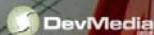


Especial Oracle - Saiba como manter seu banco de dados atualizado e não corra riscos de segurança

Feita para Desenvolvedores de Software e DBAs

SQL magazine

Edição 44 :: Ano 4 : R\$11,90



SQL Server

Aprenda a trabalhar com os mecanismos
de busca textual do SQL Server Parte 2

**Data Warehouse e
OLAP – Parte 5**

Usando cálculos avançados e MDX

MySQL

Conheça as técnicas de
backup e recuperação de dados

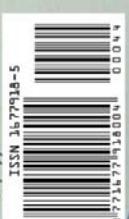
Soluções PRONTAS de MODELAGEM

- Call Center
- Controle de Estoque
- Controle de Ponto
- Faturamento e Cobrança

Brinde!

Compre esta edição e ganhe **10 vídeo-aulas**
sobre Padrões de Projeto com Java

- Factory method
- Builder
- Singleton e Template method
- Observer
- Composite
- Facade
- Bridge
- Factory
- Adapter
- Strategy



ISSN 1677-9184-5

Desafio SQL Magazine

Mais um desafio de modelagem de dados.

Desenvolvimento

Implemente testes unitários em bases de
dados com DBUnit



Técnicas de backup e recuperação de dados em MySQL

[Banco de Dados]

A utilização de um SGBD tem se tornado cada vez mais popular para as mais variadas aplicações. Neste contexto, a confiabilidade dos dados torna-se crucial, uma vez que todo o processo de gestão passa a depender cada vez mais destas bases de dados. Entretanto, em ambientes de produção podem ocorrer problemas de natureza variada que podem comprometer este armazém de dados, tais como falhas de hardware, software ou até mesmo execução de comandos indesejados por parte dos usuários – DROP DATABASE – por exemplo.

Com o intuito de evitar a perda de informações nestas situações de desastres é preciso manter uma cópia de segurança dos dados, conhecida como backup, para que este repositório possa ser restaurado completamente sem causar prejuízos para a instituição que o utiliza.

O objetivo deste artigo é discutir os principais fatores relacionados à execu-

ção de uma rotina consistente de backup e apresentar as principais ferramentas existentes no MySQL para que se tenha uma cópia íntegra de suas informações. Além disto, serão apresentadas técnicas de recuperação de dados a partir do backup de forma a garantir que toda a informação existente no momento do desastre seja restaurada sem haver comprometimento da integridade ou até mesmo perda de dados.

Procedimento de backup de dados

Ao se executar uma rotina para copiar as informações armazenadas pelo SGBD é preciso ter alguns cuidados para que se tenha uma imagem dos dados que seja fiel ao dado original e, portanto, confiável. Sabemos que em sistemas em produção a base de dados possui um comportamento dinâmico e a cada fração de tempo pode se encontrar em uma situação distinta. Neste caso, o backup deve espelhar coerentemente a situação dos dados em um ponto qualquer do tempo.



Éber Duarte

(eber@eacsoftware.com.br)

é bacharel em Ciência da Computação, pós-graduado em Engenharia Elétrica e MySQL Professional Certified. Trabalha há 5 anos na EAC Software (BH/MG) como Analista e desenvolvedor de sistemas, atuando especialmente no desenvolvimento de sistemas Web. Atualmente, também é consultor e instrutor do banco de dados MySQL.

A **Figura 1** ilustra a evolução de uma base de dados e o resultado da execução de uma cópia consistente.

Percebe-se a partir da análise da **Figura 1** que o procedimento de cópia não é discreto, ou seja, ele se estende por um período de tempo. Assim, a rotina de cópia de dados deve garantir uma imagem da base de dados que reflita a sua situação no momento em que o backup se iniciou (T1) ou que contenha a posição dos dados ao término do procedimento (T2). Neste ponto, surge uma questão fundamental que é determinar se a base de dados copiada estará ou não acessível aos seus usuários no momento em que a cópia de segurança está sendo executada. Sob esta ótima é possível proceder de três formas distintas:

1. Fazer a cópia com o SGBD em execução, não impondo nenhuma restrição de acesso aos dados, conhecido como *backup online*.
2. Parar a execução do SGBD de forma que não haja nenhum acesso aos dados durante a execução da cópia de dados.
3. Impor restrições de escrita aos dados com o uso de *locks* (bloqueios), permitindo apenas o acesso de leitura >s informações durante a execução da rotina de backup.

