Documentos

Setembro, 2003 31

ISSN 1677-9274

Recomendações para *Back up* nos Sistemas de Gerenciamento de Banco de Dados MySQL, PostgreSQL e OpenIngres

Embrapa

ISSN 1677-9274
Setembro, 2003

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 31

Recomendações para *Back up* nos Sistemas de Gerenciamento de Banco de Dados MySQL, PostgreSQL e OpenIngres

Marcelo Gonçalves Narciso

Campinas, SP
2003

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária "Zeferino Vaz" Barão Geraldo
Caixa Postal 6041
13083-970 Campinas, SP
Telefone (19) 3789-5743 Fax (19) 3289-9594
URL: http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.br

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Editoração eletrônica: *Área de Comunicação e Negócios (ACN)*

1ª edição on-line - 2003

Narciso, Marcelo Gonçalves.
Recomendações para *back up* nos sistemas de gerenciamento de banco de dados MySQL, PostgreSQL e OpenIngres / Marcelo Gonçalves Narciso. — Campinas : Embrapa Informática Agropecuária, 2003.
28 p. : il. — (Documentos / Embrapa Informática Agropecuária ; 31)

ISSN 1677-9274

1. *Back up*. 2. Sistema de gerenciamento de banco de dados. 3. MySQL. 4. PostgreSQL. 5. OpenIngres. I. Título. II. Série.

CDD - 005.86 21st ed.

Autor

Marcelo Gonçalves Narciso
Dr. Em Computação Aplicada, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo
13083-970 - Campinas, SP.
Telefone (19) 3789-5762
e-mail: narciso@cnptia.embrapa.br

Apresentação

Este trabalho descreve como realizar *back up* e recuperação de dados nos sistemas de gerenciamento de banco de dados (SGBD) MySQL, PostgreSQL e OpenIngres. Para tanto, são usados comandos do próprio SGBD. Exemplos e *scripts* para *back up* e recuperação também estão descritos neste trabalho. Também será feita uma comparação entre os *back up* dos três bancos de dados.

O *back up* pode ser feito quando o SGBD está ativo ou não (*on-line* ou *off-line*). Como o banco de dados deve sempre estar em atividade, o ideal é que o *back up* seja feito com o banco *on-line*. Nos finais de semana ou à noite, caso o banco tenha pouco ou nenhum acesso, pode-se fazer *back up* com o banco fora do ar (*off-line*). Este documento mostra como fazer o *back up* nos dois casos.

Os SGBDs PostgreSQL e MySQL são enfocados neste documento pelo fato de terem grande aceitação no mercado pois são *freeware* e fáceis de serem instalados e mantidos. O MySQL é muito usado em aplicações voltadas para a Internet e o PostgreSQL é um banco robusto, comparável aos SGBDs comerciais. O OpenIngres também é enfocado neste trabalho por ser usado na Embrapa (e outras importantes empresas, como a Petrobrás, por exemplo).

Um importante banco de dados, o Oracle, que é líder mundial de mercado, não será enfocado neste trabalho mas em um outro trabalho a parte, visto que é muito detalhado e, para tanto, merece uma publicação específica.

Espera-se que este trabalho seja útil ao usuário de SGBD ou ao administrador quando estes necessitarem fazer um *back up* ou recuperação. A partir deste documento, o usuário terá base para elaborar sua própria política de *back up*, visto que cada projeto tem suas próprias necessidades.

José Gilberto Jardine
Chefe-Geral

Sumário

Recomendações para *Back up* nos Sistemas de Gerenciamento de Banco de Dados MySQL, PostgreSQL e OpenIngres

Marcelo Gonçalves Narciso

Introdução

Para evitar perdas de dados ou minimizar esta perda, por uma série de razões (vírus, problemas no disco rígido, ataque de crackers, etc.), tem-se como uma solução o *back up*.

Com respeito a ambiente de banco de dados, existem comandos apropriados para se fazer *back up* e recuperação, fornecidos pelo próprio Sistema de Gerenciamento de Banco de Dados (SGBD). Estes comandos podem ser usados isoladamente ou em *scripts* para realizar *back up* ou recuperação.

Este trabalho tem como objetivo mostrar como se faz um *back up* em ambiente PostgreSQL, OpenIngres (ou ingres II) e MySQL e também como recuperar os dados usando apenas comandos do próprio SGBD. Considerações sobre estes SGBDs em relação a *back up*, recuperação e freqüência de *back up* de bases de dados também serão enfocados.

Estes SGBDs foram escolhidos pelo fato do PostgreSQL e MySQL serem *freeware* e terem grande aceitação no mercado e o OpenIngres pelo fato de ser usado na Embrapa (e outras empresas de porte tais como Petrobrás, DuPont, etc.). O Oracle, embora líder de mercado, não será enfocado pois merece um documento a parte devido ao fato do *back up* e a recuperação serem mais detalhados (complexos).

