQL ==

$virementOK=Virement($Num_Cpt_Debit,$Num_Cpt_Credit,$Montant_V
irement);
   // ------------------------------------------------
   if ($virementOK)
   {
    // -- On affiche le résultat du virement --
    $Tab_deux_comptes[0]=Info_Compte($Num_Cpt_Debit);
    $Tab_deux_comptes[1]=Info_Compte($Num_Cpt_Credit);
    if (!$ERR_TRAIT)
      Affichage_Liste_Comptes("Résultat : Virement de
$Montant_Virement_formate, du compte $Num_Cpt_Debit -> le
compte $Num_Cpt_Credit",$Tab_deux_comptes);
```

```
   }
  }
 }
 // ************** Sous-programmes ******************
 // ========================================================
 // -- Fonction d'affichage de l'état de tous les comptes --
 // ========================================================
 function Affiche_Etat_Comptes($texte)
 {
  global
$ERR_TRAIT,$TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_A
DM,$MDP_ADM;
  try {
   // -- Contexte pour le message d'erreur --
   $contexte="Connexion base de données";
   // -- Connexion de la base de données --
   $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
$MDP_ADM,
         array(PDO::ATTR_PERSISTENT => true));
   // -- Définition du codage en UTF8 --
   $bdd->exec("SET CHARACTER SET utf8");
   // -- Initialisation des Exceptions PDO pour prepare --
   $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
   // -- Limitation aux comptes de dépôt --
   $type_compte="Compte_Dépôts";
   // -- On affiche les comptes courant Avant le virement --
   // -- Contexte pour le message d'erreur --
   $contexte="Problème de requête";
   // -- Préparation de la requête --
   $requete_sql='SELECT
cb.ID_Cpt,cb.Agence,cb.Numero,cb.Type,cl.Nom,cl.Prenom,ROUND(c
b.Solde,2) Solde_Compte FROM comptes_bancaires cb INNER JOIN
clients_bancaires cl ON cb.ID_Clt=cl.ID_Clt WHERE
Type=:type_compte';
   $RequetePreparee = $bdd->prepare($requete_sql);
   // -- Liaison avec les paramètres --
   $RequetePreparee->bindParam(':type_compte', $type_compte);
   // -- Exécution de la requête --
    $RequetePreparee->execute();
   // -- Retourne un tableau associatif --
   $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
   // -- Boucle de traitement de chaque client --
   $Tab_Comptes=$RequetePreparee->fetchAll();
   // -- Conversion de la colonne solde au format français --
   foreach ($Tab_Comptes as $Num => $un_cpt)
   {
```

```
$un_cpt['Solde_Compte']=number_format($un_cpt['Solde_Compte'],
2,","," ")." €";
     $Tab_Comptes[$Num]=$un_cpt;
   }
   // -- Affichage des données retournées --
   Affichage_Liste_Comptes($texte,$Tab_Comptes);
   // -- Fermeture de la requête --
   $RequetePreparee->closeCursor();
   return $Tab_Comptes;
  }
  catch(Exception $e)
  {
   echo $contexte.' : '.$e->getMessage().PHP_EOL;
   $ERR_TRAIT=true;
  }
 }
 // ===============================================
 // -- Fonction de saisie d'un numéro de compte --
 // ===============================================
 function Saisie_Numero_Compte_Valide($type_saisie)
 {
  global
$Infos_Cpt_Debit,$Infos_Cpt_Credit,$Tab_Colonne_IDCpt,$Tab_Tou
s_les_Comptes;
  $Infos_Cpt_xxx=array();
  if ($type_saisie == "Debit") $Texte_Action="à débiter ";
  else $Texte_Action="à créditer";
  // -- Boucle de saisie --
  while (count($Infos_Cpt_xxx) == 0)
  {
   echo "Numéro du compte ".$Texte_Action." : ";
   fscanf(STDIN,"%d",$Num_Cpt) ;
   // -- On récupère l'indice numérique de la case --
   $numcase=array_search($Num_Cpt,$Tab_Colonne_IDCpt);
   // -- array_search() retourne le numéro de la case --
   // -- du tableau ou bien false en cas d'échec --
   // -- attention il faut utiliser le triple = afin de --
   // -- résoudre le problème de la donnée trouvée dans --
   // -- la case 0 valeur qui peut être interprétée comme --
   // -- false si le test est noté : if (!$numcase) --
   if ($numcase == false)
     echo "Compte $Num_Cpt inexistant !".PHP_EOL;
   else
   {
     $Infos_Cpt_xxx=$Tab_Tous_les_Comptes[$numcase];
     if ($type_saisie == "Debit")
        $Infos_Cpt_Debit=$Infos_Cpt_xxx;
```

```
     else
        $Infos_Cpt_Credit=$Infos_Cpt_xxx;
   }
  }
  return $Num_Cpt;
 }
 // ========================
 // -- Fonction de virement --
 // ========================
 function Virement($NumCptDebit,$NumCptCredit,$MtVirt)
 {
  global
$TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_ADM,$MDP_ADM
;
  $virement_effectue=true;
  $ConnexionBDD=false;
  $TransactionDemarree=false;
  try
  {
   // -- Contexte pour le message d'erreur --
   $contexte="Connexion base de données";
   // -- Connexion de la base de données --
   $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
$MDP_ADM,
         array(PDO::ATTR_PERSISTENT => true));
   $ConnexionBDD=true;
   // -- Initialisation des Exceptions PDO pour prepare --
   $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
   // -- Définition du codage en UTF8 --
   $bdd->exec("SET CHARACTER SET utf8");
   // -- On gère le virement  dans une transaction SQL --
   // ============================
   // == On débute la transaction ==
   // ============================
   // -- Contexte pour le message d'erreur --
   $contexte="Initialisation virement";
   $bdd->beginTransaction();
   $TransactionDemarree=true;
   // ========================
   // == On débite le compte ==
   // ========================
   // -- Contexte pour le message d'erreur --
   $contexte="Débit du compte"     ;
   // -- Préparation de la requête --
   $requete_sql='UPDATE comptes_bancaires SET
Solde=Solde+:MontantVir WHERE Id_Cpt=:NumCptOperation';
```

