

# Stocker des images dans MySQL

Par BiDOuille 

Date de publication : 1 juillet 2005

Dernière mise à jour : 23 mars 2016


Ce tutoriel va vous apprendre à gérer vos images dans MySql.

Pour réagir à ce tutoriel, un espace de dialogue vous est proposé sur le forum : **[Commentez](#)**

I - Introduction.....	3
II - Pour ou contre ?.....	3
III - Côté MySQL.....	4
III-A - Un mot sur le BLOB.....	4
III-B - La table des images.....	4
IV - Côté PHP.....	5
IV-A - L'envoi de l'image.....	5
IV-B - L'enregistrement de l'image.....	6
IV-C - L'affichage de l'image.....	10
V - Conclusion.....	11
VI - Remerciements.....	12

## I - Introduction

Ce manuel est ici pour vous apprendre à **gérer des images stockées dans MySQL depuis PHP** et non pas à débiter en PHP. Je pars donc du principe que vous n'êtes pas un néophyte et que vous connaissez déjà les bases de la programmation dans ce langage.

 *Bien que PHP 5 soit la dernière version disponible, je n'aborde dans ce manuel que la syntaxe et les possibilités de PHP 4. J'ai seulement annoté par endroit certaines évolutions qui seraient éventuellement à signaler.*

Pour la rédaction des exemples de code, j'utilise donc au minimum PHP dans sa version **4.3.0** (qui est d'ailleurs déjà ancienne). Ceci est notamment valable pour la syntaxe des fonctions et surtout des variables superglobales. Si vous développez avec une version antérieure, je vous conseille de vous reporter sur la documentation officielle pour la correspondance des fonctions.

Pour la programmation, je me sers du célèbre **EasyPHP** dans sa version **1.7** (disponible gratuitement sur [www.easyphp.org](http://www.easyphp.org)). Je ne détaille pas la procédure d'installation et vous renvoie sur leur site pour ces détails.

Pour le SGBD, j'utilise **MySQL** dans sa version 4.0.15 qui est fournie avec **EasyPHP**.

J'ai cependant testé les exemples avec une version 3.21 et cela n'a posé aucun problème particulier.

Ce manuel est en partie inspiré du travail de **Kevin Waterson**.

## II - Pour ou contre ?

Si vous utilisez **MySQL** (ou un autre SGBD), vous avez bien sûr à y stocker des données de type texte, mais également à gérer des données binaires comme des images (JPEG, GIF, PNG) ou des fichiers tels que des PDF, de la bureautique ou des ZIP.

C'est une question que vous allez sûrement vous poser : « *faut-il stocker tous ces fichiers directement dans la base ?* »

La question n'est alors pas tranchée. Elle ne le sera d'ailleurs jamais, car tout dépend du contexte d'utilisation.

Si vous n'utilisez pas ce concept, vous devez alors utiliser un système de répertoires pour y stocker vos fichiers. Vous n'enregistrez dans votre base que le chemin et le nom vers ces derniers. C'est un simple champ texte dans une table de type **VARCHAR**.

Le principe du stockage directement dans la base a un certain nombre d'avantages :

- la gestion de l'envoi des fichiers est beaucoup plus simple. Il n'y a pas de problème de droits en écriture notamment sous plateforme Linux. Tout est centralisé en un seul point ;
- la sauvegarde des données est grandement facilitée. Il vous suffit de faire un « *DUMP* » de la base et vous n'avez pas besoin de récupérer en plus les fichiers disséminés sur le disque ;
- il n'y a plus de risque d'incohérence entre un fichier stocké sur le disque et son chemin enregistré dans la base. L'intégrité est donc toujours respectée ;
- vos fichiers sont protégés, car on ne peut plus y accéder directement par une URL.