Na literatura, são feitas comparações entre os SGBDs em vários quesitos (desempenho, velocidade de acesso, etc.). Porém, não é enfocado com detalhes o quesito "*back up* e recuperação". A ênfase é dada para outros quesitos, ficando a segurança em segundo plano. Assim, este trabalho também tem o mérito de fazer uma comparação quanto a procedimento de *back up* e recuperação, o que não é enfocado com detalhes na literatura.

SGBD MySQL

O SGBD MySQL é um servidor de base de dados baseado na linguagem de consulta SQL, a qual é a mais popular e difundida linguagem de consulta a bancos de dados no mundo. Este SGBD é multiusuário, e seus principais objetivos são a velocidade, robustez e facilidade de uso. Possui como características principais a facilidade de gerenciamento e o alto desempenho no que se refere a execução de consultas e manipulação de dados. Este SGBD é *freeware*, salvo se for usado para fins comerciais (MySQL AB, 2003).

Back up de uma base de dados *on-line* (comando *mysqldump*)

Com respeito a *back up* em ambiente MySQL, o mesmo é muito simples de se fazer. Sem perda de generalidade, admita que o projeto *Suporte* tenha uma base de dados *suportedb*. Suponha que o usuário *admin* tenha senha *admin#1* e seja dono da base *suportedb*. Para fazer o *back up*, basta o usuário *admin* executar o comando *mysqldump*. Um exemplo pode ser o comando a seguir, isto é,

mysqldump -u *admin* - p*admin#1* *--add-drop-table* *suportedb* > *backup.sql*

A opção -u serve para determinar o usuário (username) que se conectará à base de dados. A opção -p indica a senha do usuário. A senha e a opção -p devem formar uma palavra só (devem vir juntos).

Todos os dados da base de dados *suportedb* que pertencem ao usuário *admin* serão direcionados (saída de dados) para o arquivo *backup.sql*. Basicamente, este arquivo conterá comandos para a criação de cada uma das tabelas do banco (create table) e comandos para inserções de dados nas tabelas (insert into).

A opção *--add-drop-table* adiciona o comando '*drop table* < *tabela* > ' antes do comando '*create table* < *tabela* > ' no script gerado (*backup.sql*) e assim, no momento da restauração, cada tabela existente no banco será apagada (drop table) e será criada em seguida a respectiva tabela, conforme o script *backup.sql*. Por exemplo, suponha a tabela *hosts*, pertencente ao banco *suportedb*. O trecho do *script backup.sql* pertinente à tabela *hosts* seria:

```
DROP TABLE IF EXISTS hosts;
CREATE TABLE hosts (id_host int(4) not null primary key, nome varchar(15));
Insert into hosts values(1, 'norma');
Insert into hosts values(2,'aldebaran');
```

Assim, quando da restauração, a tabela *hosts* seria apagada e, em seguida, seria recriada e os dados seriam inseridos logo a seguir. Se a tabela não for removida

(*drop table*), vai ocorrer um erro (tentativa de criar tabela já existente e tentativa de inserção de dados já existentes).

Caso o usuário esteja em uma máquina que não seja a servidora, o mesmo procedimento já citado pode ser feito. Basta indicar o nome do host. Suponha que o servidor de SGBD MySQL esteja na máquina canopus. Assim, o usuário *admin* poderia executar o seguinte comando a partir de sua máquina:

mysqldump -u *admin* -h *canopus* -p*admin#1 --add-drop-table suportedb > backup.sql*

Caso o usuário admin queira apenas fazer *back up* de uma tabela (*hosts*, por exemplo) da base de dados *suportedb*, basta executar o comando:

mysqldump -u *admin* -h *canopus* -p*admin#1 --add-drop-table suportedb hosts > backup.sql*

No exemplo citado, a tabela *hosts* é adicionada logo após o nome da base de dados *suportedb*.

Para a recuperação da base de dados, ou de uma tabela, deve ser executado o seguinte comando geral:

mysql -u < nome do usuario > -p < senha > < nome da base > < nome da tabela > < < arquivo.sql >

Observe que agora o comando mudou de *mysqldump* para *mysql*, e o redirecionamento (entrada de dados) também. O procedimento para recuperação da base de dados *suportedb* (ou uma tabela) seria feito por apenas um comando, isto é:

/export/home/mysql/bin/mysql -u admin -padmin#1 suportedb < backup.sql

Os comandos mencionados foram de *back up* e recuperação de uma base de dados (*suportedb*). Se o DBA do banco quiser fazer *back up* de todas as bases existentes, pode ser feito através de um comando, isto é:

mysqldump -A --add-drop-table -u root -p < senha de root > > backupGeral.sql

A opção *-A* indica que todas as bases de dados em ambiente *mysql* devem ser armazenadas no arquivo *backupGeral.sql*. Assim, se houver as bases db, db1, db2,...,dbn, todas serão armazenadas no arquivo backupGeral.sql.

