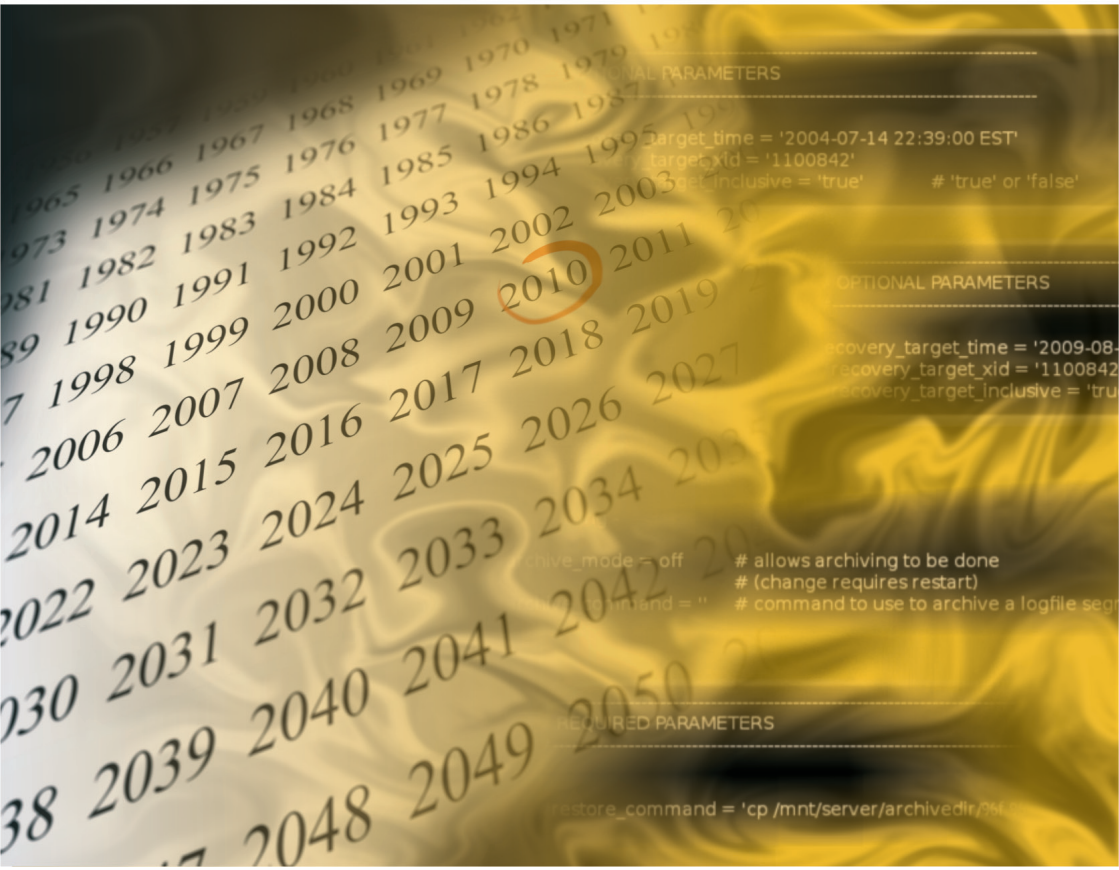
Documentos

ISSN 1677-9274
Dezembro, 2009
97

Backup Incremental e Recuperação Cronológica de Dados no PostgreSQL



Embrapa

ISSN 1677-9274
Dezembro, 2009

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

Documentos 97

Backup Incremental e Recuperação Cronológica de Dados no PostgreSQL

Evandro Porto de Souza

Embrapa Informática Agropecuária
Campinas, SP
2009

Embrapa Informática Agropecuária
Av. André Tosello, 209 - Barão Geraldo
Caixa Postal 6041 - 13083-886 - Campinas, SP
Fone: (19) 3211-5700 - Fax: (19) 3211-5754
www.cnptia.embrapa.br
sac@cnptia.embrapa.br

1ª edição on-line 2009

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Souza, Evandro Porto de

Backup incremental e recuperação cronológica de dados no PostgreSQL / Evandro Porto de Souza. - Campinas : Embrapa Informática Agropecuária, 2009.

24 p. il. - (Documentos / Embrapa Informática Agropecuária, ISSN 1677-9274; 97).

