

Blob fields in MySQL Databases

Publicado em: 08/05/2003

Há algum tempo trabalho com bases de dados padrão SQL voltados principalmente para o desenvolvimento de aplicações de Internet. Estes dias um colega me perguntou:

- Posso colocar imagens ou outros tipos de arquivos binários dentro de uma tabela num banco?

Sim, sem dúvida isso é possível. Mas como?

Na busca de algum tutorial ou algo do gênero, não encontrei nada que fosse de fácil entendimento e que realmente explicasse como colocar uma imagem ou um arquivo mp3, por exemplo, dentro de uma base de dados. Assim, cá está o tutorial sobre este assunto.

Observação: Não erre o nome do tutorial; é Blob Fields e não BOB Fields (os que conhecem a história de nosso país sabem do que estou falando).

O que é um campo blob?

O blob (Binary Large Object - grande objeto binário) é um campo criado para o armazenamento de qualquer tipo de informações em formato binário, dentro de uma tabela de um banco de dados. A primeira vista isto pode parecer interessantíssimo, mas gostaria que, antes de partir para esta solução, pesquise se realmente precisa dela. Não é difícil encontrar problemas como "buracos" no banco e lentidão de acesso a estas informações.

O MySQL trabalha com campos blob, que são na verdade campos texto (TEXT) com uma única diferença: campos texto são "case-insensitive", ao contrário dos blob's.

Os blob's são divididos em quatro tipos (no MySQL), sendo que a diferença existente de um para o outro é unicamente a capacidade de armazenamento e trabalho do campo. Estes são:

TINYBLOB - campo blob de armazenamento máximo igual a 255 caracteres (8 bits) mais 1 de controle;

BLOB - o mesmo que o Tinyblob, porém armazenando até 16535 caracteres (16 bits) mais 2 de controle;

MEDIUMBLOB - o mesmo que o tinyblob, porém armazenando até 16777216 caracteres (24 bits) mais 3 de controle;

LOB - o mesmo que o tinyblob, porém armazenando até 4294967295 caracteres (32 bits) mais 4 de controle.

Além disto, existem as seguintes particularidades com os campos blobs:

Não podem ser chaves primárias (excluindo Tinyblob);

Não é possível usar os comandos GROUP e SORT com campos blob;

São reconhecidos como um LONGVARCHAR para drivers ODBC.

Enfim, vamos ver como colocar arquivos binários dentro do banco.

Armazenando arquivos binários em tabelas no banco

Criaremos uma tabela para exemplo dentro do MySQL a fim que possamos fazer nossos testes. Esta tabela é bem simples e conta somente com dois campos, um que é a chave de registro e outro que é o campo onde os arquivos binários serão armazenados.

Nota: Não discuto aqui se você conhece ou não a linguagem SQL ou o banco MySQL. A premissa é que conheça.

Criando uma nova tabela dentro do banco.

Inicialmente vamos criar um novo banco e uma nova tabela para que possamos trabalhar.

Assim:

```
mysql> create database blobs;
Query OK, 1 row affected (0.09 sec)
```

Alteramos agora para o banco que acabamos de criar:

```
mysql> use blobs;
Database changed
```

Agora, criamos a tabela:

```
mysql> CREATE TABLE teste (Id_Blob INT(10) NOT NULL PRIMARY KEY,
Na_Imagem MEDIUMBLOB);
Query OK, 0 rows affected (0.23 sec)
```

Muito bem! Vamos dar uma olhada o que existe agora no banco blobs

```
mysql> SHOW TABLES;
+-----+
| Tables_in_blobs |
+-----+
| teste           |
+-----+
1 row in set (0.01 sec)
```

```
mysql> SHOW COLUMNS FROM teste;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Id_Blob    | int(10)       |      | PRI | 0        |       |
| Na_Imagem  | mediumblob    | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)
```