```
   $reponse = $bdd->prepare($requete_sql);
   // -- Liaison avec les paramètres --
   $reponse->bindParam(':MontantVir', $MontantVir);
   $reponse->bindParam(':NumCptOperation', $NumCptOperation,
PDO::PARAM_INT);
   // -- Affectation des valeurs pour les paramètres --
   $MontantVir      = -$MtVirt     ;
   $NumCptOperation = $NumCptDebit ;
   // -- Exécution de la requête --
   $reponse->execute();
   // -- Fermeture de la requête --
   $reponse->closeCursor();
   // ==========================
   // == On crédite le compte ==
   // ==========================
   // -- Contexte pour le message d'erreur --
   $contexte="Crédit du compte";
   // -- La requête est déjà préparée, elle ne change pas --
   // -- les paramètres sont déjà liés (bind) ils ne changent
pas --
   // -- Affectation des valeurs pour les paramètres --
   $MontantVir      = +$MtVirt      ;
   $NumCptOperation = $NumCptCredit ;
   // -- Exécution de la requête --
   $reponse->execute();
   // -- Fermeture de la requête --
   $reponse->closeCursor();
   // ============================
   // == On valide la transaction ==
   // ============================
   // -- Si aucune erreur, on valide la transaction --
   $contexte="Validation virement";
   $reponse = $bdd->commit();
   // -- On confirme le virement --
   echo 'Virement effectué !'.PHP_EOL;
  }
  catch(Exception $e)
  {
   // ============================
   // == On annule la transaction ==
   // ============================
   if ($ConnexionBDD && $TransactionDemarree)
   {
    try
    {
       $bdd->rollback();
    }
    catch(Exception $er)
```

```
    {
     // -- On affiche un message d'erreur --
     echo 'Problème sur le virement - Transaction
annulée'.PHP_EOL;
     echo 'Annulation : '.$er->getMessage().PHP_EOL;
     $virement_effectue=false;
    }
   }
   // -- On affiche un message d'erreur --
   echo 'Problème sur le virement - Transaction
annulée'.PHP_EOL;
   echo $contexte.' : '.$e->getMessage().PHP_EOL;
   $virement_effectue=false;
  }
  return $virement_effectue;
 }
 // =======================================================
 // -- Fonction d'information de l'état d'un seul compte --
 // =======================================================
 function Info_Compte($compte)
 {
  global
$ERR_TRAIT,$TYPE_DBB,$SERVEUR,$BASEDD,$TABLEPERSONNES,$LOGIN_A
DM,$MDP_ADM;
  try
  {
   // -- Contexte pour le message d'erreur --
   $contexte="Connexion base de données";
   // -- Connexion de la base de données --
   $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
$MDP_ADM,
          array(PDO::ATTR_PERSISTENT => true));
   // -- Définition du codage en UTF8 --
   $bdd->exec("SET CHARACTER SET utf8");
   // -- Initialisation des Exceptions PDO pour prepare --
   $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
   $contexte="Problème de requête sur la table";
   // -- Préparation de la requête --
   $requete_sql='SELECT
cb.ID_Cpt,cb.Agence,cb.Numero,cb.Type,cl.Nom,cl.Prenom,ROUND(c
b.Solde,2) Solde_Compte FROM comptes_bancaires cb INNER JOIN
clients_bancaires cl ON cb.ID_Clt=cl.ID_Clt WHERE
ID_Cpt=:compte';
   $RequetePreparee = $bdd->prepare($requete_sql);
   // -- Liaison avec les paramètres --
```

```
   $RequetePreparee->bindParam(':compte', $compte,
PDO::PARAM_INT);
   // -- Exécution de la requête --
   $RequetePreparee->execute();
   // -- Retourne un tableau associatif --
   $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
   // -- Boucle de traitement de chaque client --
   $Tab_Infos_Cpt=$RequetePreparee->fetchAll();
   // -- Conversion de la colonne solde au format français --
   foreach ($Tab_Infos_Cpt as $Num => $un_cpt)
   {

$un_cpt['Solde_Compte']=number_format($un_cpt['Solde_Compte'],
2,","," ")." €";
    $Tab_Infos_Cpt[$Num]=$un_cpt;
   }
   $Tab_Infos_Un_Cpt=$Tab_Infos_Cpt[0];
   return $Tab_Infos_Un_Cpt;
  }
  catch(Exception $e)
  {
   echo $contexte.' : '.$e->getMessage().PHP_EOL;
   $ERR_TRAIT=true;
  }
 }
 // ====================================
 // -- Fonction d'affichage du tableau --
 // ====================================
 function Affichage_Liste_Comptes($texte,$tab_mixte)
 {
  if (count($tab_mixte)==0)
    echo 'Aucun élément à afficher.';
  else
  {
   // -- Affichage entête du tableau --
   reset($tab_mixte);
   $un_compte=current($tab_mixte);
   $liste_champs=array_keys($un_compte);
   echo "-----------------------------------------------------
----------------------------------".PHP_EOL;
   echo "               $texte".PHP_EOL;
   echo "-----------------------------------------------------
----------------------------------".PHP_EOL;
   foreach($liste_champs as $nom_champ)
   {
    if ($nom_champ == "ID_Cpt")
     fprintf(STDOUT,"%6s|",$nom_champ);
    else if ($nom_champ == "Agence")
```

