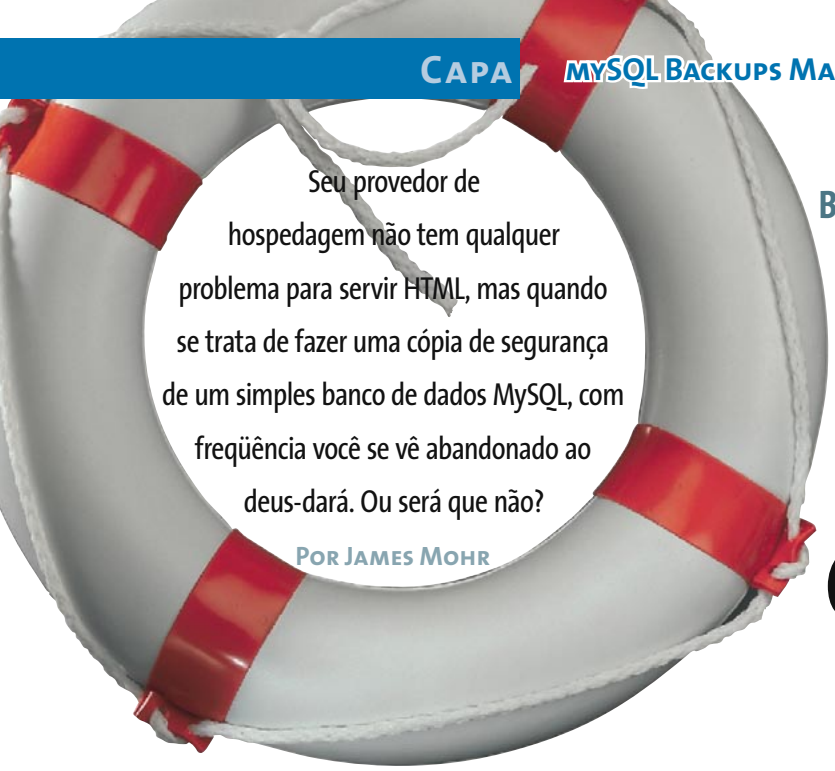

Seu provedor de hospedagem não tem qualquer problema para servir HTML, mas quando se trata de fazer uma cópia de segurança de um simples banco de dados MySQL, com freqüência você se vê abandonado ao deus-dará. Ou será que não?

Por James Mohr

Backup de bancos de dados MySQL

Perdi meu banco de dados!

Mesmo que você esteja só com um sitezinho humilde, não-comercial, os dados armazenados num banco de dados *MySQL* são muito importantes para você. Se esse banco for freqüentemente alterado – como é o caso dos sites interativos com bastante tráfego – um backup confiável torna-se ainda mais importante. Infelizmente, as ferramentas fornecidas pelos provedores de hospedagem para o backup de seus dados são, na melhor das hipóteses, desajeitadas.

Recentemente fiz diversas alterações importantes no banco de dados de meu sistema de testes e me peguei pensando sobre o que aconteceria se eu realmente o estragasse. Faço backups regulares por conta própria em minha máquina local, simplesmente copiando os arquivos do MySQL. Isso funciona, mas o ato de recuperar-se de uma tragédia apresenta problemas únicos. Por isso, comecei a procurar outras soluções. Depois de considerar diversas alternativas para fazer cópias de segurança de bancos SQL (veja o quadro **Opções de backup no MySQL**), fui cativado pelo utilitário *MySQL Backups Manager* da Indexsoft [1].

Você pode rodar o MySQL Backups Manager num servidor web usando qualquer navegador ou como um script na linha de comando. Não seria incorreto dizer que o MySQL Backups Manager é basicamente um *front-end* do programa *mysqldump*, mas essa prática ferramenta de backup definitivamente faz muito mais. Por exemplo, a informação de conexão para seu bancos de dados é armazenada no arquivo principal de configuração e basta fazer referência ao banco em questão para que o backup seja feito. Além disso, o MySQL Backups Manager traz ainda uma porção de recursos adicionais que simplificam a às vezes ingrata tarefa de gerenciar backups em múltiplas máquinas.

O MySQL Backups Manager é distribuído sob uma licença comercial, mas o custo de meros US$15,00 vale o investimento. O utilitário é escrito em Perl com ferramentas padrão do MySQL e pode rodar em qualquer sistema que possua esses dois programas. Embora o Apache não seja obrigatório, você também precisará dele se quiser ter o conforto da interface web do MySQL Backups Manager.