1. Recuperação de dados. 2. Point-in-Time Recovery. 3. Backup. 4. PostgreSQL. 5. Banco de Dados. 6. Database. 7. SGBD. I. Título. II. Série.

CDD 005.86 21 ed.

Autor

Evandro Porto de Souza

Bacharel em Ciência da Computação,
Analista B - Administração de Banco de Dados da
Embrapa Informática Agropecuária
Av. André Tosello, 209, Barão Geraldo
Caixa Postal 6041 - 13083-886 - Campinas, SP
Telefone: (19) 3211-5772
e-mail: evandro@cnptia.embrapa.br

Apresentação

Este documento tem o propósito de apresentar uma solução de recuperação cronológica de dados para PostgreSQL, que possibilite recuperar uma base de dados por completo ou a um determinado instante na linha do tempo.

O trabalho a ser apresentado consiste no estudo e implementação do Mecanismo *Point-in-Time Recovery* do Sistema Gerenciador de Banco de Dados PostgreSQL, bem como a configuração necessária para a sua execução.

Durante o desenvolvimento deste trabalho, serão apresentados os procedimentos utilizados, os comandos necessários, bem como os resultados obtidos e as considerações, positivas e negativas, envolvendo todo o processo.

É esperado que, ao seu final, tenha-se em mãos um guia que facilite, contribua e oriente outros usuários em como proceder em situações similares.

Kleber Xavier Sampaio de Souza
Chefe-Geral
Embrapa Informática Agropecuária

Sumário

Backup incremental e recuperação cronológica de dados no PostgreSQL

Evandro Porto de Souza

Introdução

Este documento consiste no estudo, análise e identificação de uma solução voltada para a recuperação de dados no Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL.

A solução em recuperação de dados, aqui apresentada, foi testada e direcionada para o sistema operacional Linux Ubuntu, porém a mesma é compatível com outras distribuições, bem como passível de ser implementada e executada em outros sistemas operacionais, nos quais funcionam o SGBD PostgreSQL, resguardadas as suas respectivas peculiaridades, ou, basicamente, a nomenclatura e o formato dos comandos empregados relativos ao sistema operacional adotado.

Objetivo

Apresentar uma solução de recuperação de dados para PostgreSQL, que permita retroceder ou restaurar uma base de dados a um determinado ponto ou momento desejado.

Backup incremental e mecanismo PITR[1]

Quando o assunto é banco de dados, é imprescindível que se tenha um plano de manutenção e *backup*[2], assim como a garantia de que este esteja sempre disponível. No PostgreSQL, os tipos de *backup* mais utilizados são: o lógico (que se dá com a geração de arquivo por meio do utilitário "*pg_dump*" e de forma on-line, ou seja, com o serviço do SGBD em execução) e o físico (que consiste na cópia do diretório de dados "*pgdata*" e de maneira off-line, ou seja, com o serviço do SGBD parado). Há, ainda, a modalidade de *backup* incremental, não tão utilizado, mas em contrapartida, oferece uma maior precisão quanto ao instante de recuperação dos dados.

Com o *backup* lógico tradicional gerado a partir do utilitário "*pg_dump*", é possível apenas recuperar os dados pertinentes ao instante em que a cópia de segurança foi realizada, com o banco sendo restaurado exatamente como o mesmo se encontrava. Essa modalidade é extremamente rápida e fácil de se manusear. Entretanto, esta não permite recuperar dados até um determinado ponto na dimensão tempo. Hipoteticamente falando, caso um *backup* fosse realizado com o "*pg_dump*" às 3:00, e no decorrer do dia, mais precisamente às 16:30 e houvesse uma exclusão indesejável de uma determinada tabela, somente seria possível a recuperação do banco de dados com a situação das 3:00, ou seja, todas as inclusões, alterações e/ou exclusões corretas, ocorridas durante o dia, seriam perdidas.

No PostgreSQL, o *backup* incremental, utilizando o mecanismo *Point-in-Time Recovery* (PITR), oferece como grande diferencial em relação aos demais modelos, uma solução para o problema descrito anteriormente, ou seja, a possibilidade de se recuperar dados, retrocedendo até um momento específico na linha do tempo.