```
     fprintf(STDOUT," %-6s|",$nom_champ);
    else if ($nom_champ == "Numero")
     fprintf(STDOUT," %-8s|",$nom_champ);
    else if ($nom_champ == "Type")
     fprintf(STDOUT," %-12s|",$nom_champ);
    else if ($nom_champ == "Nom")
     fprintf(STDOUT," %-15s|",$nom_champ);
    else if ($nom_champ == "Prenom")
     fprintf(STDOUT," %-16s|",$nom_champ);
    else if ($nom_champ == "Solde_Compte")
     fprintf(STDOUT,"%-11s|",$nom_champ);
    else
     echo "$nom_champ\t";
   }
   echo PHP_EOL;
   echo "-----------------------------------------------------
---------------------------------".PHP_EOL;
   // -- boucle de traitement de chaque compte --
   foreach ($tab_mixte as $un_compte)
   {
   // -- On affiche le contenu des champs --
   foreach($liste_champs as $nom_champ)
   {
    if ($nom_champ == "ID_Cpt")
     fprintf(STDOUT,"%5s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Agence")
     fprintf(STDOUT," %-6s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Numero")
     fprintf(STDOUT," %-8s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Type")
     fprintf(STDOUT,"%-13s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Nom")
     fprintf(STDOUT," %-15s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Prenom")
     fprintf(STDOUT," %-16s|",$un_compte[$nom_champ]);
    else if ($nom_champ == "Solde_Compte")
     fprintf(STDOUT,"%13s|",$un_compte[$nom_champ]);
    else
     echo $un_compte[$nom_champ]."\t";
   }
   echo PHP_EOL;
   }
   echo "-----------------------------------------------------
---------------------------------".PHP_EOL;
  }
 }
?>
```

Voici un exemple d'exécution. Les saisies et les comptes sur lesquels porte le virement sont en gras :

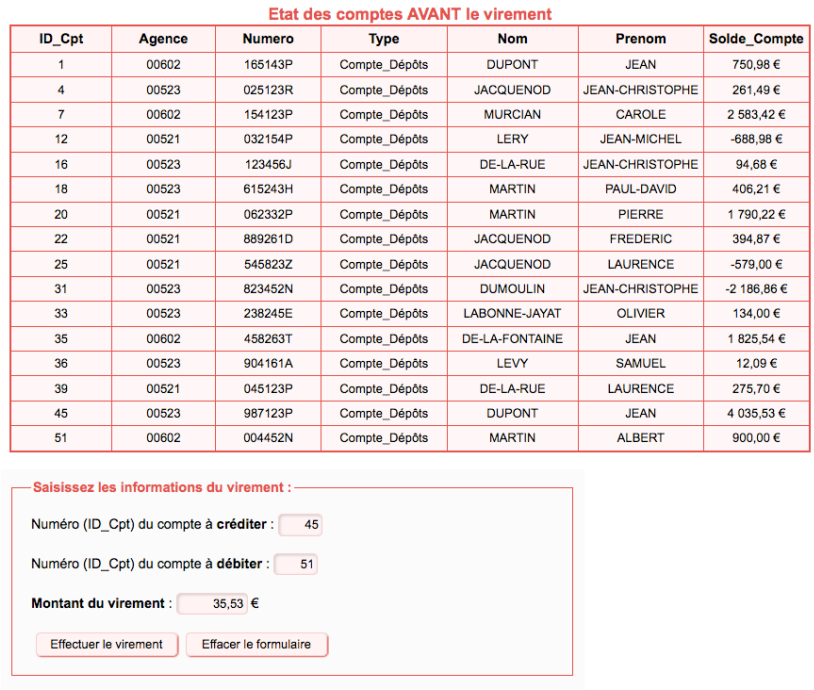
Listing 10-1.11 : Exécution de MySQL_PDO_transaction_secure_prepare_shell.php