O comando para restauração é feito tal como foi mostrado para a base de dados *suportedb*. Supondo que a senha do usuário *dba (root)* seja *dba#1*, tem-se o seguinte comando para restauração da base:

```
/export/home/mysql/bin/mysql -u root -pdba#1 < backupGeral.sql
```

Caso seja necessário fazer o *back up* geral, mas separando cada base de dados, isto é, um *back up* para cada base de dados, tem-se o seguinte *script*, em ambiente Unix, admitindo-se que a senha do dba (usuário root) seja dba#1:

```
#!/bin/csh -f
mysql -u root -pdba#1 -e "show tables;" mysql > ./ backupGeral.sql

# a senha de root neste exemplo eh dba#1 e deve vir imediatamente apos
# a opcao -p (nao deve haver espaço em branco)
# a opcao -e indica que será executado um comando (show tables) na base de
# dados mysql (a qual contem todas as bases e usuarios do sgbd)
# em sintese, o comando acima verifica todas as bases de dados e direciona
# para o arquivo backupGeral.sql
#
# os comandos abaixo fazem o back up das bases de dados
# contidas em backupGeral.sql.

  foreach linha( `cat ./ backupGeral.sql` )
    set d = `date | awk '{print $1$2$3$6}'`
    set arq = `echo "./"$linha$d".txt"`
    set arq1 = `echo "./"$linha$d".txt.Z"`
    set dump = `echo " -u root -pdba#1 " $linha `
    mysqldump $dump > $arq
    compress $arq
    chmod 700 $arq1
  end
rm -rf ./ backupGeral.sql
```

Assim, o dba (usuário root) pode copiar todos os arquivos para *back up* de uma só vez, sem executar shutdown nos processos do banco. Esta tarefa (*script*) deve ser inserida em um agendador de tarefas, que para o ambiente Unix seria o crontab.

Back up físico (ou *off-line*)

Uma outra maneira de fazer *back up* no SGBD Mysql seria copiar os dados físicos do SGBD MySQL. Suponha, sem perda de generalidade, que o SGBD Mysql esteja instalado no diretório /export/home/mysql (diretório home do mysql). Os dados de cada base de dados ficam abaixo do diretório /export/home/mysql/data. Após se executar um shutdown nos processos do mysql (mysqld stop), basta copiar todos os arquivos que ficam abaixo do diretório /export/home/mysql/data para uma fita (ou um outro diretório) que se terá o *back up* de todo o banco. Após isto, inicialize o banco (mysqld start). O shutdown deve ser feito para garantir que exista

sincronismo em todo o banco, isto é, todas as operações pendentes são executadas e então o banco é desativado.

Um *script* para *back up* em fita das bases de dados, em ambiente Unix, seria:

```
#!/bin/csh -vf
/etc/init.d/mysql stop
tar -cvf /dev/rmt/0 /export/home/mysql/data
/etc/init.d/mysql start
```

No exemplo acima, /etc/init.d/mysql é um *script* de stop/start do mysql, que vem com o pacote de instalação do mysql.

O *back up* pode ser apenas de uma base, caso seja necessário. Suponha o caso da base de dados *suportedb*. Abaixo do diretório /export/home/mysql/data existe o diretório suportedb. Assim, o diretório /export/home/mysql/data/suportedb conteria os arquivos componentes da base do mysql. Um *script* para *back up* desta base seria:

```
#!/bin/csh vf
/etc/init.d/mysql stop
tar -cvf /dev/rmt/0 /export/home/mysql/data/suportedb
/etc/init.d/mysql start
```

A recuperação, neste caso, é muito simples. Suponha que a base de dados suportedb tenha sido corrompida. Basta copiar o conteúdo da fita, relativo ao *back up* de *suportedb*, no diretório /export/home/mysql/data/suportedb. Um *script* para isto, admitindo que a fita tenha somente o *back up* da base suportedb, seria:

```
#!/bin/csh -vf
/etc/init.d/mysql stop
cd /export/home/mysql/data/suportedb
tar -xvf /dev/rmt/0
/etc/init.d/mysql start
```

SGBD PostgreSQL

O PostgreSQL é um SGBD relacional e objeto (ORDBMS). Este SGBD é *freeware* e tem como características principais: compatibilidade com o padrão SQL ANSI, integridade referencial, suporte a linguagens procedurais, estabilidade. Além do PostgreSQL ser multiplataforma, possui também como características interessantes: herança, subselects, triggers, views, rules e suporte nativo a SSL. Mais informações sobre o PostgreSQL podem ser consultadas em PostgreSQL Global Development Group (2003).

Com relação ao *back up* e recuperação de dados, serão mostrados os comandos para fazer *back up* com o banco *on-line* (pg_dump, pg_dumpall, psql) e *off-line* (*back up* físico, usando o sistema operacional para copiar os arquivos que contêm os dados do banco de dados).