Instalação e configuração básica

O pacote do MySQL Backups Manager é um pequeno arquivo "zipado". Ao descompactá-lo, encontram-se o script `backupsmanager.cgi` e um diretório `modules`, que contém os módulos em Perl para FTP e email. Embora as instruções mandem copiar o script CGI para o lugar de onde ele será executado (normalmente o diretório `cgi-bin` de seu servidor web), isso não funciona a menos que o diretório `modules` esteja no caminho de busca do interpretador Perl. O modo mais fácil é simplesmente copiar tudo para seu diretório `cgi-bin`.

O primeiro passo ao configurar o pacote é rodar o script CGI num navegador web. Por exemplo, acesse a URL: **www.meuprovedor.com.br/cgi-bin/backupsmanager.cgi**. Essa etapa cria o arquivo de configuração (chamado `backupsmanager.pm`) e o coloca numa área de texto de um formulário HTML padrão.

Todos os valores padrão das diversas opções de configuração tanto para a interface web quanto para a linha de comando são definidas neste arquivo. Por padrão, apenas um banco de dados é definido, de forma semelhante a esta:

```
$mySQLDBName[0] = 'mydb0';
$mySQLHost[0] = 'localhost';
$mySQLUser[0] = 'root';
$mySQLPassword[0] = '';
$mySQLDBEmail[0] = '';
```

Como ele tem informações de conexão com seu banco de dados, é preciso certificar-se de que as permissões estão

corretamente atribuídas no arquivo de configuração para evitar acesso indesejado, especialmente se ele estiver em seu servidor web.

Há muitas outras variáveis aqui, como as que definem o diretório padrão para armazenar os arquivos de dump, as que determinam opções para os diversos programas auxiliares e assim por diante. Em geral, a função de cada uma delas é facilmente identificada.

Após terminar de configurar o programa, crie backups a partir da interface web ou simplesmente rode o comando:

```
backupsmanager.cgi -db=#
```

Onde # representa o *número* do banco de dados definido no arquivo de configuração. A localização é definida pela variável `$DBDumpsDir`; por padrão, é o diretório `dumps`, sob o diretório onde está o script `backupsmanager.cgi`. Embora não seja possível mudar o diretório alvo, você também pode usar o parâmetro de linha de comando `--filename=` para especificar um arquivo diferente do padrão, cujo nome usa a sintaxe `YYYY-MM-DD.sql` ou seja, *ano-mês-dia*.sql.

Transferindo o Dump

Mesmo que seja possível agendar uma tarefa em seu servidor web que inicie o MySQL Backups Manager (usando o *cron*), armazenar os arquivos localmente nem sempre é a melhor opção. Se você quiser manter diversos backups, mas tem espaço limitado no servidor, provavelmente precisará de um modo de levá-los a outra máquina. Como mencionei, é possível enviar os dumps via email. Mesmo se você não puder enviar emails para *outras* máquinas, deve ser capaz de enviá-los localmente (para seu endereço de "webmaster").

Opções de backup do MySQL

Meu provedor oferece um modo de fazer um *dump* (cópia crua) de cada um de meus bancos de dados por meio de uma página de administração. A desvantagem: preciso visitar a página para criar o backup – tudo manualmente, nada pode ser automatizado. A página web que faz a cópia de segurança tem um link HTML simples que inicia o backup e o envia a meu navegador. Assim posso abri-lo ou baixá-lo, como qualquer outro arquivo.

Essa ferramenta é muito útil para fazer cópias de segurança de dados. Porém, sou demasiado distraído para me lembrar de fazer backups com esse método com tanta freqüência quanto deveria. Por isso, preciso tentar de outro jeito.

O próprio MySQL oferece um ou dois modos de fazer cópias de segurança de seus dados, como o utilitário *mysqldump*. Ele pode ser utilizado localmente ou a partir de uma máquina remota. Em uma de suas formas mais simples, você pode ter algo parecido com isto:

```
/usr/bin/mysqldump -host=mysql.meuprovedor.com.br ➥
-user=USUÁRIO --password=SENHA nome_do_banco
```

Usando a forma longa, é óbvio o que cada uma das opções faz (sendo `--host` o nome do zcomputador remoto onde o MySQL está). Dependendo de seu sistema e de como você pretende usar o dump resultante, isso é tudo o que você precisa.

O mysqldump oferece todos os comandos SQL necessários para restaurar seu banco de dados. Isso inclui os comandos `create database`, assim como as inserções (comando `insert`) e outros mecanismos necessários para devolver o banco de dados a seu estado original.

O lado ruim deste exemplo é que com freqüência é necessário começar com um banco de dados completamente vazio. Os comandos de criação não vão funcionar se a tabela já existir e, se tiverem uma estrutura diferente, todas as inserções também falharão.