Mecanismo *Point-in-Time Recovery*

Toda e qualquer transação de alteração de dados no SGBD PostgreSQL, antes de efetivada a gravação nos arquivos de dados do servidor, é registrada em uma área específica denominada de Registro Prévio da Escrita

[1] *Point-in-Time Recovery*

[2] Termo em inglês para "cópia de segurança", comumente utilizado entre profissionais de informática.

ou, do inglês WAL (*Write Ahead Log*). A função da área de WAL é garantir a consistência e segurança quanto à gravação dos dados em disco, principalmente em situações adversas, como queda de eletricidade, e propiciar um aumento de desempenho por meio da redução significativa do número de escritas em disco, já que esta representa uma operação de alto custo e extremamente onerosa ao sistema operacional.

O mecanismo *Point-in-Time Recovery* (PITR) consiste na geração de uma cópia física completa dos dados e no arquivamento de *log*[3] das transações realizadas no banco de dados após a geração do *backup* inicial, para sua posterior restauração.

É importante ressaltar que o mecanismo PITR apenas garantirá a possibilidade de recuperação de dados, a partir do momento em que as três operações cruciais sejam executadas, são elas: habilitar o arquivamento de log das transações (WAL); realizar a cópia de segurança com identificador de base inicial; e armazenar os arquivos de *log* das transações realizadas (WAL) após o *backup* inicial. É imprescindível a ocorrência de tais ações para o funcionamento desse recurso.

Configuração do arquivo "*postgresql.conf*"

Para utilizar o mecanismo PITR do servidor PostgreSQL, faz-se necessário a adoção de alguns procedimentos e configurações prévias, conforme apresentadas a seguir.

Dando início, deve-se abrir um terminal de linha de comando ou console, e efetuar *login* com o usuário "*postgres*" (administrador do servidor PostgreSQL), conforme a Figura 1, com o qual serão executados todos os comandos a seguir.

```
$su - postgres
```

Figura 1. *Login* do usuário "*postgres*".

[3] *Log*: Arquivo de registro das transações que geraram modificações na Base de Dados.

A seguir, é preciso criar o diretório onde serão armazenados os arquivos de *log* (WAL), conforme Figura 2. O mesmo poderá ser criado em qualquer caminho desejado (como por exemplo: "*/tmp/wal*"), porém, é recomendado que esse diretório esteja localizado em outro disco físico ou outra partição diferente daquele em que esteja alocado o sistema operacional, visando, com isso, obter maior segurança dos dados, uma vez que a perda de tais arquivos resultará na impossibilidade de recuperação de dados e, consequentemente, implicará no não funcionamento do mecanismo PITR.

```
$mkdir /home/postgres/8.3/backup/wal/
```

Figura 2. Criar diretório para os arquivos de *log* WAL.

Uma vez criado o repositório acima, deve-se ativar o modo de arquivamento do WAL e direcioná-lo para tal repositório. Para isso, faz-se necessário abrir e editar o arquivo de configurações "*postgresql.conf*", localizado, geralmente, no diretório principal do PostgreSQL ("*/etc/postgresql/8.3/main/*") ou no diretório de dados do PostgreSQL ("*pgdata*", disponível em "*/var/lib/postgresql/8.3/main/*" ou no caminho definido pelo usuário durante a instalação). Para tanto, será adotado neste documento, o uso do utilitário de terminal "vim" (Vi-Improved), conforme apresentado na Figura 3. Entretanto, poderá ser utilizado qualquer outro editor de texto de sua preferência.

```
$vim /home/postgres/8.3/pgdata/postgresql.conf
```

Figura 3. Abrir arquivo de configurações "*postgresql.conf*".

Dentro do arquivo "*postgresql.conf*", localize o item "*Archiving*" na área de configurações relativos à "*WRITE AHEAD LOG*", conforme Figura 4.

Localizado o trecho de configuração mencionado acima, deve-se editar os dois parâmetros apresentados na Figura 4. Assim, deve-se alterar o valor do parâmetro "*archive_mode*" de "*off*" para "*on*", bem como retirar o caractere de identificação de comentário "*#*" (cerquilha). Quanto ao parâmetro

```
# - Archiving -

#archive_mode = off        # allows archiving to be done
                           # (change requires restart)
#archive_command = ''     # command to use to archive a logfile segment
```

Figura 4. Arquivamento de WAL no arquivo "*postgresql.conf*".

"*archive_command*", deve-se inserir neste, o comando do sistema operacional (aqui, será utilizado o comando "cp") o qual realizará as cópias dos arquivos de *log*, como também deve-se excluir o caractere de identificação de comentário "#" dessa linha, conforme Figura 5.

```
# - Archiving -

archive_mode = on

archive_command = 'cp %p /home/postgres/8.3/backup/wal/%f'
```

Figura 5. Ativando o arquivamento de *log* WAL no arquivo "*postgresql.conf*".