Veja que possuímos agora uma tabela com o nome de teste que contém dois campos, Id_Blob e outro Na_Imagem, onde serão armazenadas as nossas imagens. Feito o banco, vamos agora fazer as inserções de imagens.

```
mysql> INSERT INTO teste (Id_Blob,Na_Imagem) VALUES
(1,LOAD_FILE("/error.gif"));
Query OK, 1 row affected (0.07 sec)
```

Pronto, ai está. A imagem error.gif dentro da coluna Na_Imagem de nossa base de dados. Quer verificar? De uma olhada no arquivo de extensão MYD da base TESTE. Poderá ver que seu tamanho, ou é idêntico ao tamanho da imagem ou é muito próximo.

Não faça um SELECT do conteúdo do banco. Irá ver somente tracinhos e não a imagem :)

Comentando a linha de inserção

Existem duas coisas que mudam na instrução INSERT realizada com imagens; a primeira é o comando do MySQL LOAD_FILE que deve ser usado para carregar o arquivo do disco e colocá-lo na base, e a segunda é o caminho onde o arquivo se encontra. Por padrão, tanto em ambiente "X" como em Windows o caminho deve ser completo, entretanto no ambiente Windows, deve ser mantido o padrão "X" de barras. Se o arquivo está em c:\windows você deve especificar dentro do comando LOAD_FILE assim: c:/windows/nome_do_arquivo.

Enfim, nada muito difícil não é? Só deixe-me comentar duas ressalvas.

Para que possa inserir um arquivo binário dentro do banco, você precisa ter permissão de escrita e leitura no arquivo. Conheci alguns desenvolvedores que fizeram tudo certo e acabaram esquecendo deste detalhe. Quando foram testar, perderam horas para encontrar o problema.

Um outro detalhe que deve ser visto é que você somente conseguirá colocar arquivos na base que não excedam o valor da variável de ambiente max_allowed_packet. Caso o arquivo exceda este limite, ele não será gravado na base, retornando um NULL.

Agora que já inserimos a imagem dentro do banco, vamos ver como pegamos esta imagem para visualizar seu conteúdo.

Acessando os arquivos binários do banco de dados

Aqui existe a ressalva:

Será que é confiável? Que vale a pena mesmo?

Mas não estou aqui para discutir isso e sim como recuperar a informação, então vamos lá:

```
mysql> SELECT Na_Imagem INTO OUTFILE "/retorno.jpg" FROM teste WHERE
Id_Blob = 1;
Query OK, 1 row affected (0.05 sec)
```

Observe que o arquivo é literalmente recriado no disco por meio do comando INTO e do comando OUTFILE. Aqui também vale a regra da inserção; é necessário ter permissão de escrita no local onde o arquivo será criado e que não exista outro com o mesmo nome, no mesmo local.

Finalizando

A utilização de bancos de dados para o armazenamento de arquivos binários pode ser uma solução para alguns problemas de usuários ou de sistemas. Mas devem ser levados em consideração os seguintes pontos:

1 - Se você deseja armazenar arquivos de páginas web, o acesso fica mais lento pois somente poderá enviar a imagem para o cliente que solicitou a página, após a criação do arquivo dentro do disco. É mais fácil usar as tags HREF e/ou SRC para isto;

2 - Também, não existe cache de disco neste caso, ou seja, é melhor manter em cache num proxy as imagens para rápido acesso do que enviá-las cada vez que são solicitadas. Em contrapartida você não tem problemas de permissão pois, após configurar determinado diretório para escrita, é possível enviar todos os arquivos para o mesmo e estes herdarem as permissões.

Enfim, valer a pena ou não é motivo para um outro tutorial.

Paulino Michelazzo
paulino@michelazzo.com.br
<http://www.michelazzo.com.br>
Blob fields in MySQL Databases
Publicado em: 08/05/2003

Há algum tempo trabalho com bases de dados padrão SQL voltados principalmente para o

desevolvimento de aplicações de Internet. Estes dias um colega me perguntou:

- Posso colocar imagens ou outros tipos de arquivos binários dentro de uma tabela num banco?