Back up de uma base dados *on-line* (comando *pg_dump*)

Este tipo de *back up* é feito com o banco em atividade (*on-line*), ou seja, não precisa ser executado um shutdown no banco. Desta forma, os usuários não ficam sem acesso às suas bases de dados. O *back up* é feito através do comando

pg_dump -h host and -p port -u - i dbname > backup.sql

No comando acima, *host* é o nome do server PostgreSQL desejado e *port* é o número da porta desejada para conexão. Se a opção da porta for ignorada, será feita a conexão na porta default (5432) do servidor, definida na instalação. Caso o usuário esteja querendo fazer *back up* no próprio servidor, a opção -h pode ser ignorada. A opção *-u* é usada para fornecer o prompt para username e a senha relativa à base de dados. A opção -i é para ignorar a diferença de versão entre o *back up* e o PostgreSQL (é útil quando a versão do PostgreSQL muda, para evitar qualquer incompatibilidade). A opção *dbname* é o nome da base de dados a ser armazenada.

Após o comando *pg_dump* ser executado, o arquivo *backup.sql* conterá todos os comandos SQL para reconstruir a base (drop table, create table, insert into, etc.). Para executar este comando, o usuário deverá ter permissão (ser o dono da base ou ter privilégios para isto). Um exemplo do *back up* da base *suportedb*, feita na máquina aldebaran, servidora de PostgreSQL, está descrito a seguir:

admin@aldebaran[52] pg_dump -u -i suportedb > backup.sql
User name: admin
Password:
pgsql@aldebaran[53]

Para que não ocorra o prompt relativo a username, pode-se usar a opção -U < username > . Assim, o comando poderia ser executado da seguinte maneira:

pg_dump -Uadmin -i suportedb > backup.sql
Password:
pgsql@aldebaran[53]

Para recuperar os dados, no servidor, basta executar o comando psql, isto é:

psql dbname < arquivo.sql

Antes de executar este comando, a base de dados dbname deverá ser criada, isto é:

dropbd -U < username > dbname	(destruir a base dbname antiga)
createdb -U < username > -W dbname	(criar a base dbname)
psql dbname < backup.sql	(executar a restauração)

Assim, para o caso da base *suportedb*, tem-se os seguintes comandos para a recuperação:

1. dropdb -Uadmin suportedb
2. createdb -Uadmin -W suportedb
Password:
CREATE DATABASE
3. psql suportedb < backup.sql

Se o tamanho da base de dados for muito grande, pode-se fazer o *back up* da seguinte maneira:

pg_dump -Uadmin -i suportedb | gzip > backup.sql.gz

Para restaurar, execute o comando

gunzip -c backup.sql.gz | psql suportedb

Caso seja necessário fazer *back up* apenas de uma tabela, basta executar o comando *pg_dump* com a opção -t, tal como no exemplo a seguir, no qual se deseja fazer *back up* da tabela *hosts*, a qual pertence à base de dados *suportedb*

pg_dump -Uadmin -i -t hosts suportedb > backup.sql

A recuperação é tal como da de uma base de dados, isto é:

dropdb -Uadmin suportedb
psql suportedb < backup.sql

Back up físico (ou *off-line*)

Admita que o diretório no qual os dados das bases de dados seja "data", o qual foi configurado na instalação através dos comandos:

```
root# mkdir /export/home/pgsql/data
root# chown pgsql /export/home/pgsql/data
root# su pgsql
pgsql$ initdb -D /export/home/pgsql/data
```

Para fazer o *back up*, faz-se necessário o seguinte:

1. shutdown nos processos do PostgreSQL

```
pg_ctl stop
```

2. cópia do file system /export/home/pgsql/data para um outro local (fita ou HD)

```
cd /export/home/pgsql/data
tar -cvf backup.sql.tar .
```

3. startup nos processos do SGBD

```
pg_ctl start -D /export/home/pgsql/data- l serverlog
```

Para fazer a recuperação da base de dados, basta fazer os seguintes passos:

1. shutdown nos processos do PostgreSQL

```
pg_ctl stop
```

2. extrair os dados do arquivo backupAnoMesDia.tar para o diretório /export/home/pgsql/data

```
cd /export/home/pgsql/data
tar -xvf backup.sql.tar .
```

3. inicializar (startup) os processos do SGBD

```
pg_ctl start -D /export/home/pgsql/data -l serverlog
```

Para fazer *back up* físico, o ideal é fazer em finais de semana ou durante a madrugada. Basta fazer um *script* para isto e usar o comando crontab (em ambiente Unix) para agendar sua execução. Um exemplo de *script* seria:

```
#/bin/csh -fv
/export/home/pgsql/bin/pg_ctl stop
cd /export/home/pgsql/data
tar -cvf /backup/backup.sql.tar .
/export/home/pgsql/bin/pg_ctl start -D /export/home/pgsql/data -l serverlog
```

PG_DUMPALL (*back up* de toda base dados *on-line*)

O comando pg_dump não é apropriado para o *back up* de muitas bases de dados. Para isto, tem-se o comando *pg_dumpall*. A execução deste comando é tal como o exemplo que segue:

pg_dumpall > backupTotalAnoMesDia.sql

A restauração do banco de dados (todas as bases) é feita pelo comando psql. Entretanto, é necessário que este comando seja executado pelo usuário dono dos processos do PostgreSQL.