Cependant, il y a aussi des inconvénients à ce système :

- le stockage risque de faire « gonfler » rapidement la taille de la base. Il faudra prévoir une partition suffisante au départ ;
- il y a un risque de surcharge et de ralentissement de la base en cas de gros fichiers. L'accès aux fichiers implique en effet de faire obligatoirement un **SELECT** ;

- il sera impossible d'accéder aux fichiers directement par URL. Il n'y aura donc aucune possibilité de visualisation directe dans les répertoires.

Il vous faudra être prudent lors de vos envois de requête SQL :

```
1. SELECT * FROM `images`
```

Cette sélection peut s'avérer risquée en termes de performance si plus tard la table comprend de nombreuses images.

Il existe des bancs d'essai pour vous montrer les différences de performance de trois méthodes d'accès pour une image :

- l'accès à l'image depuis une simple balise <a href...> ;
- le stockage dans un répertoire dédié et l'enregistrement du chemin dans la base ;
- le stockage directement dans la base.

Les résultats sont surprenants.

C'est maintenant à vous de voir quelle méthode vous allez retenir.

### III - Côté MySQL

Vous avez fait le choix d'enregistrer vos fichiers binaires directement dans la base. Nous allons maintenant voir ce qui se passe dans MySQL pour arriver à cela.

#### III-A - Un mot sur le BLOB

Le stockage d'un fichier binaire dans MySQL se fait dans un champ particulier dont le type est BLOB. BLOB est l'abréviation anglaise de Binary Large Object. MySQL propose quatre types de BLOB :


- BLOB ;
- TINYBLOB ;
- MEDIUMBLOB ;
- LONGBLOB.

Selon le type de fichier à enregistrer, vous devez choisir le bon BLOB. Reportez-vous sur la section *ad hoc* du manuel d'aide de MySQL pour plus d'information sur ce sujet.

#### III-B - La table des images

La première étape consiste bien sûr à créer une table pour stocker nos images à l'intérieur :

```
1. CREATE TABLE `images` (  
2.   `img_id` INT NOT NULL AUTO_INCREMENT ,  
3.   `img_nom` VARCHAR( 50 ) NOT NULL ,  
4.   `img_taille` VARCHAR( 25 ) NOT NULL ,  
5.   `img_type` VARCHAR( 25 ) NOT NULL ,  
6.   `img_desc` VARCHAR( 100 ) NOT NULL ,  
7.   `img_blob` BLOB NOT NULL ,  
8.   PRIMARY KEY ( `img_id` )  
9. )
```

 *Côté MySQL, je vous recommande l'utilisation de l'outil PhpMyAdmin notamment pour la création de la table.*

Cela va nous donner :

Champ	Type [Documentation]	Taille Valeurs*	Attribut*	Extra	Primaire	Index	Unique	...	Texte entier
img_id	INT			auto_increment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_nom	VARCHAR	50			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
img_taille	VARCHAR	25			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
img_type	VARCHAR	25			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
img_desc	VARCHAR	100			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
img_blob	BLOB				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Voici quelques précisions sur la structure de la table créée :

- nous avons un identifiant unique (img\_id) auto-incrémenté et indexé en tant que clé primaire. Ceci afin de ne pas avoir de risque de confusion lors des sélections ;
- nous gardons précieusement le nom de l'image d'origine avec img\_nom ;
- l'information de la taille (img\_taille) peut être importante. Il faut éviter le chargement intempestif d'image dont le volume est excessif ;
- le type de l'image (img\_type) sera récupéré via PHP par le type MIME ;
- une petite description de l'image (img\_desc) afin d'avoir un résumé ou des mots-clés ;
- enfin, le contenu binaire de l'image qui sera stocké dans img\_blob.

**i** Pour améliorer le classement des images, nous aurions pu ajouter un champ catégorie et une seconde table liée.

## IV - Côté PHP

Nous allons maintenant passer du côté PHP et voir la programmation de notre interface pour la gestion de nos images.

Il s'agit dans l'ordre d'envoyer l'image depuis le client, de récupérer celle-ci sur le serveur et de l'enregistrer.