Esse problema pode ser contornado quando o banco de dados é criado simplesmente mandando o MySQL jogar fora (comando `drop` da linguagem SQL) a tabela primeiro, antes de tentar criá-la. Isso é feito automaticamente com a opção `--add-drop-table` do mysqldump. Claro, todos os dados nas tabelas existentes desaparecerão, mas a idéia é essa mesmo.

Como os bancos de dados de meu tutorial de Linux são bem pequenos, não estou realmente preocupado com velocidade e eficiência quando baixo a cópia do banco de dados ou a recupero. Porém, se seu banco contiver milhões de registros, você deve se preocupar com o tempo de duração do processo (talvez o backup possa ser programado para o meio da noite, mas normalmente você está sentado diante da máquina durante uma recuperação).

Um dos modos de aumentar a velocidade da recuperação é aumentar a velocidade das inserções. No MySQL 4.0 e anteriores, o comportamento padrão era que o comando mysqldump gerasse uma inserção por registro. Com milhões de registros, isso levava séculos. A solução é usar a opção `--extended-insert`. Isso compacta todas as inserções numa única declaração SQL, aumentando dramaticamente a eficiência.

Note que nem sempre é possível usar inserções estendidas. Alguns provedores de hospedagem forçam as novas versões do mysqldump para *não* usar as inserções estendidas por razões de compatibilidade. Em geral, o mysqldump usa as declarações SQL que qualquer banco de dados SQL pode entender. Para certificar-se de que você poderá carregar os dados em seu novo banco, seu provedor de hospedagem web pode desabilitar as inserções estendidas.

Como os serviços oferecidos pelos provedores variam dramaticamente, nem todos podem fazer uso da mesma solução. É possível que seu provedor permita acesso ao mysqldump apenas localmente – ou seja, você tem que se conectar a seu servidor. O meu provedor permite configurar quais máquinas terão acesso a meus bancos de dados. O seu talvez não permita isso.

Uma alternativa é criar um script em sua máquina local que execute o mysqldump. Pode ser um script de CGI iniciado de um navegador ou uma tarefa agendada no *cron* lá do servidor (contanto que seu provedor de hospedagem o permita).

Talvez você tenha acesso via *ssh*, que pode ser usado com facilidade para criar backups automáticos. Em áureos tempos eu costumava usar o comando a seguir (quase um *script*), que emprega o mysqldump para criar uma cópia de cada um dos meus bancos de dados, envia para a saída padrão e redireciona essa saída para o comando `ssh`. Não lembro exatamente como era, mas é algo semelhante a isto:

```
ssh user@mysql.meuprovedor.com.br "mysqldump --user=USUÁRIO
--password=SENHA nome_do_banco" > backup.date
```

Já passei por alguns provedores de hospedagem baratinhos que oferecem um banco de dados MySQL mas absolutamente nenhuma garantia de que seus dados estejam seguros. Eles não oferecem mecanismos de backup direto, nem acesso ao cron – nem mesmo um mísero *shell* – e portanto você não pode usar o mysqldump remotamente. Em tais casos, somos freqüentemente surpreendidos com as calças na mão, não é? Bem, isso ainda depende de seu provedor de hospedagem, mas também de sua própria engenhosidade. O MySQL Backups Manager da Indexsoft é uma boa opção para as situações em que seu provedor não oferece um mecanismo de backup direto.

Note que configurar o endereço de email para algo como apenas `webmaster` provavelmente não vai funcionar. Em vez disso, é preciso especificar um nome de host (ex: **webmaster@localhost**), *mesmo* que você esteja mandando o dump para um usuário local.

O MySQL Backups Manager usa o módulo Perl `SendMail.pm` para realmente enviar a mensagem. Se isso não funcionar, você pode definir, na variável `$SMTP`, um comando para enviar o email. Como, por exemplo:

```
$SMTP = '|/usr/sbin/sendmail -t';
```

O envio do arquivo dump por email pode ser feito tanto a partir da interface web quanto através da linha de comando usando o parâmetro `--mail`. Note que, se a variável `$mySQLDBEmail[]` estiver definida em seu arquivo de configuração, seu valor será sempre usado, mesmo se você especificar o endereço de email na linha de comando usando o parâmetro `--email`. Entretanto, se a variável `$mySQLDBEmail[]` estiver vazia, você precisa especificar o endereço na linha de comando ou ajustar a opção `$DefaultEmail` no arquivo de configuração.

Além de enviar o arquivo dump via email, você também pode usar FTP para transferí-lo. O MySQL Backups Manager usa a informação padrão de conexão especificada no arquivo `backupsmanager.pm` para transferir o arquivo.