psql < backupTotalAnoMesDia.sql

Se o tamanho da base de dados for muito grande, pode-se fazer o *back up* da seguinte maneira:

pg_dumpall | gzip > backupTotalAnoMesDia.sql.gz

Para restaurar, execute o comando

gunzip -c backupTotalAnoMesDia.sql.gz | psql

Back up em Ambiente OpenIngres

(válido também para Ingres II, sucessor do OpenIngres)

O OpenIngres é um SGBD de grande porte, capaz de suportar grande volume de dados. Este SGBD não é um dos mais usados no mercado, mas é encontrado em grandes empresas tais como Embrapa, Petrobrás, Bradesco, DuPont, Embratel, Universidade Federal de Goiás e outras grandes empresas do país e do mundo. Mais detalhes sobre o OpenIngres podem ser consultados em Computer Associates International (2003).

Para executar um backup de uma base de dados qualquer bastam poucos comandos. Serão mostradas duas formas de *back up*: *back up on-line* (feita através dos comandos *unloaddb* e *ckpdb*) e *back up off-line*.

Back up de uma base dados *on-line* (comando *unloaddb*)

O *back up* completo é feito usando-se o comando *unloaddb*. Este *back up*, considerando a base de dados *suportedb*, pode ser resumido conforme descrito a seguir, supondo os processos do OpenIngres ativos:

1. Em um diretório qualquer (por exemplo, /tmp/suportedb), execute o comando

unloaddb -c suportedb

A opção -c é para a transferência ser no modo ASCII. Sem esta opção, a transferência (cópia) será no modo binário.

2. Em seguida, execute o comando

Unload.ing

Será gerada uma série de arquivos. Todos estes arquivos deverão ser copiados para fita ou para outra mídia confiável para *back up*.

Deve-se guardar o nome do diretório /tmp/suportedb, pois caso seja necessário uma recuperação da base de dados, esta recuperação deve ser feita a partir deste diretório.

Um exemplo de *script* de *back up*, tal como descrito acima, para a base de dados *suportedb*, pode ser

```
#!/bin/csh
#################################################################
#############
#
# Backup da base suportedb
#
####################

    setenv II_SYSTEM /home
    set path = ($II_SYSTEM/ingres/{bin,utility,lib}$path)
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/home/ingres/lib
    set term = vt100
    ingsetenv II_DATE_FORMAT GERMAN
    mkdir /tmp/suportedb
    cd /tmp/suportedb
    unloaddb -c  suportedb
    unload.ing
    /bin/rm -f /tmp/backup.tar
    tar -cvf /tmp/backup.tar .
    compress /tmp/backup.tar
    set a = `date | awk '{print $3$2$6}'`
    set arq = `echo "bcpsuportedb"$a".tar.Z"`
    mv /tmp/backup.tar.Z /home/ingres/gerencia/$arq
```

Após este *script* ter sido executado, o resultado vai para a pasta /home/ingres/gerencia com o nome de bcpsuportedb21Jul2003.tar.Z (21Jul2003 é a data de quando foi feito o *back up*).

Para fazer a recuperação, os passos são simples e podem ser descritos da seguinte forma:

1. Certificar que a base antiga *suportedb* não existe e então criar uma nova

createdb -uadmin suportedb

Caso a base exista, esta deve ser removida. Basta executar o comando

destroydb -uadmin suportedb

no prompt do sistema operacional.

2. Executar o comando

reload.ing

(reload.ing e unload.ing, dois dentre vários arquivos gerados no *back up*, devem ter permissão de execução, caso o sistema operacional seja Unix ou compatível). Após isto, a base de dados será a mesma que havia no momento da criação do *back up*.

Resumindo, para fazer *back up* usando o comando *unloaddb*, tem-se o seguinte *script*

```
mkdir /tmp/suportedb
cd /tmp/suportedb
unloaddb -c suportedb
unload.ing
```

Para restaurar a base de dados, no caso *suportedb*, faça assim;

```
cd /tmp/suportedb
destroydb -uadmin suportedb
createdb -uadmin suportedb
reload.ing
```

Back up dinâmico

Este tipo de *back up* é feito para recuperar a base de tal forma que os dados sejam atualizados até momentos antes do erro ou crash. Suponha que as 12h45 ocorreu um problema na base. É possível recuperar os dados até momentos antes (12h43, por exemplo).

Para entender como funciona este *back up*, faz-se necessário definir alguns termos, isto é, *checkpoint, journal*, e *rollforwarddb*.

Chekpoint faz com que todas as transações realizadas (commit) sejam efetivadas no banco. O comando para isto, considerando a base *suportedb* é

ckpdb -d *suportedb*

Assim, a partir da execução deste comando, tem-se que todas as transações com commit foram gravadas no banco de dados. Desta forma, o *checkpoint* fornecerá uma imagem atualizada do banco de dados no momento da execução do comando ckpdb.