```
$ php MySQL_PDO_transaction_secure_prepare_shell.php
------------------------------------------------------------
            Etat des comptes AVANT le virement
------------------------------------------------------------
ID_Cpt|Agence|Numero|Type       |Nom    |Prenom| Solde_Compte|
------------------------------------------------------------
   1|00602|165143P|Compte_Dépôts|DUPONT   |JEAN   |   750,98€|
   4|00523|025123R|Compte_Dépôts|JACQUENOD|JEAN-CH|  -140,17€|
   7|00602|154123P|Compte_Dépôts|MURCIAN  |CAROLE | 2 985,08€|
  12|00521|032154P|Compte_Dépôts|LERY     |JEAN-MI|  -688,98€|
  16|00523|123456J|Compte_Dépôts|DE-LA-RUE|JEAN-CH|    94,68€|
  18|00523|615243H|Compte_Dépôts|MARTIN   |PAUL-DA|   406,21€|
  20|00521|062332P|Compte_Dépôts|MARTIN   |PIERRE | 1 790,22€|
  22|00521|889261D|Compte_Dépôts|JACQUENOD|FREDERI|   394,87€|
  25|00521|545823Z|Compte_Dépôts|JACQUENOD|LAURENC|  -679,08€|
  31|00523|823452N|Compte_Dépôts|DUMOULIN |JEAN-CH|-2 186,86€|
  33|00523|238245E|Compte_Dépôts|LABONNE-J|OLIVIER|   234,02€|
  35|00602|458263T|Compte_Dépôts|DE-LA-FON|JEAN   | 1 825,54€|
  36|00523|904161A|Compte_Dépôts|LEVY     |SAMUEL |    12,09€|
  39|00521|045123P|Compte_Dépôts|DE-LA-RUE|LAUREN |   275,70€|
  45|00523|987123P|Compte_Dépôts|DUPONT   |JEAN   | 4 572,10€|
  51|00602|004452N|Compte_Dépôts|MARTIN   |ALBERT |   363,49€|
------------------------------------------------------------
Numéro du compte à débiter  : 7
Numéro du compte à créditer : 4
Montant du débit            : 185,08
------------------------------------------------------------
   Résumé : Virement de 185,08 €, du compte 7 -> le compte 4
------------------------------------------------------------
ID_Cpt|Agence|Numero|Type       |Nom    |Prenom| Solde_Compte|
------------------------------------------------------------
   7|00602|154123P|Compte_Dépôts|MURCIAN  |CAROLE | 2 985,08€|
   4|00523|025123R|Compte_Dépôts|JACQUENOD|JEAN-CH|  -140,17€|
------------------------------------------------------------
Confirmez le virement (o/n) : o
Virement effectué !
------------------------------------------------------------
  Résultat : Virement de 185,08 €, du compte 7 -> le compte 4
------------------------------------------------------------
ID_Cpt|Agence|Numero|Type       |Nom    |Prenom| Solde_Compte|
------------------------------------------------------------
   7|00602|154123P|Compte_Dépôts|MURCIAN  |CAROLE | 2 800,00€|
   4|00523|025123R|Compte_Dépôts|JACQUENOD|JEAN-CH|    44,91€|
```

Pour le web

Le programme `MySQL_PDO_transaction_nosecure_query_shell.php` est la version non sécurisée, utilisant la méthode `query()` à la place de la méthode `prepare()`. Le programme `MySQL_PDO_transaction_secure_prepare_web.php` est la version web du programme `MySQL_PDO_transaction_secure_prepare_shell.php`. Voici un exemple d'exécution :

La figure 10-1.20 présente le premier écran qui affiche la liste des comptes, et un formulaire de saisie du numéro du compte à débiter, du numéro du compte à créditer et du montant du virement.



Etat des comptes AVANT le virement

ID_Cpt	Agence	Numero	Type	Nom	Prenom	Solde_Compte
1	00602	165143P	Compte_Dépôts	DUPONT	JEAN	750,98 €
4	00523	025123R	Compte_Dépôts	JACQUENOD	JEAN-CHRISTOPHE	261,49 €
7	00602	154123P	Compte_Dépôts	MURCIAN	CAROLE	2 583,42 €
12	00521	032154P	Compte_Dépôts	LERY	JEAN-MICHEL	-688,98 €
16	00523	123456J	Compte_Dépôts	DE-LA-RUE	JEAN-CHRISTOPHE	94,68 €
18	00523	615243H	Compte_Dépôts	MARTIN	PAUL-DAVID	406,21 €
20	00521	062332P	Compte_Dépôts	MARTIN	PIERRE	1 790,22 €
22	00521	889261D	Compte_Dépôts	JACQUENOD	FREDERIC	394,87 €
25	00521	545823Z	Compte_Dépôts	JACQUENOD	LAURENCE	-579,00 €
31	00523	823452N	Compte_Dépôts	DUMOULIN	JEAN-CHRISTOPHE	-2 186,86 €
33	00523	238245E	Compte_Dépôts	LABONNE-JAYAT	OLIVIER	134,00 €
35	00602	458263T	Compte_Dépôts	DE-LA-FONTAINE	JEAN	1 825,54 €
36	00523	904161A	Compte_Dépôts	LEVY	SAMUEL	12,09 €
39	00521	045123P	Compte_Dépôts	DE-LA-RUE	LAURENCE	275,70 €
45	00523	987123P	Compte_Dépôts	DUPONT	JEAN	4 035,53 €
51	00602	004452N	Compte_Dépôts	MARTIN	ALBERT	900,00 €

Figure 10-1.20
Virement : écran de saisie.

La figure 10-1.21 présente l'écran suivant qui affiche l'état des comptes avant le virement, et demande une confirmation :

Résumé : Virement de 35,53 €, du compte 45 -> le compte 51

ID_Cpt	Agence	Numero	Type	Nom	Prenom	Solde_Compte
45	00523	987123P	Compte_Dépôts	DUPONT	JEAN	4 035,53 €
51	00602	004452N	Compte_Dépôts	MARTIN	ALBERT	900,00 €

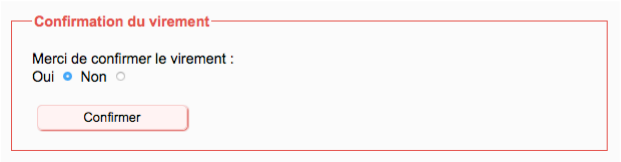


Figure 10-1.21
Virement : état des comptes sélectionnés avant virement

La figure 10-1.22 présente l'écran qui confirme la validation du virement et affiche l'état des comptes après le traitement.

Résultat : Virement de 35,53 €, du compte 45 -> le compte 51

ID_Cpt	Agence	Numero	Type	Nom	Prenom	Solde_Compte
45	00523	987123P	Compte_Dépôts	DUPONT	JEAN	4 000,00 €
51	00602	004452N	Compte_Dépôts	MARTIN	ALBERT	935,53 €

Figure 10-1.22
Virement : état des comptes sélectionnés après virement

Voici le programme `MySQL_PDO_transaction_secure_prepare_web.php` :

Listing 10-1.12 : Programme MySQL_PDO_transaction_secure_prepare_web.php

```
<?php
// On démarre la session AVANT d'écrire du code HTML
// pour les variables de session
session_start();
include
'INCLUDE/MySQL_PDO_transaction_include_sprog_commun_web.php';
include 'INCLUDE/MySQL_PDO_transaction_include_param_dbb.php';
?>
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Virement bancaire</title>
    <link href="CSS/MySQL_PDO_transaction.css"
rel="stylesheet" type="text/css" />
 </head>
 <body>
 <?php
 $ERR_TRAIT=false;
 // -------------------------
```

```
 // -- Début du traitement --
 // ------------------------
 // ------------------------------------------------------
 // Page appelée de plusieurs manières différentes
 // ------------------------------------------------------
 if (!empty($_POST['confirmation_virement']))
 {
  if (isset($_POST['RepVir'])) $ConfirmationVirement =
$_POST['RepVir'] ;
  else $ConfirmationVirement = 'non' ;
  if ($ConfirmationVirement == "oui")
  {
   $Num_Cpt_Debit            = $_SESSION['Num_Cpt_Debit']   ;
   $Num_Cpt_Credit           = $_SESSION['Num_Cpt_Credit']  ;
   $Montant_Virement_formate =
$_SESSION['Montant_Virement_formate'] ;
   $Montant_Virement         = $_SESSION['Montant_Virement'];

   // == On gère le virement  dans une transaction SQL ==

$virementOK=Virement($Num_Cpt_Debit,$Num_Cpt_Credit,$Montant_V
irement);
   // ---------------------------------------------------
   if ($virementOK)
   {
    // -- On affiche le résultat du virement --
    $Tab_deux_comptes[0]=Info_Compte($Num_Cpt_Debit) ;
    $Tab_deux_comptes[1]=Info_Compte($Num_Cpt_Credit);
    if (!$ERR_TRAIT)
      Affichage_Liste_Comptes("R&eacute;sultat : Virement de
$Montant_Virement_formate, du compte $Num_Cpt_Debit -> le
compte $Num_Cpt_Credit",$Tab_deux_comptes);
   }
  }
  else
  {
    $TitreMessage="Aucun virement"       ;
    $TexteMessage="Aucun virement n'a &eacute;t&eacute;
effectu&eacute; !";
    Affiche_Message_Erreur($TitreMessage,$TexteMessage);
  }
 }
 elseif (!empty($_POST['info_virement']))
 {
  // -- On récupère la variable de session le tableau des
comptes --
  $Tab_Tous_les_Comptes=$_SESSION['Tab_Tous_les_Comptes'] ;
```