A opção número 1, que é o *backup online*, é fundamental para aplicações onde não há a possibilidade de limitar ou interromper o acesso ao sistema. Neste caso, a estratégia de backup deve conter mecanismos para realizar uma espécie de congelamento dos dados, refletindo a situação da base de dados no momento em que foi iniciada no seu término. Isto se deve ao fato de que em um SGBD existem informações relacionadas que por regras de normalização ficam armazenadas em tabelas distintas. Neste caso, se durante a cópia há uma transação em andamento e esta realiza uma alteração nestas relações, dependendo da ordem em que as tabelas são copiadas, pode-se gerar uma imagem incorreta da base de dados - copiar os itens de uma nota fiscal inexistente, por exemplo. Desta forma, o *backup* não poderá ser utilizado em caso de falhas, uma vez que não há consistência das informações nele armazenadas.

A segunda opção de *backup* garante que

a base de dados estará consistente uma vez que não haverá nenhum acesso aos dados durante a cópia, dado que o SGBD não estará em execução. Esta solução pode ser utilizada apenas nas aplicações onde o sistema não opera 24 horas por dia, por isto não se aplica a todos os contextos.

Finalmente, a terceira opção pode ser utilizada caso o sistema não possa ficar completamente fora do ar, mas que permite uma interrupção de apenas uma parte de suas funcionalidades. Por exemplo, em sistemas onde há um elevado número de acessos de leitura e as atualizações são esporádicas, pode-se aplicar os mecanismos de *locks* para garantir que não haverá escrita durante a cópia dos dados. Assim, não há interrupção completa do sistema, mas garante-se que haverá ao final do *backup* uma imagem coerente dos dados que poderá ser utilizada em casos de falhas do sistema.

Backup lógico versus backup físico

Uma vez definida a estratégia que será adotada para realização da cópia dos dados, deve-se definir quais os tipos de informações serão copiadas. Neste aspecto, podem-se utilizar duas formas de backup, que são o arquivamento apenas das informações armazenadas pelo SGBD, conhecida como backup lógico, ou copiar toda a estrutura de arquivos e diretórios gerados pelo mesmo, conhecida como backup físico.

A escolha de uma abordagem dependerá de três fatores chave que são: desempenho, segurança e portabilidade dos dados.

A execução do backup físico é mais rápida, pois não tem que acessar todos os dados armazenados pelo SGBD, sendo assim, o tempo para executar a tarefa será o tempo de copiar os arquivos no sistema operacional, que em geral é um processo rápido. Além disto, a restauração desta cópia é mais rápida uma vez que basta retornar com os arquivos do *backup* para o local onde o MySQL armazena os seus arquivos. Em contrapartida, esta metodologia não é tão segura, visto que durante o processo podem ser copiados arquivos corrompidos sem que se perceba a falha nos dados, e assim gerar uma cópia inconsistente ou incorreta dos dados. Outra

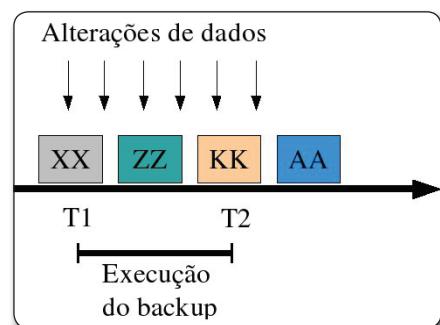


Figura 1. Evolução da base de dados durante o procedimento de *backup*

desvantagem deste método é que como o MySQL suporta a utilização de vários engines para a definição de tabelas (MyISAM, InnoDB, etc), a restauração dos dados deverá sempre ser feita em termos do engine utilizado no momento do backup. Além disto, não é possível restaurar um backup do MySQL em outro SGBD, o que é relevante em ambientes onde se opera com vários SGBDs distintos e onde há compartilhamento de dados entre eles.