Journal é usado para *back up* dinâmico, isto é, recuperar a base até momentos antes do "crash" ou perda de dados. Com o *Journal*, todas as mudanças feitas após o *checkpoint* são armazenadas. Assim, a partir do último *checkpoint* aplicado mais as "tabelas de jornalização", é possível recuperar os dados até momentos antes da perda de dados. Para que o processo de *journal* funcione, basta executar o comando

ckpdb -d +j *suportedb*

A opção -d remove o último chepoint e atualiza. A opção +j aciona o processo de journal.

O comando *rollforwarddb* é usado quando da recuperação da base "jornalizada" (mais *checkpoint*). O comando para isto é

rollforwarddb *suportedb*

Após a execução deste comando, a base de dados é recuperada até próximo do erro ou crash.

Back up físico (ou *off-line*)

Este *back up* é tal como foi mencionado para os SGBDs MySQL e PostgreSQL. Basta executar um *shutdown* nos processos do banco, fazer cópia física dos

arquivos e depois inicializar os processos do SGBD. Isto poderia ser feito da seguinte forma, levando-se em conta a base de dados *suportedb*, e que o diretório onde ficam os arquivos relacionados a esta base fiquem em /export/home/ingres/data/default (que pode ser visto pelo comando *ingprenv*, o qual mostra as variáveis de ambiente do ingres, e entre estas está a variável *ii_database*, que contém a localização do diretório onde estão os arquivos da base de dados):

```
ingstop
cd /export/home/ingres/data/default
tar -cvf /dev/rmt/0 ./suportedb
ingstart
```

para fazer *back up off-line* de todo o banco, basta copiar todos os diretórios que estiverem abaixo do diretório *default*, isto é:

```
ingstop
cd /export/home/ingres/data/default
tar -cvf /dev/rmt/0 .
ingstart
```

Assim, tem-se o *back up off-line* do banco (*back up* com processos do SGBD OpenIngres inativos).

Para a recuperação da base de dados *suportedb*, basta copiar os dados da fita para o diretório /export/home/ingres/data/default. O mesmo raciocínio se aplica ao *back up* geral. O *script* seria o mesmo feito para o *back up*, com a alteração da opção -c pela -x do comando tar.

```
ingstop
cd /export/home/ingres/data/default
tar -xvf /dev/rmt/0 ./suportedb
ingstart
```

Importante: o comando *ingstop* deverá ser executado sem as opções *-kill (ingstop -kill)* e *-force (ingstop -force)* para evitar inconsistência na base e conseqüente erro no momento da recuperação da base de dados.

Copydb

O comando copydb é usado para copiar tabelas específicas.

Caso seja necessário o *back up* de apenas tabelas, pode-se usar o comando *copydb.* Utilizando a base de dados *suportedb*, o uso do comando *copydb* pode ser exemplificado da seguinte forma:

1.Crie um diretório temporário
mkdir /tmp/copia
cd /tmp/copia

2. Execute o comando *copydb*
copydb suportedb hosts

3. Após o passo 2, são criados os scripts sql copy.in e copy.out. Execute script copy.out, isto é:
sql suportedb < copy.out

Para fazer a recuperação da tabela, execute os seguintes passos:

1. r para o diretório /tmp/copia

2. Copiar os arquivos do *back up* neste diretório

3. Executar o comando
sql suportedb < copy.in

Após a execução dos passos citados, a tabela estará recuperada.

Freqüência de *Back up*

A freqüência de *back up* deve ser específica para cada projeto ou instituição. Bancos (instituição financeira), indústria (produção não pode parar), grandes redes de lojas e outros segmentos do comércio e indústria não toleram perdas de dados por mais do que alguns minutos. Uma parada na produção de uma indústria por algumas horas, por exemplo, pode significar um prejuízo de milhões de reais. Entretanto, existem instituições que toleram uma perda de um dia ou mais (as atualizações no banco de dados não são constantes). Um exemplo é o software de avaliação anual de funcionários de uma instituição. Os dados são somente atualizados algumas vezes no ano. Neste caso, o *back up* deverá ser feito com alta freqüência durante o período de avaliação. Durante os demais períodos, a freqüência pode ser menor, visto que terá pouca atualização. Desta forma, a periodicidade do *back up* vai depender do quão freqüente a base é atualizada e qual impacto produzirá caso o banco fique indisponível ao usuário por minutos, horas ou dias.

Visto que a periodicidade do *back up* vai depender de uma série de fatores, tal já exposto, neste tópico serão apresentadas algumas sugestões de *back up*, as quais deverão ser adaptadas a cada projeto, conforme suas necessidades.

Para determinar com que freqüência o *back up* deve ser feito, deve-se levar em conta o fato de que o usuário não deverá ter o acesso ao banco de dados prejudicado. Assim, o *back up* dos SGBDs MySQL, OpenIngres e PostgreSQL, quando for feito em horário de expediente, deverá ser *on-line*. O *back up* poderá ser *off-line* desde que seja feito em horário de pouco ou nenhum acesso (final de semana, por exemplo).