```
$Tab_Colonne_IDCpt=array_column($Tab_Tous_les_Comptes,'ID_Cpt'
);
  // -- Initialisation des infos sur les comptes --
  $Infos_Cpt_Debit =array();
  $Infos_Cpt_Credit=array();
  // -- On récupère les valeurs saisies --
  if (isset($_POST['Num_Cpt_Debit'])) $Num_Cpt_Debit =
$_POST['Num_Cpt_Debit'] ;
  else $Num_Cpt_Debit = ''     ;
  if (isset($_POST['Num_Cpt_Credit'])) $Num_Cpt_Credit =
$_POST['Num_Cpt_Credit'] ;
  else $Num_Cpt_Credit = ''  ;
  if (isset($_POST['MtVir'])) $MtVir = $_POST['MtVir'] ;
  else $MtVir     = ''     ;
  // -- Protection contre l'injection HTML --
  $Num_Cpt_Debit  = strip_tags($Num_Cpt_Debit) ;
  $Num_Cpt_Credit = strip_tags($Num_Cpt_Credit);
  $MtVir          = strip_tags($MtVir)         ;
  // -- Normalisation au format entier ou réel --
  $Num_Cpt_Debit  = intval($Num_Cpt_Debit)               ;
  $Num_Cpt_Credit = intval($Num_Cpt_Credit)              ;
  $MtVir          = floatval(str_replace(",",".",$MtVir));
  if (($Num_Cpt_Debit!=0)||($Num_Cpt_Credit!=0)||($MtVir!=0))
  {
   $Montant_Virement_formate=number_format($MtVir,2,","," ")."
&euro;";
   // -- On récupère l'indice numérique de la case --
   $numcaseDebit  =
array_search($Num_Cpt_Debit,$Tab_Colonne_IDCpt) ;
   $numcaseCredit =
array_search($Num_Cpt_Credit,$Tab_Colonne_IDCpt);
   // array_search() retourne le numéro de la case du tableau
   // ou bien false en cas d'échec
   // attention il faut utiliser le triple = afin de résoudre
   // le problème de la donnée trouvée dans la case 0
   // valeur qui peut être interprétée comme false si le test
   // est noté : if (!$numcaseDebit)
   if ($numcaseDebit == false)
   {
    $TitreMessage = "Num&eacute;ro de compte invalide"     ;
    $TexteMessage = "Compte &agrave; d&eacute;biter
num&eacute;ro $Num_Cpt_Debit inexistant !";
    Affiche_Message_Erreur($TitreMessage,$TexteMessage);
    $ERR_TRAIT    = true;
   }
   elseif ($numcaseCredit == false)
   {
```

```
    $TitreMessage = "Num&eacute;ro de compte invalide"     ;
    $TexteMessage = "Compte &agrave; cr&eacute;diter
num&eacute;ro $Num_Cpt_Credit inexistant !";
    Affiche_Message_Erreur($TitreMessage,$TexteMessage);
    $ERR_TRAIT    = true;
   }
   else
   {
     // -- On affiche le résultat du virement --
     $Tab_deux_comptes[0] = Info_Compte($Num_Cpt_Debit) ;
     $Tab_deux_comptes[1] = Info_Compte($Num_Cpt_Credit);
     if (!$ERR_TRAIT)
        Affichage_Liste_Comptes("R&eacute;sum&eacute; :
Virement de $Montant_Virement_formate, du compte
$Num_Cpt_Debit -> le compte
$Num_Cpt_Credit",$Tab_deux_comptes);
     $_SESSION['Num_Cpt_Debit']            = $Num_Cpt_Debit  ;
     $_SESSION['Num_Cpt_Credit']           = $Num_Cpt_Credit ;
     $_SESSION['Montant_Virement_formate'] =
$Montant_Virement_formate ;
     $_SESSION['Montant_Virement']         = $MtVir          ;
     ?>
     <br/>
     <form
action="MySQL_PDO_transaction_secure_prepare_web.php"
method="post">
     <fieldset>
     <legend>Confirmation du virement</legend><br/>
     Merci de confirmer le virement :<br/>
     Oui <input type="radio" name="RepVir" value="oui">
     Non <input type="radio" name="RepVir" value="non"
checked="checked"> <br/><br/>
     <input type="submit" name="confirmation_virement"
value="Confirmer" />
     </fieldset>
     </form>
     <?php
   }
  }
 }
 else
 {
  // -- On affiche l'état des comptes AVANT la transaction --
  $Tab_Tous_les_Comptes=Affiche_Etat_Comptes("Etat des comptes
AVANT le virement");
  // -- On récupère la colonne des ID_Cpt --
  if (!$ERR_TRAIT)
  {
```

```
   // Conserve en variable de session le tableau des comptes
   $_SESSION['Tab_Tous_les_Comptes']=$Tab_Tous_les_Comptes ;
   // Affichage du formulaire de saisie
   ?>
   <br/>
   <form action="MySQL_PDO_transaction_secure_prepare_web.php"
method="post">
   <fieldset>
   <legend>Saisissez les informations du virement
:</legend><br/>
   Num&eacute;ro (ID_Cpt) du compte &agrave;
<b>cr&eacute;diter</b> : <input type="text"
name="Num_Cpt_Debit" size="3" maxlength="3" required
pattern="[1-9][0-9]{0,2}" placeholder="7"
autofocus/><br/><br/>
   Num&eacute;ro (ID_Cpt) du compte &agrave;
<b>d&eacute;biter</b>  : <input type="text"
name="Num_Cpt_Credit" size="3" maxlength="3" required
pattern="[1-9][0-9]{0,2}" placeholder="12" /><br/><br/>
   <b>Montant du virement</b>  : <input type="text"
name="MtVir" size="8" maxlength="8" placeholder="185,33"
required pattern="[1-9][0-9\.\,]{0,7}" /> &euro;<br /><br />
   <input type="submit" name="info_virement" value="Effectuer
le virement" />
   <input type="reset" value="Effacer le formulaire" />
   </fieldset>
   </form>
   <?php
  }
 }
 ?>
 </body>
</html>
```

Voici le fichier `MySQL_PDO_transaction_include_sprog_commun_web.php`, contenant les différentes fonctions utilisées dans le programme.

Listing 10-1.13 : fichier MySQL_PDO_transaction_include_sprog_commun_web.php