O backup lógico, em contrapartida, é um processo mais lento, uma vez que toda a base de dados deverá ser lida para que apenas as suas informações sejam exportadas. Neste caso, a vantagem é a possibilidade de restauração dos dados em qualquer engine ou até mesmo em outro SGBD. Outro benefício é que caso haja arquivos corrompidos no momento do acesso aos dados para a geração do backup, a falha será detectada. Desta forma, a cópia de dados corrompidos é evitada e garante-se a integridade do backup. O contraponto desta abordagem é que tanto a cópia quanto a restauração dos dados é lenta, visto que todos os dados deverão ser exportados/importados nesta hora. No caso de uma base de dados volumosa, o processo pode consumir um tempo elevado, causando assim uma indisponibilidade considerável do sistema em caso de falhas.

Veremos a partir de agora algumas opções para realização de backup no MySQL.

mysqldump

O MySQL possui algumas ferramentas que permitem fazer backup lógico, físico, online e off-line. A primeira delas é o mysqldump, que permite converter

toda a estrutura da base de dados para comandos SQL. Isto é, serão gerados os comandos CREATE DATABASE de cada banco de dados, para cada tabela serão gerados os comandos CREATE TABLE e os INSERTs para cada um de seus registros.

Este cliente gera a saída de dados utilizando a saída padrão do sistema (stdout). Entretanto, você pode redirecioná-la para um arquivo, que será o backup, ou até mesmo para outro servidor MySQL, permitindo a sincronização de duas instâncias. O grande benefício desta ferramenta é que ele funciona independente do engine utilizado para as suas tabelas, e também apresenta um vasto conjunto de opções que permitem controlar quais as informações devem ser copiadas ou até mesmo o formato da saída gerada, XML por exemplo.

A **Listagem 1** ilustra a execução de um backup do banco de dados Producao, armazenando os dados em um arquivo texto.

No comando anterior, a opção “-u” indica o usuário do MySQL, a opção “-p” faz com que o mysqldump peça a senha deste usuário e o “-B” permite informar o nome do banco de dados a ser copiado. O parâmetro “--master-data” faz com que todas as tabelas sejam bloqueadas (*lock*) durante a cópia, gerando assim uma cópia consistente dos dados. Esta ainda inclui no *backup* o nome do *log* binário e a posição em que o MySQL se encontrava no momento da cópia. O argumento “--flush-logs” é utilizado para gerar um novo arquivo de *log* binário sincronizando assim a imagem dos dados com o *log* de alterações, cuja função será explicada mais adiante neste artigo.

Para redirecionar os dados para o arquivo, basta utilizar o operador “>” e informar o nome do arquivo. Este arquivo contém os comandos SQL, em modo texto, podendo assim ser compactado e armazenado em uma unidade externa, ocupando assim menos espaço.

A **Listagem 2** ilustra o uso do mysqldump para copiar os dados de uma instância de MySQL para outra que está em outro endereço na rede.

Neste exemplo, o banco de dados Producao será copiado de uma instância para outra, ou seja, os comandos gerados

Listagem 1. Criando backup através do comando mysqldump.

```
shell>mysqldump -u usuario -p -B Producao --master-data --flush-logs > /backup/producao.dump
```

Listagem 2. Copiando os dados entre instâncias do MySQL

```
shell>mysqldump -u usuario -p -B Producao --master-data --flush-logs | mysql -h 192.168.10.2 -u user_remoto -psenhausuario
```

Listagem 3. Gerando backup online no MySQL via mysqldump.

```
shell>mysqldump -u usuario -p -B Producao --single-transaction --master-data --flush-logs > /backup/producao.dump
```

Listagem 4. Gerando backup através do comando SELECT.

```
mysql>SELECT * INTO OUTFILE '/backup/usuarios.txt'
      -> FIELDS TERMINATED BY '|';
      -> LINES TERMINATED BY '\n';
      -> FROM usuarios;
```

Listagem 5. Gerando backup através do mysqlhotcopy.

```
shell>mysqlhotcopy -u root Producao /backup
```

pelo mysqldump serão executados automaticamente pela instância remota do MySQL, que no exemplo se encontra no endereço 192.168.10.2.