Sim, sem dúvida isso é possível. Mas como?

Na busca de algum tutorial ou algo do gênero, não encontrei nada que fosse de fácil entendimento e que realmente explicasse como colocar uma imagem ou um arquivo mp3, por exemplo, dentro de uma base de dados. Assim, cá está o tutorial sobre este assunto.

Observação: Não erre o nome do tutorial; é Blob Fields e não BOB Fields (os que conhecem a história de nosso país sabem do que estou falando).

O que é um campo blob?

O blob (Binary Large Object - grande objeto binário) é um campo criado para o armazenamento de qualquer tipo de informações em formato binário, dentro de uma tabela de um banco de dados. A primeira vista isto pode parecer interessantíssimo, mas gostaria que, antes de partir para esta solução, pesquise se realmente precisa dela. Não é difícil encontrar problemas como "buracos" no banco e lentidão de acesso a estas informações.

O MySQL trabalha com campos blob, que são na verdade campos texto (TEXT) com uma única diferença: campos texto são "case-insensitive", ao contrário dos blob's.

Os blob's são divididos em quatro tipos (no MySQL), sendo que a diferença existente de um para o outro é unicamente a capacidade de armazenamento e trabalho do campo. Estes são:

TINYBLOB - campo blob de armazenamento máximo igual a 255 caracteres (8 bits) mais 1 de controle;

BLOB - o mesmo que o Tinyblob, porém armazenando até 16535 caracteres (16 bits) mais 2 de controle;

MEDIUMBLOB - o mesmo que o tinyblob, porém armazenando até 16777216 caracteres (24 bits) mais 3 de controle;

LOB - o mesmo que o tinyblob, porém armazenando até 4294967295 caracteres (32 bits) mais 4 de controle.

Além disto, existem as seguintes particularidades com os campos blobs:

Não podem ser chaves primárias (excluindo Tinyblob);

Não é possível usar os comandos GROUP e SORT com campos blob;

São reconhecidos como um LONGVARCHAR para drivers ODBC.

Enfim, vamos ver como colocar arquivos binários dentro do banco.

Armazenando arquivos binários em tabelas no banco

Criaremos uma tabela para exemplo dentro do MySQL a fim que possamos fazer nossos testes. Esta tabela é bem simples e conta somente com dois campos, um que é a chave de registro e outro que é o campo onde os arquivos binários serão armazenados.

Nota: Não discuto aqui se você conhece ou não a linguagem SQL ou o banco MySQL. A premissa é que conheça.

Criando uma nova tabela dentro do banco.

Inicialmente vamos criar um novo banco e uma nova tabela para que possamos trabalhar.

Assim:

```
mysql> create database blobs;
```

```
Query OK, 1 row affected (0.09 sec)
```

Alteramos agora para o banco que acabamos de criar:

```
mysql> use blobs;
```

```
Database changed
```

Agora, criamos a tabela:

```
mysql> CREATE TABLE teste (Id_Blob INT(10) NOT NULL PRIMARY KEY, Na_Imagem  
MEDIUMBLOB);
```

```
Query OK, 0 rows affected (0.23 sec)
```

Muito bem! Vamos dar uma olhada o que existe agora no banco blobs

```
mysql> SHOW TABLES;
```

```
+-----+
```

```
| Tables_in_blobs |
```

```
+-----+
```

```
| teste          |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

```
mysql> SHOW COLUMNS FROM teste;
```

```
+-----+-----+-----+-----+-----+-----+
```

```
| Field      | Type      | Null | Key | Default | Extra |
```

```
+-----+-----+-----+-----+-----+-----+
```

```
| Id_Blob   | int(10)   |      | PRI | 0       |      |
```

```
| Na_Imagem | mediumblob | YES  |     | NULL    |      |
```

```
+-----+-----+-----+-----+-----+-----+
```

```
2 rows in set (0.05 sec)
```

Veja que possuímos agora uma tabela com o nome de teste que contém dois campos, Id_Blob e outro Na_Imagem, onde serão armazenadas as nossas imagens.

Feito o banco, vamos agora fazer as inserções de imagens.