Enfin, pour vérifier que tout s'est bien passé, nous allons afficher les images enregistrées dans la table.

Pour les besoins de cet exemple, créez un répertoire « blob » dans lequel vous enregistrerez vos scripts.

### IV-A - L'envoi de l'image

L'envoi de l'image (ou *upload*) se fait tout simplement par un formulaire HTML. Celui-ci permet l'envoi vers le serveur web de n'importe quel fichier du disque du client grâce au champ `<input type="file">`.

Enregistrons notre script en tant que **index.php** :

```

1. <html>
2.   <head>
3.     <title>Stock d'images</title>
4.   </head>
5.   <body>
6.     <h3>Envoi d'une image</h3>
7.     <form enctype="multipart/form-data" action="#" method="post">
8.       <input type="hidden" name="MAX_FILE_SIZE" value="250000" />
9.       <input type="file" name="fic" size=50 />
10.      <input type="submit" value="Envoyer" />
11.    </form>
12.  </body>
13. </html>

```

Il est très important de ne pas oublier le paramètre **enctype="multipart/form-data"** afin que le navigateur du client transfère les données binaires correctement.

Le champ caché avec le paramètre **MAX\_FILE\_SIZE** permet d'empêcher au niveau du client, le transfert de fichier supérieur à 250 000 octets. Il est fortement recommandé de vérifier via **PHP** la taille du fichier.

Exécutez maintenant le formulaire depuis votre navigateur favori. Cela va nous donner :



Bien sûr, cela ne fonctionne pas encore. Passons à l'étape suivante...

## IV-B - L'enregistrement de l'image

Nous allons programmer la réception du fichier envoyé depuis le formulaire en créant une fonction pour enregistrer dans la base le contenu binaire et les informations de taille, de type et de nom.

Dans l'ordre les étapes seront les suivantes :

- 1 Vérification que le fichier est bien sur le serveur ;
- 2 Vérification que l'image ne dépasse pas la taille maximum fixée ;
- 3 Connexion à la base ;
- 4 Formater le contenu binaire pour l'insertion ;
- 5 Insertion des données dans la table.

**i** Nous allons utiliser la **superglobale \$\_FILES** pour récupérer le contenu binaire. Reportez-vous sur l'aide de **PHP** pour l'explication des différentes syntaxes.

Nous allons maintenant créer un script que nous nommerons **transfert.php** pour réaliser les étapes 1 et 2 :


```
1. <?php
2.     function transfert(){
3.         $ret      = false;
4.         $img_blob  = '';
5.         $img_taille = 0;
6.         $img_type  = '';
7.         $img_nom   = '';
8.         $taille_max = 250000;
```

```

9.         $ret          = is_uploaded_file($_FILES['fic']['tmp_name']);
10.
11.         if (!$ret) {
12.             echo "Problème de transfert";
13.             return false;
14.         } else {
15.             // Le fichier a bien été reçu
16.             $img_taille = $_FILES['fic']['size'];
17.
18.             if ($img_taille > $taille_max) {
19.                 echo "Trop gros !";
20.                 return false;
21.             }
22.
23.             $img_type = $_FILES['fic']['type'];
24.             $img_nom  = $_FILES['fic']['name'];
25.         }
26.     }
27. ?>

```

Nous avons créé une fonction et pour le moment, nous n'avons fait que mettre les variables à jour, grâce au contenu du tableau **\$\_FILES**.

 *Pensez toujours à vérifier ce qui est envoyé côté PHP. Ici, nous testons à nouveau la taille du fichier précédemment fixée dans le formulaire HTML.*


Nous allons maintenant réaliser l'étape 3 pour établir la connexion à notre base **MySQL**.