O primeiro parâmetro (*archive_mode*) habilita o modo de arquivamento de *log* do WAL, enquanto que o segundo (*archive_command*) define o comando a ser passado para o interpretador para que seja realizada a cópia de segurança do segmento da série de arquivos de log WAL quando este é completado para um local desejado. Os caracteres "*%p*" presentes na linha de comando serão substituídos no interpretador pelo caminho do arquivo cuja cópia de segurança será feita, e os caracteres "*%f*" serão substituídos no interpretador pelo nome do arquivo a ser copiado. Sem essa configuração, o servidor PostgreSQL irá descartar os arquivos de *log* do WAL, sempre que o segmento for completado e as transações forem efetivadas em disco, o que torna impossível a recuperação de dados ou o retorno à situação de um determinado momento do banco de dados.

Por fim, é necessário, para garantir que tais mudanças estejam em funcionamento, reiniciar o serviço do SGBD PostgreSQL, por meio do

utilitário “pg_ctl”. Devendo ser executadas as opções “*stop*” (Figura 6) e a seguir “*start*” (Figura 7), ou pura e simplesmente a opção “*restart*”, ou ainda, reiniciar o servidor, a partir do script de inicialização automática do PostgreSQL disponível em “/etc/init.d/”, com as mesmas opções anteriormente mencionadas.”

```
$pg_ctl stop
```

Figura 6. Parada do serviço do PostgreSQL.

```
$pg_ctl start
```

Figura 7. Inicialização do serviço do PostgreSQL.

Criação de *Backup*-base

Os passos do item anterior asseguram o funcionamento da tarefa de arquivamento de *log* WAL e armazenamento dos arquivos no local pré-definido, entretanto, para permitir a possibilidade de recuperação de dados com base nos arquivos de *log* armazenados é necessário que seja gerada uma cópia de segurança inicial do banco de dados, a qual será utilizada como base para uma futura recuperação de dados utilizando os arquivos de *log*.

Antes de gerar a cópia de segurança propriamente dita, acesse o banco de dados, preferencialmente via terminal iterativo “*psql*” do PostgreSQL, com o superusuário “*postgres*” ou outro usuário que tenha permissão do grupo “*postgres*”, conforme apresentado na Figura 8, onde na opção “-U” deve ser fornecido o usuário que realizará tal acesso. Em instalações padrões, pode ser usado também simplesmente o comando “*psql*”.

```
$psql -U postgres
```

Figura 8. Acesso ao banco de dados via terminal “*psql*”.

Uma vez realizado o acesso ao terminal "*psql*", utilize a função "*pg_start_backup*", conforme apresentado na Figura 9. Esse comando irá gerar um arquivo denominado de "*backup_label*" que contém informações a respeito do *backup* a ser realizado, isso permitirá que o servidor PostgreSQL, durante uma possível recuperação de dados, identifique o ponto exato de onde teve início a cópia de segurança para que possa garantir a total compatibilidade e continuidade entre o *backup*-base e os arquivos de *log* das transações realizadas posteriormente. O texto "rotulo_*backup*_base", passado como parâmetro à função "*pg_start_backup*", é um texto ou rótulo para identificação da cópia de segurança, e é de livre escrita ao usuário. O resultado retornado pela função, corresponde a um identificador interno do servidor PostgreSQL, não tendo nenhuma relevância ao usuário, uma vez que o processo é transparente ao mesmo.

```
# SELECT pg_start_backup('rotulo_backup_base');
```

Figura 9. Função para identificar início do *backup.*

Acionada a função acima, deve-se, a seguir, desconectar-se do terminal "*psql*" (para isso, utilize o comando "*q*") e iniciar a cópia de segurança do diretório de dados do PostgreSQL. Para realizar essa operação, acesse o caminho onde se localiza o diretório de dados, conforme apresentado na Figura 10. Lembrando que, neste trabalho, o diretório de dados é denominado de "*pgdata*" e o mesmo encontra-se alocado no caminho "*/home/postgres/8.3/*", entretanto, a sua denominação e o seu caminho podem variar de acordo com o tipo de instalação (pacote ou código-fonte) do sistema operacional utilizado (Linux, Windows, entre outros), da distribuição adotada (para o caso do Linux) e do desejo do próprio usuário que poderá personalizar sua instalação. Em instalações padrões no Linux Ubuntu, o diretório de dados do PostgreSQL 8.x tem a denominação de "main" e encontra-se no caminho "*/var/lib/postgresql/8.x/*".

```
$cd /home/postgres/8.3/
```

Figura 10. Acesso ao diretório raiz do PostgreSQL.