Caso o mysqldump seja utilizado para a cópia de tabelas utilizando os engines que implementam o controle de transação, por exemplo InnoDB, é possível realizar uma cópia de dados consistente sem que haja bloqueio dos dados, ou seja, uma cópia online. Para isto, basta utilizar a opção “--single-transaction” que fará com que o backup seja feito em uma única transação. Isto significa dizer que no início do backup será feita uma imagem consistente dos dados utilizando o MVCC (*Multi-Versioning Concurrency Control*), e então apenas os dados que estiverem gravados (*committed*) até este ponto estarão presentes no backup. A **Listagem 3** ilustra esta situação.

Vale ressaltar que ao utilizar opção “--single-transaction” combinada com o “--master-data”, a primeira faz com que não seja realizado nenhum *lock* nas tabelas que estão sendo copiadas, logo a segunda não terá qualquer efeito.

SELECT

Outra forma de realizar uma cópia de dados é através da utilização do comando SELECT, onde é possível redirecionar a saída para um arquivo texto formatado, conforme ilustra a **Listagem 4**.

O exemplo faz uma cópia dos dados da tabela *usuarios* para um arquivo chamado *usuarios.txt*. Este arquivo con-

terá apenas os dados da tabela com as colunas separadas pelo caractere “|” e os registros delimitados pelo símbolo “\n” (que indica término de linha). Este comando deve ser executado para cada tabela, lembrando sempre de fazer um lock de todas as tabelas antes de iniciar a cópia, garantindo assim a consistência dos dados.

Tabelas MyISAM – Cópia de Arquivos

Caso a opção seja a cópia de arquivos, deve-se considerar o engine utilizado. Para o backup físico de tabelas MyISAM, deve-se executar o seguinte procedimento:

1. Parar o SGBD ou travar as tabelas para garantir a integridade.
2. Executar um comando *flush tables* para que as tabelas que estiverem em memória sejam gravadas em disco.
3. Copiar os arquivos .frm, .MYD, .MYI, .TRG e .TRN.
4. Liberar o *lock* das tabelas ou reiniciar o MySQL.

Vale ressaltar que o passo 2 só será executado caso a opção seja não parar o SGBD. O procedimento descrito não copiaria as *stored procedures* e funções existentes, uma vez que estas são armazenadas em um banco de dados do MySQL. Portanto, para que estes objetos sejam copiados, deve-se incluir na rotina de backup a cópia da tabela *proc* armazenada no banco de dados *mysql*.

Tabelas MyISAM – mysqlhotcopy

Existe uma ferramenta escrita em perl, chamada *mysqlhotcopy* que executa os quatro passos do procedimento descrito anteriormente, com a exceção de que não copia as triggers da tabela (TRG e TRN). A **Listagem 5** ilustra a sua utilização.

No exemplo da **Listagem 5**, todas as tabelas do banco de dados *Producao* serão copiadas para o diretório */backup*, sendo que as tabelas devem ser do tipo MyISAM.

Tabelas InnoDB – Cópia de Arquivos

Caso a sua base de dados possua tabelas do tipo InnoDB, o procedimento para a cópia de arquivos difere daquele utilizado para MyISAM, pois o mesmo apresenta uma estrutura de arquivos diferente. Assim, o procedimento para a cópia de arquivo é:

1. Parar o SGBD ou travar as tabelas para garantir a integridade.
2. Executar um comando *flush tables* para que as tabelas que estiverem em memória sejam gravadas em disco.
3. Copiar os arquivos de dados (*data files*).
4. Copiar os arquivos de log (*log files*).
5. Copiar o arquivo de configuração, por segurança.