```
<?php
define("WEB_EOL","<br/>");
// ======================================================
// -- Fonction d'affichage de l'état de tous les comptes --
// ======================================================
function Affiche_Etat_Comptes($texte)
{
```

```
 global
$ERR_TRAIT,$TYPE_DBB,$SERVEUR,$BASEDD,$LOGIN_ADM,$MDP_ADM,$TAB
LECOMPTES,$TABLECLIENTS;
 try {
  // -- Contexte pour le message d'erreur --
  $contexte="Connexion base de donn&eacute;es";
  // -- Connexion de la base de donn&eacute;es --
  $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
  $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
  // -- Définition du codage en UTF8 --
  $bdd->exec("SET CHARACTER SET utf8");
  // -- Initialisation des Exceptions PDO pour prepare --
  $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  // Limitation aux comptes de dépôt
  $type_compte="Compte_Dépôts";
  // On affiche tous les comptes courant Avant le virement
  // -- Contexte pour le message d'erreur --
  $contexte="Probl&egrave;me de requ&ecirc;te";
  // -- Préparation de la requête --
  $requete_sql='SELECT
cb.ID_Cpt,cb.Agence,cb.Numero,cb.Type,cl.Nom,cl.Prenom,ROUND(c
b.Solde,2) Solde_Compte FROM '.$TABLECOMPTES.' cb INNER JOIN
'.$TABLECLIENTS.' cl ON cb.ID_Clt=cl.ID_Clt WHERE
Type=:type_compte';
  $RequetePreparee = $bdd->prepare($requete_sql);
  // -- Liaison avec les paramètres --
  $RequetePreparee->bindParam(':type_compte', $type_compte);
  // -- Exécution de la requête --
  $RequetePreparee->execute();
  // -- Retourne un tableau associatif --
  $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
  // -- Boucle de traitement de chaque client --
  $Tab_Comptes=$RequetePreparee->fetchAll();
  // -- Conversion de la colonne solde au format français --
  foreach ($Tab_Comptes as $Num => $un_cpt)
  {

$un_cpt['Solde_Compte']=number_format($un_cpt['Solde_Compte'],
2,","," ")." €";
   $Tab_Comptes[$Num]=$un_cpt;
  }
  // -- Affichage des données retournées --
  Affichage_Liste_Comptes($texte,$Tab_Comptes);
  // -- Fermeture de la requête --
  $RequetePreparee->closeCursor();
  return $Tab_Comptes;
```

```
 }
 catch(Exception $e) {
  $TitreMessage = $contexte       ;
  $TexteMessage = $e->getMessage();
  Affiche_Message_Erreur($TitreMessage,$TexteMessage);
  $ERR_TRAIT=true;
 }
}
// ======================================================
// -- Fonction outil d'affichage du message d'erreur  --
// ======================================================
function Affiche_Message_Erreur($titre,$message)
{
 ?>
 <fieldset>
 <legend><?php echo $titre ?></legend><br/>
 <b><?php echo $message ?></b><br />
 </fieldset>
 <?php
}
// ========================================================
// -- Fonction outil d'affichage d'un tableau de comptes  --
// ========================================================
function Affichage_Liste_Comptes($titre,$tcomptes)
{
 // -- Affichage entête du tableau --
 reset($tcomptes);
 $un_compte=current($tcomptes);
 $liste_champs=array_keys($un_compte);
 ?>
 <table summary="Tableau de r&eacute;sultat">
  <caption><?php echo $titre;?></caption>
  <thead>
  <tr>
  <!-- entête du tableau -->
  <?php
  echo "<tr>";
  $nbchamps=0;
  foreach($liste_champs as $nom_champ)
  {
   echo "<th>$nom_champ</th>";
   $nbchamps++;
  }
  echo "</tr>";
  // -- Affichage des lignes du tableau --
  if (count($tcomptes) ==0)
  {
```

```
   echo "<td colspan=\"$nbchamps\"><b>Aucun compte &agrave;
afficher</b></td>";
  }
  else
  {
   foreach ($tcomptes as $indice => $un_compte)
   {
    echo "<tr>";
    // importation des variables à partir de l'étiquette des
champs
    extract($un_compte,EXTR_OVERWRITE);
    echo "<tr>";
    foreach($liste_champs as $nom_champ)
    {
     $val=$$nom_champ;
     echo "<td>$val</td>";
    }
    echo "</tr>";
   }
  }
  ?>
 </table>
 <?php
}
// ========================================================
// -- Fonction d'information sur l'état d'un seul compte --
// ========================================================
function Info_Compte($compte)
{
 global
$ERR_TRAIT,$TYPE_DBB,$SERVEUR,$BASEDD,$LOGIN_ADM,$MDP_ADM,$TAB
LECOMPTES,$TABLECLIENTS;
 try {
  // -- Contexte pour le message d'erreur --
  $contexte="Connexion base de donn&eacute;es";
  // -- Connexion de la base de donn&eacute;es --
  $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
  $MDP_ADM,array(PDO::ATTR_PERSISTENT => true));
  // -- Définition du codage en UTF8 --
  $bdd->exec("SET CHARACTER SET utf8");
  // -- Initialisation des Exceptions PDO pour prepare --
  $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  $contexte="Probl&egrave;me de requ&ecirc;te sur la table";
  // -- Préparation de la requête --
  $requete_sql='SELECT
cb.ID_Cpt,cb.Agence,cb.Numero,cb.Type,cl.Nom,cl.Prenom,ROUND(c
```