Dentro do diretório principal, utilize o compactador de sua preferência, neste trabalho foi adotado o uso conjunto dos aplicativos de linha de comando TAR (empacotador de arquivos) e GZIP (compressor de arquivos). A Figura 11 apresenta o comando a ser executado para empacotar e comprimir o diretório "pgdata" e armazenar o arquivo gerado ("*pgdata.tar.gz*") no local desejado.

Figura 11. *Backup* do diretório PGDATA usando TAR e GZIP.

Uma vez gerada a cópia de segurança do diretório de dados, há ainda um último passo que consiste em informar ao servidor PostgreSQL sobre o fim do processo de *backup*, para isso acesse novamente o terminal "*psql*" (Figura 8) e em seguida execute a função "*pg_stop_backup*", conforme apresentado na Figura 12. Assim como na função anterior (Figura 9), o resultado retornado por esta, também corresponde a um identificador interno do servidor PostgreSQLe não tem nenhuma relevância para o usuário.

Figura 12. Função para identificar fim do *backup.*

Pronto, resta somente iniciar novamente o serviço do PostgreSQL, conforme já apresentado na Figura 7. De posse do *backup*-base já criado e dos arquivos de *log WAL* que serão criados em decorrência das transações que venham a alterar a base de dados, tem se a ideia de "*Backup* incremental" no PostgreSQL. Esse recurso associado ao Mecanismo PITR que, a partir desse momento, já encontra-se habilitado, permite a recuperação total ou a recuperação cronológica de dados.

Em bancos de dados com grande volume de transações de manipulação de dados (*insert*, *update* e *delete*), os arquivos de *log* WAL podem crescer consideravelmente. Portanto, atente-se ao espaço de armazenamento disponível e, periodicamente, caso necessário, exclua os arquivos de *log*

WAL e refaça os procedimentos do item “Criação de *Backup*-base” (a partir da Figura 8 até o parágrafo anterior).

O *backup* incremental apresentado aqui é de grande utilidade e é um recurso de segurança mais voltado para periodicidades menores (por exemplo, semanal ou mensal) haja vista a sua exigência por maior espaço de armazenamento, bem como o fato de não ser muito usual ou fazer muito sentido, a necessidade de se retroceder a situação do banco de dados a um período maior do que um mês. Essa ocorrência seria um caso excepcional e muito improvável, já que implicaria na perda de muitos dados, que deixariam de existir em consequência do retorno na linha do tempo a esse estado desejável. Logo, para fins de cópias de segurança de maior periodicidade (acima de um mês) é recomendado o uso paralelo de métodos mais eficientes como o *backup* físico tradicional (cópia do diretório de dados) ou, principalmente, o *backup* lógico (usando o utilitário “*pg_dump*”). Portanto, o *backup* incremental deve ser utilizado em paralelo a outro método de cópia de segurança e não em substituição a este, já que seu propósito principal aqui é possibilitar a recuperação cronológica de dados, preferencialmente em um espaço de tempo recente.

Recuperação cronológica de dados

As rotinas de *backup* são indispensáveis quando o assunto é banco de dados, e devem garantir a recuperação dos dados caso seja necessário. Entretanto, essa operação tem custo operacional e daí a necessidade de terem horários programados para sua execução, geralmente uma vez ao dia. Logo, os métodos tradicionais de *backup* (físico ou lógico) são extremamente eficazes, mas estão sempre com defasagem de atualização, já que são gerados de acordo com o pré-agendamento e não em tempo real.

Fazendo uma analogia, se um determinado banco de dados tenha sido corrompido, ou tenha uma tabela excluída acidentalmente (por exemplo, no período da tarde) e seja necessário a sua recuperação, isso poderá ser possível recuperando a partir da cópia de segurança. Entretanto, caso a cópia de segurança tenha sido gerada durante a madrugada, terá, então, um grande dilema: o banco e a tabela serão recuperados, contudo, os dados inseridos, atualizados ou excluídos nesse período (intervalo entre

a criação do *backup* e o instante em que houve o problema no banco de dados) não poderão ser recuperados.