É possível fazer o *backup* físico *online* utilizando o InnoDB Hot Backup, que é uma ferramenta de *backup* comercial que pode ser adquirida através do site www.innodb.com. Para realizar esta cópia, deve-se criar um arquivo de configuração (*backup.cnf*, **Listagem 6** – linhas 8 a 14) com as informações de onde serão copiados os arquivos do InnoDB. Este arquivo pode ser colocado em qualquer diretório, mas por conveniência pode ser mantido no mesmo diretório onde os arquivos de backup serão armazenados. Assim, para executar a cópia basta informar o arquivo de configuração do MySQL, que contém as informações de onde estão os *data files* e *log files* do InnoDB (*my.cnf*, **Listagem 6** – linhas 1 a 7), bem como o arquivo de configuração de *backup* definido anteriormente. Desta forma, basta invocar a ferramenta *ibbackup*, informando os dois arquivos de configuração conforme realizado na linha 15 da **Listagem 6**.

Listagem 6. Arquivos de configuração e backup com a ferramenta InnoDB Hot Backup.

```

/etc/my.cnf

1. [mysqld]
2. datadir = /home/mysql/data
3. innodb_data_home_dir = /home/mysql/data
4. innodb_data_file_path = ibdata1:10M:autoextend
5. innodb_log_group_home_dir = /home/mysql/data
6. set-variable = innodb_log_files_in_group=2
7. set-variable = innodb_log_file_size=20M/home/mysql/backup/backup.cnf
8. [mysqld]
9. datadir = /home/mysql/backup
10. innodb_data_home_dir = /home/mysql/backup
11. innodb_data_file_path = ibdata1:10M:autoextend
12. innodb_log_group_home_dir = /home/mysql/backup
13.set-variable = innodb_log_files_in_group=2
14.set-variable = innodb_log_file_size=20M

Execução da rotina de backup
15.shell>ibbackup /etc/my.cnf /home/mysql/backup/backup.cnf

Dados copiados
16.shell>$ ls -lh /home/mysql/backup
17. total 824M
18. -rw-r-- 1 ed users 22M Jan 21 17:42 ibbackup_logfile
19. -rw-r-- 1 ed users 10M Jan 21 17:36 ibdata1

```

O arquivo de configuração do *backup* é similar ao arquivo de configuração do MySQL, ou seja, contém a definição do *tablespace* e arquivos de log (**Listagem 6** – linhas 10 a 14) do InnoDB, exatamente como está configurado no *my.cnf*. A única diferença é que a variável *datadir* (linha 2) do *backup.cnf* contém o local onde os arquivos de *backup* serão colocados ao executar o *ibbackup*, enquanto o *datadir* encontrado no *my.cnf* aponta para o diretório de dados do MySQL, isto é, onde estão os arquivos do SGBD.

Todos os arquivos do InnoDB definidos no *my.cnf* serão armazenados no diretório */home/mysql/backup*, conforme define o arquivo *backup.cnf* (linha 9). O detalhe importante é que o InnoDB armazena todas as tabelas, independentemente do banco de dados em que se encontram, em um *tablespace* compartilhado que pode ser constituído de vários arquivos. O *ibbackup* fará a cópia deste *tablespace*, isto é, de todos os arquivos que o constituem, e neste caso fará o *backup* de todas as tabelas InnoDB, de todos os bancos existentes. O resultado do *backup* utilizando esta ferramenta está descrito nas linhas 16 a 19 da **Listagem 6**.

A cópia parcial de dados em InnoDB, ou seja, fazer *backup* apenas de uma tabela e não de todos os bancos de dados conforme ocorre na **Listagem 6**, só será possível caso o SGBD esteja configurado para utilizar múltiplos *tablespaces*. Quando há a utilização de um *tablespace*

compartilhado todas as tabelas InnoDB, independente do banco a qual elas pertencem, são colocadas em arquivos pré-configurados, que são copiados pela ferramenta. Dessa forma, não é possível copiar uma tabela separadamente.