```
b.Solde,2) Solde_Compte FROM '.$TABLECOMPTES.' cb INNER JOIN
'.$TABLECLIENTS.' cl ON cb.ID_Clt=cl.ID_Clt WHERE
ID_Cpt=:compte';
  $RequetePreparee = $bdd->prepare($requete_sql);
  // -- Liaison avec les paramètres --
  $RequetePreparee->bindParam(':compte', $compte,
PDO::PARAM_INT);
  // -- Exécution de la requête --
  $RequetePreparee->execute();
  // -- Retourne un tableau associatif --
  $RequetePreparee->setFetchMode(PDO::FETCH_ASSOC);
  // -- Boucle de traitement de chaque client --
  $Tab_Infos_Cpt=$RequetePreparee->fetchAll();
  // -- Conversion de la colonne solde au format français --
  foreach ($Tab_Infos_Cpt as $Num => $un_cpt)
  {

$un_cpt['Solde_Compte']=number_format($un_cpt['Solde_Compte'],
2,","," ")." €";
   $Tab_Infos_Cpt[$Num]=$un_cpt;
  }
  $Tab_Infos_Un_Cpt=$Tab_Infos_Cpt[0];
  return $Tab_Infos_Un_Cpt;
 }
 catch(Exception $e) {
  $TitreMessage = $contexte       ;
  $TexteMessage = $e->getMessage();
  Affiche_Message_Erreur($TitreMessage,$TexteMessage);
  $ERR_TRAIT=true;
 }
}
// ==========================
// -- Fonction de virement --
// ==========================
function Virement($NumCptDebit,$NumCptCredit,$MtVirt)
{
 global
$ERR_TRAIT,$TYPE_DBB,$SERVEUR,$BASEDD,$LOGIN_ADM,$MDP_ADM,$TAB
LECOMPTES,$TABLECLIENTS;
 $virement_effectue   = true ;
 $ConnexionBDD        = false;
 $TransactionDemarree = false;
 try {
  // -- Contexte pour le message d'erreur --
  $contexte="Connexion base de donn&eacute;es";
  // -- Connexion de la base de donn&eacute;es --
  $bdd = new
PDO($TYPE_DBB.":host=".$SERVEUR.";dbname=".$BASEDD,$LOGIN_ADM,
  $MDP_ADM, array(PDO::ATTR_PERSISTENT => true));
```

```
  $ConnexionBDD=true;
  // -- Initialisation des Exceptions PDO pour prepare --
  $bdd->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  // -- Définition du codage en UTF8 --
  $bdd->exec("SET CHARACTER SET utf8");
  // -- On gère le virement  dans une transaction SQL --
  // ==============================
  // == On débute la transaction ==
  // ==============================
  // -- Contexte pour le message d'erreur --
  $contexte="Initialisation virement";
  $bdd->beginTransaction() ;
  $TransactionDemarree=true;
  // ========================
  // == On débite le compte ==
  // ========================
  // -- Contexte pour le message d'erreur --
  $contexte="D&eacute;bit du compte" ;
  // -- Préparation de la requête --
  $requete_sql='UPDATE '.$TABLECOMPTES.' SET
Solde=Solde+:MontantVir WHERE Id_Cpt=:NumCptOperation';
  $reponse = $bdd->prepare($requete_sql);
  // -- Liaison avec les paramètres --
  $reponse->bindParam(':MontantVir', $MontantVir);
  $reponse->bindParam(':NumCptOperation', $NumCptOperation,
PDO::PARAM_INT);
  // -- Affectation des valeurs pour les paramètres --
  $MontantVir      = -$MtVirt     ;
  $NumCptOperation = $NumCptDebit ;
  // -- Exécution de la requête --
  $reponse->execute()    ;
  // -- Fermeture de la requête --
  $reponse->closeCursor();
  // ==========================
  // == On crédite le compte ==
  // ==========================
  // -- Contexte pour le message d'erreur
  $contexte="Cr&eacute;dit du compte";
  // La requête est déjà préparée, elle ne change pas
  // les paramètres sont déjà liés (bind) ils ne changent pas
  // Affectation des valeurs pour les paramètres
  $MontantVir      = +$MtVirt      ;
  $NumCptOperation = $NumCptCredit ;
  // -- Exécution de la requête --
  $reponse->execute()    ;
  // -- Fermeture de la requête --
  $reponse->closeCursor();
```

```
  // ==============================
  // == On valide la transaction ==
  // ==============================
  // -- Si aucune erreur, on valide la transaction --
  $contexte = "Validation virement";
  $reponse  = $bdd->commit();
 }
 catch(Exception $e) {
  // ==============================
  // == On annule la transaction ==
  // ==============================
  if ($ConnexionBDD && $TransactionDemarree)
  {
   try {
    $bdd->rollback();
   }
   catch(Exception $er) {
    $TitreMessage = 'Probl&egrave;me sur le virement -
Transaction annul&eacute;e';
    $TexteMessage = 'Annulation : '.$er->getMessage();
    Affiche_Message_Erreur($TitreMessage,$TexteMessage);
    $virement_effectue=false;
   }
  }
  $TitreMessage = 'Probl&egrave;me sur le virement -
Transaction annul&eacute;e';
  $TexteMessage = $contexte.' : '.$e->getMessage();
  Affiche_Message_Erreur($TitreMessage,$TexteMessage);
  $virement_effectue=false;
 }
 return $virement_effectue;
}
?>
```

Voici le fichier `MySQL_PDO_transaction_include_param_dbb.php` :

Listing 10-1.14 : fichier MySQL_PDO_transaction_include_param_dbb.php

```
<?php
// -- Paramètres de connexion à la base de données --
$TYPE_DBB="mysql";
$SERVEUR="localhost";
$BASEDD="CoursPHP";
$TABLECOMPTES="comptes_bancaires";
$TABLECLIENTS="clients_bancaires";
$LOGIN_ADM="root";
$MDP_ADM="xxxx";
```