Essa situação pode ser solucionada com uso do Mecanismo PITR, esse recurso permite que se recupere todos os dados (total) em um determinado intervalo ou somente até determinado instante (parcial). Isso é possível porque todas as transações SQL de DDL[4], DML[5] e DCL[6] são, antes de efetivadas em disco, gravadas nos arquivos de *log* WAL, podendo essas serem recuperadas a partir do incremento em um *backup*-base gerado previamente. Assim, com o *backup*-base criado e o Mecanismo PITR habilitado, o usuário passa a ter em mãos uma poderosa ferramenta de recuperação de dados total ou parcial, conforme desejado. Desta forma, o usuário poderá restaurar ou recuperar sua base de dados a qualquer instante sempre que houver uma situação que implique na perda dos dados.

Por padrão, quando ativado, o mecanismo PITR fará a restauração completa do banco de dados, ou seja, efetuará o incremento de todos os arquivos de *log* WAL armazenados junto ao *backup*-base, restaurando assim, a base de dados ao instante que precedeu essa operação. Essa opção é recomendada para casos em que o banco de dados foi excluído acidentalmente ou tenha corrompido seus arquivos. Caso o problema seja de inclusão, alteração ou exclusão não desejada de objetos do banco de dados (por exemplo, tabelas, colunas, registros, chaves, restrições, entre outros), será necessário informar o "id" da transação ou a data e horário do instante em que aconteceu a operação não desejada, para que o PITR recupere os dados até aquele determinado instante. No item a seguir serão apresentados procedimentos para realizar essa operação.

É importante ressaltar que o Mecanismo PITR fará a restauração de todo o *Cluster*[7] de Banco de Dados, e não apenas de um banco de dados específico. Logo, ambientes que operam com vários bancos de dados no mesmo *Cluster* devem ter cuidado ao realizarem uma restauração parcial, ou seja, uma recuperação cronológica de dados até um determinado instante, pois

---

[4] DDL - *Data Definition Language* (ou Linguagem de definição de dados)

[5] DML - *Data Manipulation Language* (ou Linguagem de manipulação de dados)

[6] DCL - *Data Control Language* (ou Linguagem de Controle de Dados)

[7] O termo *Cluster* empregado aqui, tem o conceito de "Agrupamento de Banco de Dados". Cada *Cluster* de banco de dados no PostgreSQL possui seu próprio diretório de dados e sua porta de acesso.

esse retorno na linha do tempo valerá para todo o cluster, ou seja, para todos os bancos de dados e não somente para aquele a qual tenha ocorrido a falha ou a operação indesejável. Nesse caso, lembre-se de antes realizar um *backup* lógico dos demais bancos de dados, os quais não deseja retroceder, para a posterior restauração individual desses.

Restauração do diretório de dados

Para iniciar os procedimentos de restauração propriamente dito, é preciso parar o serviço do PostgreSQL, conforme já apresentado na Figura 6.

Por uma questão de segurança, no caso de haver arquivos de segmento WAL não copiados, mantenha o diretório de dados atual guardado, renomeando-o conforme visualizado na Figura 13, ou caso deseje, o mesmo poderá ser movido para outro local desejado. Lembrando que, em instalações padrões no Linux Ubuntu, o diretório de dados do PostgreSQL 8.x tem a denominação de "*main*" e encontra-se no caminho "*/var/lib/postgresql/8.x/*", assim, o comando para esse ambiente seria: "*mv /var/lib/postgresql/8.x/main /var/lib/postgresql/8.x/main.old*".

```
$mv /home/postgres/8.3/pgdata /home/postgres/8.3/pgdata.old
```

Figura 13. Renomear o diretório de dados.

O passo a seguir é copiar o *backup*-base "*pgdata.tar.gz*" para o local onde é alocado o diretório de dados do PostgreSQL (Figura 14).

```
$cp /home/postgres/8.3/backup/pgdata.tar.gz /home/postgres/8.3/
```

Figura 14. Copiar *backup*-base.

Em seguida, acesse o diretório onde foi realizado a cópia do arquivo, conforme a Figura 10, e descompacte o arquivo "*pgdata.tar.gz*" nesse mesmo local, assim como apresentado na Figura 15.