No caso da utilização de múltiplos *tablespaces*, cada tabela será colocada em um arquivo separado. Com isto, pode-se informar ao InnoDB Hot Backup quais arquivos devem ser copiados, permitindo assim a indicação para cópia de apenas alguns arquivos (1 arquivo por tabela), e por consequência, de apenas algumas tabelas do seu banco de dados.

Vale ressaltar que a ferramenta *ibbackup* faz apenas a cópia de arquivos que poderia ser feita com o comando *cp* ou *copy* do próprio sistema operacional. O detalhe importante é que a ferramenta permite a cópia consistente com o SGBD em execução, enquanto ao utilizar os comandos de cópia do sistema operacional manualmente deve-se interromper a execução do SGBD para garantir a consistência dos arquivos e dados.

Replicação de dados

Outra opção interessante para a cópia de segurança é a replicação, pois esta permite configurar um servidor *master* que receberá todas as alterações de dados e um servidor *slave* que copiará todas as alterações ocorridas neste *master*. Desta forma, tem-se um sincronismo de dados online, dado que tudo que ocorrer no

master será copiado pelo slave imediatamente.

A desvantagem desta metodologia é a necessidade de ter dois servidores para a configuração do recurso, mas por outro lado fornece capacidade de *fail-over* para o seu sistema, ou seja, o servidor *slave* assume o papel do servidor *master* caso esse tenha algum problema e fique “fora do ar”. Vale ressaltar que esta abordagem não elimina a necessidade de um backup dos dados utilizando as ferramentas apresentadas anteriormente, pois a replicação espelha todos os comandos executados no servidor *master*. Portanto, se um usuário apaga uma tabela por engano no servidor *master*, a mesma será apagada no *slave*.

Ilustrando um backup físico e lógico

Para ilustrar as características destas duas abordagem de *backup* de dados, será utilizada a ferramenta mysqldump e a

cópia direta de arquivos em um banco de dados *world*, que pode ser encontrado no endereço <http://downloads.mysql.com/docs/world.sql.gz>. O objetivo é exemplificar as duas abordagens permitindo um comparativo entre elas no que diz respeito a tempo de execução e espaço físico do *backup* gerado. Esta base de dados consiste de três tabelas totalizando 488 KB de dados.

Neste exemplo, será utilizado um MySQL instalado no ambiente Linux, cujo diretório de dados se encontra em /usr/local/mysql/data. O diretório de dados é o local onde o MySQL armazena, por padrão, as informações de todos os bancos de dados e arquivos de log do sistema.

A primeira abordagem é a cópia direta dos dados, que por se tratar de tabelas MyISAM, basta que o diretório *world* seja copiada para o local do *backup*. A Listagem 7 ilustra esta situação.

Neste caso, o tamanho do *backup* é igual

ao tamanho da base de dados original, uma vez que a mesma é uma cópia de todos os arquivos do banco, conforme ilustra a Listagem 7 (linha 4). O *backup* foi realizado com o SGBD desligado (linha 1) para garantir a consistência dos dados.

Para ilustrar o *backup* lógico de dados, será realizada a cópia do banco de dados *world* utilizando a ferramenta mysqldump, gerando um arquivo no diretório de *backup* conforme ilustra a Listagem 8.

O resultado da cópia de dados utilizando o mysqldump difere daquela ilustrada na Listagem 7. Nesta última há apenas um arquivo texto (*world.dump*) contendo os comandos CREATE TABLE de cada tabela e os comandos INSERT para cada um de seus registros. No exemplo, o tamanho do arquivo de *backup* é ligeiramente inferior ao tamanho do dado original (238K x 488k), mas em geral não é o cenário comum, pois em bases de dados volumosas a geração de arquivo texto contendo todas as informações fornece um arquivo com tamanho superior aos dos arquivos de dados do SGBD.

Restauração do backup

Uma vez definida a política de *backup*, é preciso discutir a forma de utilização dos mesmos para a recuperação de dados em momentos de falhas diversas.

A Figura 2 ilustra uma situação em um momento de falha.