Pour ce faire nous allons enregistrer un script **connexion.php** :

```

1. <?php
2.     $hote = 'localhost';
3.     $base = 'test';
4.     $user = 'root';
5.     $pass = '';
6.     $cnx = mysql_connect($hote, $user, $pass) or die(mysql_error());
7.     $ret = mysql_select_db($base) or die(mysql_error());
8. ?>

```

 *Il est indispensable et très pratique d'utiliser la fonction **mysql\_error** afin de pouvoir visualiser au mieux les problèmes de connexion.*

À vous de changer les valeurs des variables pour la connexion..)

Nous allons inclure ce script dans **transfert.php** :

```

1. <?php
2.     function transfert () {
3.         ...
4.         $img_type = $_FILES['fic']['type'];
5.         $img_nom  = $_FILES['fic']['name'];
6.         include ("connexion.php");
7.     }
8. ?>

```

Nous voici arrivés à l'étape 4 afin de formater le fichier en vue de son insertion dans la base.

Pour le moment, le fichier est stocké physiquement dans un répertoire temporaire du serveur. Il s'agit donc de récupérer le contenu binaire dans une variable.

Or, on ne peut pas passer le contenu d'un fichier directement dans une variable. Pour exécuter cette opération, nous allons utiliser la fonction **file\_get\_contents** :

```
<?php
function transfert () {
    ...
    include ("connexion.php");
    $img_blob = file_get_contents ($_FILES['fic']['tmp_name']);
}
?>
```

**i** Je vous rappelle que la fonction **file\_get\_contents** n'est disponible qu'à partir de la version **4.3.0 de PHP**.

Enfin, nous allons réaliser l'étape 5 pour enregistrer dans la base MySQL le contenu des informations :

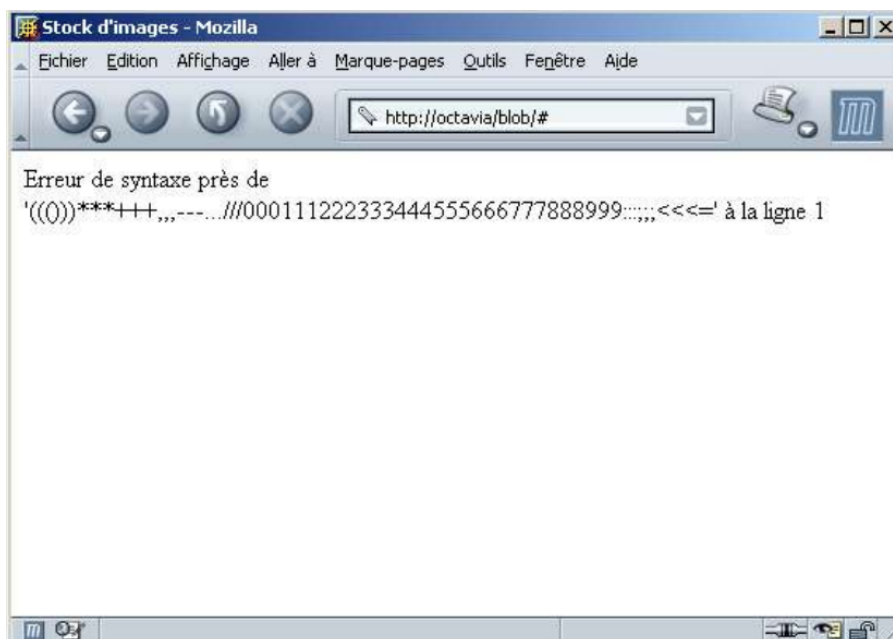
```
1. <?php
2.     function transfert () {
3.         ...
4.         include ("connexion.php");
5.         $img_blob = file_get_contents ($_FILES['fic']['tmp_name']);
6.         $req = "INSERT INTO images ( " .
7.                 "img_nom, img_taille, img_type, img_blob " .
8.                 ") VALUES ( " .
9.                 "'" . $img_nom . "', " .
10.                "'" . $img_taille . "', " .
11.                "'" . $img_type . "', " .
12.                "'" . $img_blob . "'" ) ";
13.         $ret = mysql_query ($req) or die (mysql_error ());
14.         return true;
15.     }
16. ?>
```