Por enquanto, não é possível especificar, no arquivo de configuração, servidores FTP diferentes para os bancos de dados individuais, assim como se faz com os endereços de email individuais. Porém, como você já deve ter adivinhado, há opções na linha de comando para especificar uma conexão FTP. Assim, você ainda pode mandar seus dumps de banco de dados para qualquer servidor que desejar, basta indicar seu endereço e parâmetros de conexão.

Roendo até o osso

Também está incluído no arquivo padrão de configuração um modelo para que você possa "sugar" todos os seus bancos de dados. O procedimento é bastante parecido com o dos bancos de dados individuais, exceto porque você especifica o nome do banco desta forma:

```
$mySQLDBName[1] = '--all-databases';
```

Note que isso *não* significa fazer cópias de todos os bancos de dados de que o *MySQL Backups Manager* já ouviu falar. Em vez disso, cria cópias de todos os bancos que pertençam ao usuário e ao servidor informados. Assim, se você estiver tentando fazer dumps de diversos servidores, precisará ter múltiplos blocos "todos-os-bancos-de-dados" (`all-databases`). Além disso, só funciona se você for o usuário root, por isso você provavelmente não será capaz de usar esse método na máquina de seu provedor de hospedagem.

Outras opções

Ter um front-end para o comando mysqldump significa que você pode usar basicamente qualquer opção suportada pelo mysqldump. Isso é o que faz a variável `$DumpDBOptions` que, por padrão, tem as opções `--quote-names` e `--add-drop-table`. Aqui, você pode adicionar qualquer opção que deseje. Por exemplo, se você quiser criar um dump que não use inserções estendidas, pode adicionar `--extended-insert=FALSE`. O problema, aqui, é que o `$DumpDBOptions` se aplica a *todos* os bancos de dados. Assim, ou você tem inserções estendidas ou não tem, não há meio-termo.

É possível contornar essa limitação se você considerar o fato de que o verdadeiro comando mysqldump é construído simplesmente concatenando o conteúdo das diferentes variáveis. Assim, pode-se fazer algo deste jeito:

```
$mySQLPassword[0] = 'PASSWORD
-extended-insert=FALSE';
```

Quando o comando mysqldump é gerado, você simplesmente tem a opção `-extended-insert` logo após a senha, e o comando mysqldump não percebe qualquer diferença. Teoricamente, você pode acrescentar isso como opção à variável `$mySQLDBName[0]`. Porém, isso acaba por criar um diretório dump com um nome esdrúxulo semelhante a este:

```
$DBDumpsDir/localhost---extended-
insert=FALSE linkbat
```

em vez de:

```
$DBDumpsDir/localhost-linkbat
```

simplesmente porque o nome do diretório foi criado como `$mySQLHost[]-$mySQLDBName[]`. Note que a capacidade de definir diferentes opções para diferentes bancos de dados já está em planejamento para uma versão futura.

Se você automatizar o processo de fazer dumps de bancos de dados, há uma ou duas outras variáveis que se deve observar. A primeira é `$ArchiverCommand`, que define o arquivo ou comando de compressão e as opções a usar. O padrão é usar *gzip* e um nível 9 de compressão. Por padrão, o arquivo dump *não* é comprimido. Porém, isso pode ser ativado a partir da linha de comando com a opção `--pack`.

A seguir há a variável `$OldBackups`, que diz ao MySQL Backups Manager que delete arquivos dump mais antigos que um certo número especificado de dias. Use a opção `--old` na linha de comando para especificar esse número. Veja que não há nenhum mecanismo que apague os arquivos antigos *independentemente* do próprio dump. Em vez disso, os arquivos antigos devem ser removidos como parte do processo de dump.

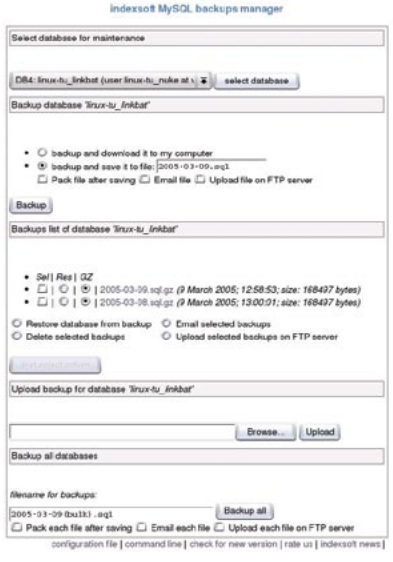

Figura 1: Uma amostra da interface web do *MySQL Backups Manager*.