Para recuperar a informação existente no momento da falha é preciso restaurar a cópia de segurança contendo a imagem dos dados no momento em que a cópia foi realizada. A partir daí, é preciso recuperar todas as informações alteradas entre o momento do *backup* e o instante em que ocorreu a falha do sistema. Neste ponto é que se torna crucial o processo de sincronização do log binário com o *backup*, citado anteriormente.

Este log do MySQL possui todos os comandos que alteram a base de dados, possibilitando assim que todas as alterações realizadas após o *backup* sejam restauradas. O log binário possui arquivos com números seqüenciais que são gerados todas as vezes que o SGBD é reiniciado ou a cada execução do comando FLUSH LOGS. Desta forma, no momento do *backup* é preciso gerar um novo arquivo

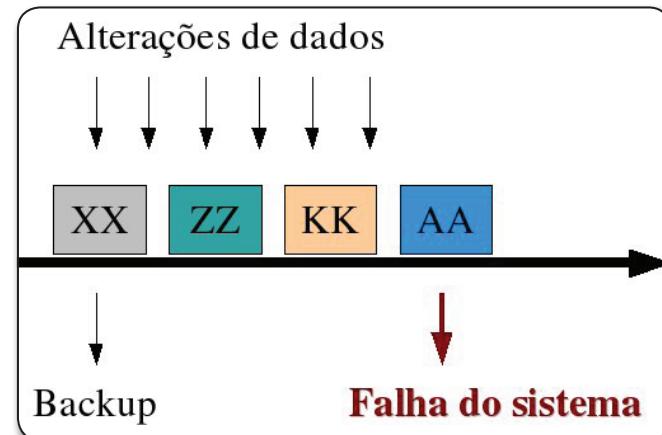


Figura 2. Rotina de *backup* e evolução dos dados até o momento da falha

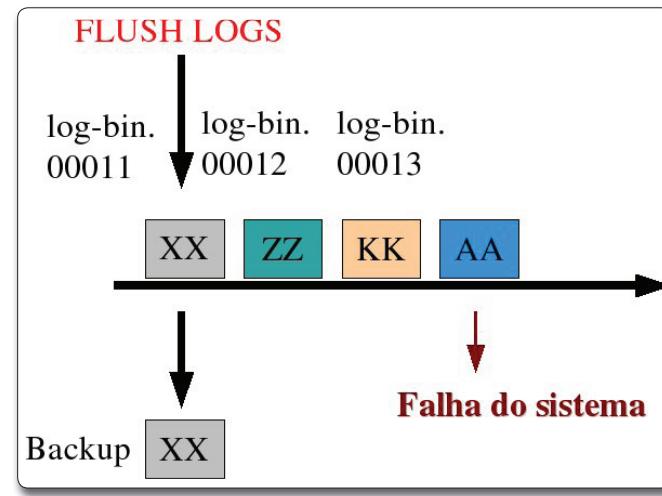


Figura 3. Ilustração do *backup* e logs binários em situação de falha

para acumular as alterações realizadas a partir dali, facilitando o processo de recuperação de informações. A **Figura 3** ilustra esta situação.

Tendo a cópia consistente dos dados e o log binário sincronizado, basta restaurar o *backup* e aplicar os *logs* que foram gerados a partir do momento deste *backup*. A restauração dos dados deve ser feita de acordo com a estratégia de cópia utilizada. Se for cópia física, os arquivos devem ser copiados de volta. Caso seja um backup de dados, restaure os dados novamente executando o arquivo de dump - execução dos comandos SQL contidos nele - ou se há apenas os dados, utilize o comando LOAD DATA INFILE para importá-los. A **Listagem 9** ilustra a restauração de um backup feito com o mysqldump.

Vale ressaltar que pelo fato de ter sido utilizada a opção -B na hora de gerar o dump, a informação do banco de dados está embutida no arquivo de *dump*, assim não é preciso informar o nome do banco de dados na restauração, conforme ilustra a **Listagem 9**.