Quando a base de dados é muito grande (acima de alguns Giga Bytes), o ideal é fazer *back up* no final de semana (*off-line*) por que a cópia de arquivos é mais rápida e um *script* pode facilmente fazer isto. Esta cópia pode ser feita diretamente em uma unidade de fita, para não ocupar espaço em disco. Durante a semana, podem ser feitos *back up* diários, com o banco em estado *on-line*. Para não comprometer o desempenho do banco de dados, o ideal é fazer o *back up* durante a madrugada.

Resumindo, o *back up* para base de dados com muitos GBs pode ser assim:

- *sábado ou domingo*: *back up off-line* (shutdown no banco, cópia da base para a fita, start no banco);

- *segunda-feira à sexta-feira*: *back up* da base de dados (durante a madrugada) *on-line*, ou em períodos de pouco acesso.

Para bases de dados cuja freqüência de atualização seja baixa, pode-se fazer *back up on-line* durante alguns dias por semana, de madrugada. Isto pode ser feito através do agendador de tarefas (se for ambiente windows) ou pelo comando crontab (se for ambiente Unix). Se a base for muito grande, o ideal é também fazer *back up off-line* no final de semana pois é mais rápido recuperar um *back up off-line* (basta executar *shutdown*, copiar os arquivos e reiniciar o SGBD) do que o *back up on-line* (admitindo-se que a base de dados esteja indisponível por algum crash ou perda de dados). Porém, fica a critério do dba responsável pelo banco de dados o que proceder.

Se o SGBD tiver n bases de dados e a soma de todas não couber em uma fita, deve-se dividir o *back up* para as n bases de dados nos dias da semana de tal forma que cada base seja armazenada em fita pelo menos duas vezes por semana e fazer *back up on-line* de madrugada (hora em que supostamente o número de acessos ao banco é menor). Se mesmo após a divisão não for possível armazenar todas as bases na semana, deverá ser aumentado o número de leitoras de fitas (inseridas em outras máquinas) e o *back up* também deverá ser feito remotamente, isto é, acessar a base a partir das máquinas que contenham leitoras de fita e fazer o *back up*. Em suma, quanto mais bases de dados existir, mais meios de armazenamento deverão estar disponíveis para realizar o *back up* de todas as bases de dados.

Para exemplificar o que foi dito, sejam as bases de dados (BD) *agencia, suporte, agroclima, bioinfo, producao, ainfo e mipweb*. Suponha que todas estas bases

não possam ser armazenadas em uma fita em um mesmo dia. O *back up* poderia ser feito como demonstrado na Tabela 1.

Tabela 1. *Back up* das bases de dados durante a semana.

Dia/BD	*Agencia*	*Agroclima*	*Ainfo*	*Bioinfo*	*Mipweb*	*Producao*	*Suporte*
Segunda-feira	X[1]	X					X
Terça-feira			X	X		X	
Quarta-feira	X	X			X		
Quinta-feira						X	X
Sexta-feira			X	X	X		
Sábado	*Off-line*	*Off-line*					*Off-line*
Domingo			*Off-line*	*Off-line*	*Off-line*	*Off-line*	

[1] Na tabela 1, x significa "executar o *back up*".

Se a uma fita ou o meio físico escolhido não puder conter este planejamento, então deverão existir mais dispositivos para *back up*, para garantir que exista pelo menos dois *back up* para cada base de dados.

Existe ainda o *back up* de banco de dados que não pode ter perdas de dados por mais do que alguns minutos ou horas. Neste caso, o ideal é usar um SGBD tal como o OpenIngres, que realiza *back up* dinâmico (*checkpoint, journal* e *rollforward*) pois, tal como foi explicado, é possível recuperar até momentos antes da perda. Para o caso dos SGBDs MySQL e PostgreSQL, o ideal é fazer um espelhamento. Assim, se ocorrer algum problema, o outro servidor de banco de dados assume, sem prejuízo para o usuário e o servidor primário poderá ser restaurado enquanto o secundário atua.

O espelhamento é indicado para todos os SGBDs nos quais os processos do banco de dados não podem parar ou ainda, para evitar perda de dados por mais do que alguns minutos. Porém, é uma solução mais cara, a qual pode ser justificada, conforme já mencionado para o caso de bancos, indústrias (produção) e outros segmentos da sociedade. Outra vantagem do espelhamento é quando o servidor primário tiver problemas de hardware. Neste caso, tendo-se um servidor secundário (espelhamento), o serviço de banco de dados não ficará parado e tem-se um tempo para restaurar o servidor primário.

Assim, cada usuário deverá avaliar a melhor forma de *back up* conforme a necessidade. A Tabela 2 a seguir é um exemplo de como decidir, conforme a necessidade (atualização da base versus tolerância a perda de dados), a forma de *back up* a ser adotada. Este é um exemplo e pode não ser consenso entre os leitores, porém é motivação para avaliar a melhor maneira de fazer *back up* (criar sua própria tabela com os itens mais relevantes a serem considerados e ver a melhor opção), considerando-se possíveis perdas e a tolerância a elas.