```
?>
```

Remarque

Il est préférable d'utiliser un compte MySQL autre que root pour effectuer cette transaction. Ce compte doit avoir le droit de consulter les données sur les tables « comptes_bancaires » et « clients_bancaires » (SELECT) et le droit de modifier les données de la table « comptes_bancaires (UPDATE). Le texte « xxxx » doit être remplacé par le vrai mot de passe ou bien par '' si aucun mot de passe n'est affecté.

Pour le web avec des listes déroulantes

Le programme `MySQL_PDO_formulaire_ajax_prepare_web.php` est une variation du programme précédent utilisant les requêtes préparées.

Il propose une interface de saisie beaucoup plus conviviale. Il utilise les listes déroulantes pour saisir successivement l'agence, le client, et le compte à débiter, ainsi que l'agence, le client et le compte à créditer.

Chaque liste déroulante est alimentée par les résultats d'une requête SQL. Le contenu de la liste suivante dépend de la sélection effectuée dans la liste précédente. Ainsi si l'agence bancaires « A » est choisie, seuls les clients de cette agence seront proposés dans la liste suivante. Si parmi cette liste le client « C » est sélectionné, seuls les comptes de ce client seront proposés dans la liste suivante.

Ce programme utilise le langage JavaScript Ajax.

Il utilise les programmes suivants :

- `traitement_clients_credit_prepare.php` : « include » qui génère la liste déroulante des clients et effectue la sélection du client à créditer ;
- `traitement_clients_debit_prepare.php` : « include » qui génère la liste déroulante des clients et effectue la sélection du client à débiter ;
- `traitement_comptes_credit_prepare.php` : « include » qui génère la liste déroulante des comptes et effectue la sélection du compte à créditer ;
- `traitement_comptes_debit_prepare.php` : « include » qui génère la liste déroulante des comptes et effectue la sélection du compte à débiter ;
- `MySQL_PDO_transaction_secure_prepare_ajax_web.php` : « include » qui effectue le virement ;
- `MySQL_PDO_fonctions_ajax_prepare.js` : « include » contenant les fonctions JavaScript Ajax ;

Nous ne présentons pas ici ce programme qui est téléchargeable sur le site de l'éditeur, mais des copies d'écran de son exécution.

La figure 10-1.23 présente le premier écran de l'interface sans aucune saisie. Seules les agences pour le compte de débit et de crédit apparaissent.

Figure 10-1.23
Virement : Liste déroulante-Ecran-1

La liste déroulante de l'agence a été alimentée par une requête SQL sur la table « agences_bancaires » (figure 10-1.24).

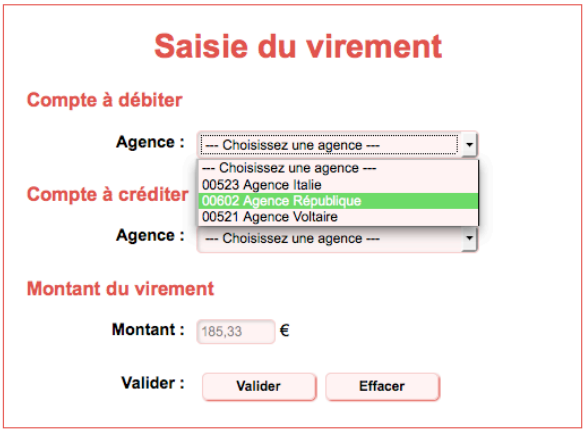


Figure 10-1.24
Virement : Liste déroulante-Ecran-2

Dès la sélection d'une agence la liste déroulante du client apparaît (figure 10-1.25).

Figure 10-1.25
Virement : Liste déroulante-Ecran-3

Elle a été alimentée par une jointure interne sur les tables « comptes_bancaires » et « clients_bancaires » afin de faire apparaître les noms et les prénoms des clients ayant un compte dans cette agence (Figure 10-1.26).

Figure 10-1.26
Virement : Liste déroulante-Ecran-4

La tentative de validation en cours de sélection fait apparaître un message d'erreur reprenant les valeurs saisies (Figure 10-1.27).

Figure 10-1.27
Virement : Liste déroulante-Ecran-5

La liste déroulante des comptes a été alimentée par une requête SQL sur les comptes de ce client dans cette agence. Seuls les comptes pouvant supporter un virement sont affichés, ce qui exclu les cartes de paiement et autres qui sont adossées à un compte courant (Figure 10-1.28).

Figure 10-1.28

Virement : Liste déroulante-Ecran-6

L'écran suivant montre l'ensemble des données renseignées (Figure 10-1.29).

Figure 10-1.29
Virement : Liste déroulante-Ecran-7

La validation affiche un écran de récapitulation et de confirmation (Figure 10-1.30).

Etat des comptes AVANT virement : Virement de 50,98 €, du compte 1 -> le compte 39

ID_Cpt	Agence	Numero	Type	Nom	Prenom	Solde_Compte
1	00602	165143P	Compte_Dépôts	DUPONT	JEAN	450,98 €
39	00521	045123P	Compte_Dépôts	DE-LA-RUE	LAURENCE	175,70 €

Confirmation du virement

Merci de confirmer le virement :
Oui Non

Confirmer

Figure 10-1.30
Virement : Liste déroulante-Ecran-8

Une fois confirmée, le virement est effectué en mode transactionnel, et le résultat est affiché (Figure 10-1.31).

Résultat : Virement de 50,98 €, du compte 1 -> le compte 39

ID_Cpt	Agence	Numero	Type	Nom	Prenom	Solde_Compte
1	00602	165143P	Compte_Dépôts	DUPONT	JEAN	400,00 €
39	00521	045123P	Compte_Dépôts	DE-LA-RUE	LAURENCE	226,68 €

Figure 10-1.31
Virement : Liste déroulante-Ecran-9

Le programme `MySQL_PDO_formulaire_ajax_query_web.php` est une variation du programme précédent utilisant la méthode « query » à la place des requêtes préparées. Il utilise les programmes suivants :

- `traitement_clients_credit_query.php` : « include » qui génère la liste déroulante des clients et effectue la sélection du client à créditer ;
- `traitement_clients_debit_query.php` : « include » qui génère la liste déroulante des clients et effectue la sélection du client à débiter ;
- `traitement_comptes_credit_query.php` : « include » qui génère la liste déroulante des comptes et effectue la sélection du compte à créditer ;
- `traitement_comptes_debit_query.php` : « include » qui génère la liste déroulante des comptes et effectue la sélection du compte à débiter ;
- `MySQL_PDO_transaction_secure_prepare_ajax_web.php` : « include » qui effectue le virement ;
- `MySQL_PDO_fonctions_ajax_query.js` : « include » contenant les fonctions JavaScript Ajax ;

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W