Terminons tout cela en insérant notre script de fonction de transfert dans notre premier script **index.php** :

```
1. <html>
2.     <head>
3.         <title>Stock d'images</title>
4.     </head>
5.     <body>
6.         <?php
7.             include ("transfert.php");
8.             if ( isset($_FILES['fic']) )
9.             {
10.                transfert();
11.            }
12.        ?>
13.        <h3>Envoi d'une image</h3>
14.        <form enctype="multipart/form-data" action="#" method="post">
15.            <input type="hidden" name="MAX_FILE_SIZE" value="250000" />
16.            <input type="file" name="fic" size=50 />
17.            <input type="submit" value="Envoyer" />
18.        </form>
19.    </body>
20. </html>
```

Si vous testez le code, ce dernier ne fonctionne pas et vous aurez à peu de chose près ceci :





Nous avons en effet, oublié d'échapper les caractères spéciaux que peut contenir un fichier binaire. Reprenons notre script **transfert.php** et modifions le texte de la requête :

```

1. <?php
2.     function transfert () {
3.         ...
4.         include ("connexion.php");
5.         $img_blob = file_get_contents ($_FILES['fic']['tmp_name']);
6.         $req = "INSERT INTO images ("
7.                 "img_nom, img_taille, img_type, img_blob "
8.             ") VALUES ("
9.                 "'" . $img_nom . "', "
10.                "'" . $img_taille . "', "
11.                "'" . $img_type . "', "
12.                "'" . addslashes
($img_blob) . "'" ); // N'oublions pas d'échapper le contenu binaire
13.         $ret = mysql_query ($req) or die (mysql_error ());
14.         return true;
15.     }
16. ?>

```

Retentez l'opération. Cette fois, aucune erreur n'apparaît et le formulaire s'affiche à nouveau.

Il nous reste à vérifier que tout s'est bien passé et que l'image est bien dans la base de données. Pour cela, reprenez votre script **index.php** et ajoutez à la fin une ancre pour l'appel à un nouveau script : **liste.php** :

```

1. <html>
2.     <head>
3.         <title>Stock d'images</title>
4.     </head>
5.     <body>
6.         <?php
7.             include ("transfert.php");
8.             if ( isset($_FILES['fic']) )
9.             {
10.                 transfert();
11.             }
12.         ?>
13.         <h3>Envoi d'une image</h3>
14.         <form enctype="multipart/form-data" action="#" method="post">
15.             <input type="hidden" name="MAX_FILE_SIZE" value="250000" />
16.             <input type="file" name="fic" size=50 />
17.             <input type="submit" value="Envoyer" />

```

```
18.     </form>
19.     <p><a href="liste.php">Liste</a></p>
20.     </body>
21. </html>
```

## IV-C - L'affichage de l'image

Nous allons d'abord créer un script pour lister les images contenues dans la table de la base **MySQL**. Nous appellerons ce script **liste.php** :

```
1. <html>
2.   <head>
3.     <title>Stock d'images</title>
4.   </head>
5.   <body>
6.     <?php
7.       include ("connexion.php");
8.       $req = "SELECT img_nom, img_id " .
9.             "FROM images ORDER BY img_nom";
10.      $ret = mysql_query ($req) or die (mysql_error ());
11.      while ( $col = mysql_fetch_row ($ret) )
12.      {
13.        echo "<a href=\"aperçu.php?id=\" . $col[1] . \">\" . $col[0] . "</a><br />";
14.      }
15.    ?>
16.  </body>
17. </html>
```

Nous allons insérer à nouveau notre script pour la connexion afin de pouvoir accéder à la base. On envoie ensuite une requête de sélection pour obtenir toutes nos images classées alphabétiquement par nom. On boucle sur le résultat pour afficher un lien sur chaque image en passant l'identifiant en paramètre à un script **aperçu.php**.