Tabela 2. Que tipo de *back up* escolher, considerando a atualização e tolerância a perdas.

Atualização versus tolerância	*Tolera perdas por grande período*	*Tolera perdas por curto período (menos do que 2 dias)*	*Tolera perdas por até 12h*	*Não tolera perdas (ou até 1h)*
Atualizações ou consultas freqüentes	*Back up off-line* ou *on-line*	Espelhamento *Back up* dinâmico	Espelhamento e *Back up* dinâmico	Espelhamento e *Back up* dinâmico
Atualizações ou consultas esporádicas	*Back up off-line* ou *on-line* (um ou dois por semana)	*Back up off-line* ou *on-line*	Espelhamento e *Back up* dinâmico ou *Back up off-line* ou *on-line* a cada 12h	Espelhamento e *Back up* dinâmico

Comparações entre os *Back up* de cada SGBD

Basicamente, os 3 SGBDs oferecem opções de *back up off-line* e *on-line*. Porém, o OpenIngres oferece uma vantagem adicional, isto é, pode recuperar toda a base até o mais próximo possível do momento da perda de dados. Desta forma, o trabalho diário não seria perdido (em certos casos, como em uma instituição financeira, algumas horas faz muita diferença). Assim, bancos de dados *freeware* não são apropriados, em termos de *back up*, para instituições que manipulam grande volume de dados e que não podem perder dados a partir de minutos ou horas antes de um crash ou perda de dados, salvo se houver um espelhamento.

Com relação aos SGBDs PostgreSQL e MySQL, os dois implementam mecanismos similares de *back up*. A recuperação também é muito similar. Os mecanismos de *back up* são simples, porém não se consegue recuperação de dados a um ponto próximo quando da perda de dados, a não ser que o *back up* tenha sido feito pouco antes da perda, o que é difícil de ocorrer. Caso um destes SGBDs sejam usados para operações onde perdas de dados não sejam toleradas, a freqüência de *back up* deverá ser alta ou então deverá haver um espelhamento.

A Tabela 3 resume a comparação feita.

Tabela 3. SGBD versus tipos de *back up*.

SGBD	*Back up off-line*	*Back up on-line (snapshot)*	*Back up on-line (dinâmico)*	*Recuperação complexa*
OpenIngres	sim	sim	sim	não
MySQL	sim	sim	não	não
PostgreSQL	sim	sim	não	não

Na Tabela 3, *snapshot* significa uma "imagem da base de dados" no momento do *back up* e *dinâmico* significa a possibilidade de recuperação momentos antes da perda de dados (ou crash do sistema).

Conclusões

Neste trabalho, procurou-se mostrar como se faz *back up*/recuperação usando comandos do próprio SGBD e do sistema operacional. Além disso, o *back up* pode ser feito automaticamente através de *scripts*, ativados por um agendador de tarefas.

O *back up* pode ser realizado com os processos do SGBD *on-line* ou *off-line*. A vantagem do modo *on-line* é que o SGBD sempre estará disponível ao usuário.

A vantagem do *back up* com o SGBD *off-line* é que é simplesmente uma cópia de arquivos, porém o banco de dados fica indisponível durante a cópia. Este tipo de *back up* também é útil quando se deseja copiar a base de dados de um servidor para outro servidor de banco de dados. Assim, basta copiar os arquivos de um servidor (relativo às bases de dados) para o outro servidor (no diretório no qual ficarão os arquivos relacionados às bases de dados). Este procedimento vale a pena, mesmo tendo que executar shutdown/start em cada um dos server de SGBD, por ser simples.

O OpenIngres tem uma pequena vantagem sobre o MySQL e o PostgreSQL pois implementa uma funcionalidade muito útil para o *back up*, que é a recuperação dos dados até momentos antes da perda. A desvantagem é que consome mais memória, porém é muito útil para SGBDs que não podem perder dados, mesmo que por um curto período de tempo, como é o caso de instituições financeiras (bancos).

A freqüência de *back up* dependerá das características do serviço que o banco deverá atender (freqüência de atualização, consultas, etc.). No caso de instituições financeiras, o serviço do banco de dados de forma alguma poderá estar fora do ar. É indicado um SGBD que possa recuperar os dados até alguns momentos antes da perda ou ainda, um espelhamento (neste caso, o PostgreSQL e o MySQL podem ser utilizados). Para banco de dados que tenham baixa freqüência de utilização, a periodicidade pode ser menor, bastando que se tenha o *back up* em algumas vezes por semana, desde que seja confiável.

Referências Bibliográficas

COMPUTER ASSOCIATES INTERNATIONAL. Computer Associates [home page]. Disponível em: < www.ca.com > . Acesso em: 06 jul. 2003.

MYSQL AB. MySQL: the world's most popular open source database. Disponível em: < http://www.mysql.com/ > . Acesso em: 06 jul. 2003.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL. Disponível em: < http://www.postgresql.org/ > . Acesso em: 06 jul. 2003.

Ministério da Agricultura, Pecuária e Abastecimento
Governo Federal