Figura 15. Descompactar arquivo "*pgdata.tar.gz*".

Visando evitar falhas durante o processo de recuperação de dados, acesse o novo diretório de dados (gerado a partir da descompactação do "*pgdata.tar.gz*") e exclua o arquivo identificador do processo "*postmaster*" do PostgreSQL, conforme Figura 16. Ainda, dentro do diretório de dados, acesse o subdiretório "*pg_xlog*" e remova todo o seu conteúdo (arquivos de segmentos WAL antigos), conforme Figura 17, a exceção do subdiretório "*archive_status*" que deverá ser mantido.

Figura 16. Remover arquivo do processo "*postmaster.pid*".

Figura 17. Remover arquivos de segmento WAL do diretório "*pgdata*".

Para garantir que as últimas transações realizadas no banco de dados, antes de sua parada ou do problema ocorrido, também estejam disponíveis, caso não tenham sido copiadas pelo Mecanismo PITR, deve se realizar a cópia dos arquivos de segmento WAL do diretório de dados antigo "*pgdata.old*" para o diretório atual "*pgdata*", conforme visualizado na Figura 18.

Figura 18. Copiar arquivos de segmento WAL do diretório "*pgdata.old*".

Configuração do arquivo "*recovery.conf*"

A última etapa para o processo de *Point-in-Time Recovery* é a configuração do arquivo "*recovery.conf*", que possui as definições necessárias para a operação de recuperação dos dados. Na biblioteca de consulta do PostgreSQL, estão disponíveis vários arquivos *template* (modelo) do servidor, entre eles o "*recovery.conf.sample*". Logo, acesse esse diretório (em instalações padrões, o mesmo encontra-se em "*/usr/share/postgresql/8.x/*") e efetue uma cópia desse arquivo para o diretório de dados atual ("*pgdata*" ou "*main*", de acordo com o seu ambiente operacional), conforme apresentado na Figura 19.

```
$cp /home/postgres/8.3/share/recovery.conf.sample /home/postgres/8.3/pgdata/
```

Figura 19. Copiar arquivo *template* "recovery.conf.sample"

Em seguida, acesse o diretório de dados "*pgdata*" (Figura 20) e renomeie o arquivo "*recovery.conf.sample*" para "*recovery.conf*" (Figura 21). Posteriormente, abra-o para edição, conforme Figura 22.

```
$cd /home/postgres/8.3/pgdata/
```

Figura 20. Acesso ao diretório de dados.

```
$mv recovery.conf.sample recovery.conf
```

Figura 21. Renomear arquivo "*recovery.conf.sample*" .

```
$vim recovery.conf
```

Figura 22. Abrir arquivo "*recovery.conf*".

Dentro do arquivo "*recovery.conf*", localize a sessão "REQUIRED PARAMETERS" e nessa, procure pelo item "*restore_command*", que se trata do parâmetro obrigatório para funcionamento do mecanismo PITR. Por padrão, o mesmo virá em formato de comentário e com valor padrão do servidor, conforme apresentado na Figura 23.

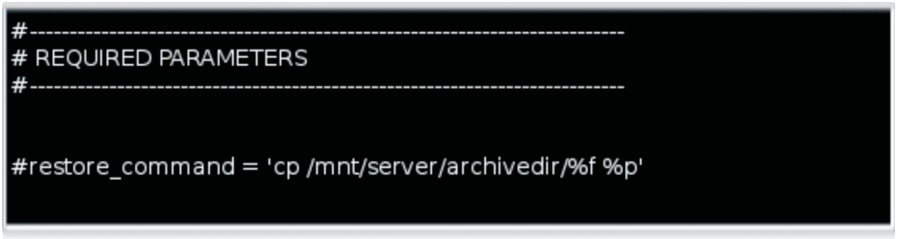


Figura 23. Parâmetro obrigatório do arquivo "*recovery.conf*" (padrão).

Edite o parâmetro, retirando-o do formato de comentário (excluindo o caractere "#") e inserindo no comando, o caminho para o *backup* dos arquivos de *log* WAL, conforme mostrado na Figura 24.