A interface web

Até agora, só mencionei a linha de comando, mas a principal razão para eu ter começado a mexer no MySQL Backups Manager foi porque eu queria *automatizar* o processo de fazer cópias de segurança dos meus dados. Para iniciar a interface web, acesse a mesma URL mencionada acima para a configuração. Ela o levará à página inicial, mostrada na **figura 1**.

A primeira seção permite selecionar o banco de dados apropriado (listado no arquivo `backupsmanager.pm`). Na segunda seção estão as opções de backup. Por padrão, arquivos dump são armazenados em subdiretórios do diretório definido por `$DBDumpsDir` (um subdiretório para cada banco de dados. Porém, ao selecionar a opção *backup and download it to my computer* (fazer backup e baixar para meu computador), o MySQL Backups Manager cria o arquivo e o manda para seu navegador, que normalmente abre a conhecida janela *Save As* (Salvar Como) padrão.

A seção seguinte (*Backups list of database*, lista de backups do banco de dados) mostra os arquivos dump listados no respectivo diretório dump. Os arquivos dump vêm para cá, sejam eles criados pela linha de comando ou pela interface web. Em ambos os casos, o proprietário do arquivo é o usuário que está executando o mysqldump. Note que, se os arquivos forem criados a partir da interface web, o proprietário é o usuário sob o qual os servidor web estiver rodando. Se estiver usando a linha de comando, o proprietário do arquivo é o usuário normal. Se tiver usado ambos os mecanismos para criar o arquivo dump, você poderá ter problemas com as permissões de acesso.

Aqui, você pode realizar diversas operações diferentes nos arquivos. Por exemplo, você pode carregar os dados para seu banco a partir de um arquivo existente. Tenha cuidado, porque o MySQL Backups Manager simplesmente usa o comando *mysql* com o arquivo especificado. Como o padrão é usar o `--add-drop-table`, as tabelas são apagadas e recriadas antes de se fazerem as inserções. Tipicamente, é isso que se poderia esperar. Se você não usar a opção `--add-drop-table`, ocorrerá um erro quando tentar criar a tabela e as inserções *acrescentarão* os registros a ela. Você também pode enviar os arquivos existentes via email para o endereço padrão, assim como carregar os arquivos dump para seu servidor de FTP padrão.

A seção *Upload backup for database* permite enviar um arquivo de uma máquina local para seu servidor web. Isto pode ser útil se, por exemplo, seu provedor de hospedagem não fizer o backup por você e algo acontecer ao banco de dados. Outro caso seria ao fazer mudanças na estrutura do banco de dados. Baixe o banco inteiro para sua máquina local, faça as mudanças e crie um novo dump, que é então carregado no servidor sem que este precise ser desativado. Então você pode restaurar essa cópia com os dados mais recentes e as mudanças no banco de dados.

Como o comando mysql é usado para restaurar os dados executando o comando SQL no arquivo dump, naturalmente surge um problema se esse arquivo estiver comprimido, porque o mysql não conseguirá lidar com ele. Por isso, será necessário descomprimí-lo antes de tudo. Você verá que, diante de cada arquivo dump, há um botão rotulado *GZ*. Isso significa que o arquivo está comprimido. Desmarcando o botão, o arquivo é descompactado para você. Se você marcar o botão, o arquivo será compactado.

Outra coisa a se notar é que, por padrão, não é possível transferir dados de um banco para outro. Por exemplo, se você tiver um sistema de teste e quiser sincronizar os dados a partir do sistema principal. Para fazê-lo, pode-se usar tanto o comando mysql diretamente ou mover manualmente os arquivos de um diretório para outro. Como a interface web simplesmente exibe uma lista dos arquivos nos diretórios respectivos, ele não tem consciência do fato de que você moveu manualmente o arquivo.

Verdadeira pechincha

Mesmo sendo um produto comercial (uma licença custa US$15), o MySQL Backups Manager vale o investimento. Eu poderia ter criado esse script sozinho, mas quando comparo o custo com o tempo que levaria para escrever o código, não vale nem mesmo a pena discutir a idéia. Tempo é dinheiro e o MySQL Backups Manager se pagou muitas vezes.

Em apenas alguns minutos, é possível configurar backups automáticos de seus dados MySQL, não importa onde estejam. Não é preciso passar horas, ou mesmo dias, programando os recursos necessários. O MySQL Backups Manager oferece uma solução prontinha para usar. ■

Informações

[1] Homepage da Indexsoft: www.indexsoft.com

[2] O comando mysqldump: dev.mysql.com/doc/mysql/en/mysqldump.html

[3] Provedor que usa o MySQL Backups Manager: www.imagelinkusa.net