Caso o *backup* tenha sido realizado com o InnoDB Hot Backup, basta executar a ferramenta *ibbackup* utilizando a opção “-apply-log” informando os arquivos de configuração do *backup* e do MySQL, assim como utilizado no momento da cópia dos dados. A **Listagem 10** ilustra esta rotina.

Uma vez restaurada a cópia de segurança, basta aplicar todos os *logs* gerados entre o momento do *backup* e a falha. Supondo a situação ilustrada pela **Figura 3**, a restauração dos logs seria feita conforme descrito na **Listagem 11**.

O mysqlbinlog converte o arquivo de log binário para comandos SQL que podem ser executados diretamente pelo cliente mysql. Esta ferramenta possui os parâmetros “--start-position”, “--start-datetime”,

Listagem 7. Backup do banco de dados world utilizando cópia de arquivos

```
1. shell#/etc/init.d/mysql stop // é necessário parar o serviço antes da
cópia
2. shell#cp -R /usr/local/mysql/data/world /backup // procedimento de cópia
3. shell#ls -la /backup/world/ // confirmação de que a cópia foi
realizada
4. total 504
5. -rw-r----- 1 root root 273293 May 28 08:50 City.MYD
6. -rw-r----- 1 root root 43008 May 28 08:50 City.MYI
7. -rw-r----- 1 root root 8710 May 28 08:50 City frm
8. -rw-r----- 1 root root 62379 May 28 08:50 Country.MYD
9. -rw-r----- 1 root root 5120 May 28 08:50 Country.MYI
10. -rw-r----- 1 root root 9172 May 28 08:50 Country frm
11. -rw-r----- 1 root root 38376 May 28 08:50 CountryLanguage.MYD
12. -rw-r----- 1 root root 23552 May 28 08:50 CountryLanguage.MYI
13. -rw-r----- 1 root root 8702 May 28 08:50 CountryLanguage frm
14. shell#/etc/init.d/mysql start
```

Listagem 8. Backup do banco de dados world utilizando o mysqldump

```
1. shell#mysqldump -u root -p senha -master-data -B world > /backup/world.dump
2.shell#ls -lah /backup/world.dump
3.-rw-r--r-- 1 root root 238K May 28 09:00 /backup/world.dump
```

Listagem 9. Restauração do backup feito na Listagem 1 através do mysqldump.

```
shell>mysql -u usuario -p < /backup/producao.dump
```

Listagem 10. Restauração de dados de um backup realizado com o InnoDB Hot Backup.

```
shell>ibbackup -apply-logs /etc/my.cnf /home/mysql/backup/backup.cnf
```

Listagem 11. Restauração de dados a partir do log binário.

```
shell>mysqlbinlog mysql-bin.000012 mysql-bin.000013 | mysql --one-database Producao -u
usuario -p
```

“--stop-position” e “--stop-datetime”, que permitem fazer a recuperação de dados em um ponto exato do tempo (point-in-time recovery), facilitando assim a recuperação de dados em casos onde há a execução de um comando indesejado, como o DROP DATABASE por exemplo. Neste caso, é possível executar partes do arquivo de log evitando a execução destes comandos que geraram a perda dos dados.

Conclusão

Existem várias técnicas de realizar cópias de segurança em MySQL. Este artigo apresentou as soluções desenvolvidas pela própria MySQL AB, mas existem diversas ferramentas desenvolvidas por terceiros que também podem ser utilizadas para esta tarefa, tais como Arkeia,

EMC Clariion, dentre outras que podem ser encontradas em <http://solutions.mysql.com/solutions/?type=24>.

O fato importante que deve ser ressaltado é a possibilidade de elaboração de uma solução de backup robusta e de baixo custo a partir da utilização das ferramentas básicas do MySQL ou até mesmo empregando componentes de software GPL. Neste caso, garante-se a segurança e confiabilidade do sistema além de uma redução no tempo de inoperabilidade do sistema em momentos de falhas.

Diversas outras técnicas podem ser associadas à tarefa de backup, visando simplificar esta tarefa e prover um ambiente mais instável, como a definição e implantação de um cronograma para backup automático dos dados no servidor. ●