Si vous avez enregistré quelques images, vous aurez le résultat suivant :



Il nous reste maintenant à afficher chaque image avec le script **aperçu.php** :

```
1. <?php
2.
3.   if ( isset($_GET['id']) ){
4.     $id = intval ($_GET['id']);
```

```

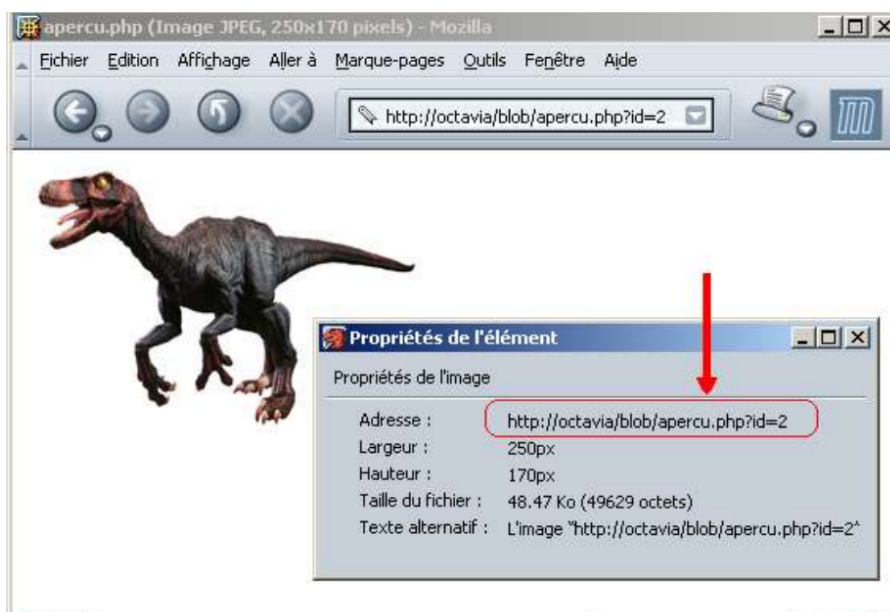
5.     include ("connexion.php");
6.     $req = "SELECT img_id, img_type, img_blob " .
7.           "FROM images WHERE img_id = " . $id;
8.     $ret = mysql_query ($req) or die (mysql_error ());
9.     $col = mysql_fetch_row ($ret);
10.
11.     if ( !$col[0] ){
12.         echo "Id d'image inconnu";
13.     } else {
14.         header ("Content-type: " . $col[1]);
15.         echo $col[2];
16.     }
17.
18. } else {
19.     echo "Mauvais id d'image";
20. }
21.
22. ?>

```

On teste d'abord la récupération de l'identifiant. On inclut ici encore, le script de connexion à la base, puis on envoie une requête de sélection sur l'identifiant soumis.

Si l'enregistrement correspondant à l'identifiant existe, on envoie alors l'entête « *Content-type* » (type de contenu) avec la fonction **header**. Il suffit alors d'afficher le contenu binaire de l'image.

Le test de ce script va nous permettre d'afficher l'image. C'est magique !



Une édition des propriétés de l'image nous montre qu'il n'y a pas d'URL directe sur cette dernière.

## V - Conclusion

Et voilà ! Avec ce support, nous venons de couvrir les bases pour la gestion d'images entre MySQL et PHP.

Pour plus d'informations sur ce sujet, vous pouvez lire l'excellente FAQ de MySQL disponible sur le site Developpez.com.

Je vous invite également sur mon site principal où vous retrouverez d'autres supports pour aller plus loin en PHP. Reportez-vous pour cela à l'adresse suivante : [www.beaussier.com](http://www.beaussier.com).

## VI - Remerciements

Nous tenons à remercier **Claude Leloup** pour sa relecture orthographique et **Malick SECK** pour la mise au gabarit.