```
mysql> INSERT INTO teste (Id_Blob,Na_Imagem) VALUES (1,LOAD_FILE("/error.gif"));
```

Query OK, 1 row affected (0.07 sec)

Pronto, ai está. A imagem error.gif dentro da coluna Na_Imagem de nossa base de dados. Quer verificar? De uma olhada no arquivo de extensão MYD da base TESTE. Poderá ver que seu tamanho, ou é idêntico ao tamanho da imagem ou é muito próximo.

Não faça um SELECT do conteúdo do banco. Irá ver somente tracinhos e não a imagem :)

Comentando a linha de inserção

Existem duas coisas que mudam na instrução INSERT realizada com imagens; a primeira é o comando do MySQL LOAD_FILE que deve ser usado para carregar o arquivo do disco e colocá-lo na base, e a segunda é o caminho onde o arquivo se encontra. Por padrão, tanto em ambiente "X" como em Windows o caminho deve ser completo, entretanto no ambiente Windows, deve ser mantido o padrão "X" de barras. Se o arquivo está em c:\windows você deve especificar dentro do comando LOAD_FILE assim: c:/windows/nome_do_arquivo.

Enfim, nada muito difícil não é? Só deixe-me comentar duas ressalvas.

Para que possa inserir um arquivo binário dentro do banco, você precisa ter permissão de escrita e leitura no arquivo. Conheci alguns desenvolvedores que fizeram tudo certo e acabaram esquecendo deste detalhe. Quando foram testar, perderam horas para encontrar o problema.

Um outro detalhe que deve ser visto é que você somente conseguirá colocar arquivos na base que não excedam o valor da variável de ambiente max_allowed_packet. Caso o arquivo exceda este limite, ele não será gravado na base, retornando um NULL.

Agora que já inserimos a imagem dentro do banco, vamos ver como pegamos esta imagem para visualizar seu conteúdo.

Acessando os arquivos binários do banco de dados

Aqui existe a ressalva:

Será que é confiável? Que vale a pena mesmo?

Mas não estou aqui para discutir isso e sim como recuperar a informação, então vamos lá:

```
mysql> SELECT Na_Imagem INTO OUTFILE "/retorno.jpg" FROM teste WHERE Id_Blob = 1;
```

Query OK, 1 row affected (0.05 sec)

Observe que o arquivo é literalmente recriado no disco por meio do comando INTO e do comando OUTFILE. Aqui também vale a regra da inserção; é necessário ter permissão de escrita no local onde o arquivo será criado e que não exista outro com o mesmo nome, no mesmo local.

Finalizando

A utilização de bancos de dados para o armazenamento de arquivos binários pode ser uma solução para alguns problemas de usuários ou de sistemas. Mas devem ser levados em consideração os seguintes pontos:

1 - Se você deseja armazenar arquivos de páginas web, o acesso fica mais lento pois somente poderá enviar a imagem para o cliente que solicitou a página, após a criação do arquivo dentro do disco. É mais fácil usar as tags HREF e/ou SRC para isto;

2 - Também, não existe cache de disco neste caso, ou seja, é melhor manter em cache num proxy as imagens para rápido acesso do que enviá-las cada vez que são solicitadas. Em contrapartida você não tem problemas de permissão pois, após configurar determinado diretório para escrita, é possível enviar todos os arquivos para o mesmo e estes herdarem as permissões.

Enfim, valer a pena ou não é motivo para um outro tutorial.

Paulino Michelazzo
paulino@michelazzo.com.br
<http://www.michelazzo.com.br>