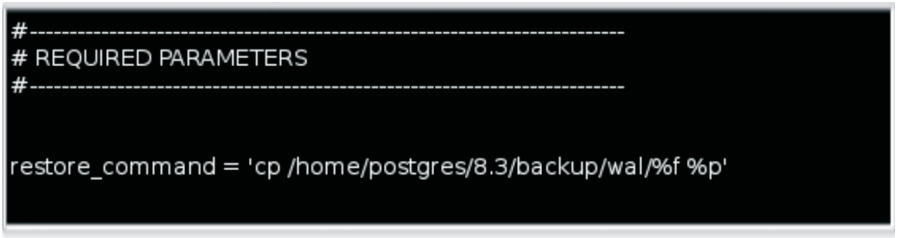


Figura 24. Parâmetro obrigatório do arquivo "*recovery.conf*" (editado)

Esse procedimento é o suficiente para, assim que o serviço do PostgreSQL for novamente iniciado, garantir a realização da operação de restauração do *cluster* de banco de dados. Contudo, vale ressaltar que o mecanismo PITR fará a restauração completa do *cluster* de banco de dados, ou seja, incrementará todos os arquivos de *log* WAL das transações ocorridas até o momento em que o servidor teve seu processo parado. Portanto, se deseja recuperar todo o banco de dados, salve e feche o arquivo "*recovery.conf*", e inicialize o serviço do PostgreSQL, conforme Figura 7.

Caso deseje realizar uma recuperação cronológica de dados até determinado instante, é necessário, ainda dentro do arquivo "*recovery.conf*", localizar a sessão "OPTIONAL PARAMETERS" e alterar, no mínimo, um dos parâmetros apresentados na Figura 25.

```
#---------------------------------------------------------------------------
# OPTIONAL PARAMETERS
#---------------------------------------------------------------------------

#recovery_target_time = '2004-07-14 22:39:00 EST'
#recovery_target_xid = '1100842'
#recovery_target_inclusive = 'true'          # 'true' or 'false'
```

Figura 25. Parâmetros opcionais do arquivo "*recovery.conf*" (padrão).

Os parâmetros "*recovery_target_time*" e "*recovery_target_xid*" são mutuamente exclusivos, ou seja, ocorrência de um elimina a possibilidade de ocorrência do outro, logo, um e somente um desses parâmetros deverá ser alterado e habilitado. O primeiro é o mais utilizado, o valor desse parâmetro corresponde à data e ao horário do instante para o qual se deseja retornar o estado do *cluster* de banco de dados. Enquanto que o segundo parâmetro corresponde ao identificador da transação para a qual se deseja retroceder o estado do *cluster*. Por fim, há ainda o parâmetro "*recovery_target_inclusive*" que é utilizado juntamente com a escolha de um dos parâmetros anteriores, e seu respectivo valor será "*true*" (valor padrão) ou "*false*". Essa opção define que se o valor for igual a "*true*", a recuperação cronológica dos dados ocorrerá até a data e horário (*time*) ou transação (*xid*) informada, ou seja, inclusive, senão (valor igual a "*false*") a recuperação cronológica dos dados ocorrerá até o instante (*time*) ou transação (*xid*) imediatamente anterior à informada.

A Figura 26 apresenta um exemplo de como ficariam os parâmetros para uma recuperação cronológica dos dados, onde são informados a data e o horário em "*recovery_target_time*" e o mesmo é retirado do formato de comentário (exclusão do caractere "#") e a opção "*recovery_target_inclusive*" não é alterada, ou seja, manteve-se o valor padrão. Com isso, salve e feche o arquivo "*recovery.conf*", inicialize o serviço do PostgreSQL (Figura 7). A recuperação do *cluster* ocorrerá de maneira automática até o estado do instante (*time*) fornecido.

Ao final da restauração do *cluster*, seja ela total ou parcial, o servidor PostgreSQL automaticamente renomeará o arquivo "*recovery.conf*" para "*recovery.done*", evitando-se com isso que em uma nova reinicialização seja realizada novamente a recuperação dos dados. Com isso, sua base de dados já estará pronta e disponível, e, em caso de nova restauração, é só repetir os procedimentos conforme sua necessidade.

```
#---------------------------------------------------------------------------
# OPTIONAL PARAMETERS
#---------------------------------------------------------------------------

recovery_target_time = '2009-08-21 10:45:00 EST'
#recovery_target_xid = '1100842'
#recovery_target_inclusive = 'true'          # 'true' or 'false'
```

Figura 26. Parâmetros opcionais do arquivo "*recovery.conf*" (editado).

Literatura recomendada

POSTGRESQL. Disponível em: <http://www.postgresql.org/> Acesso em: 04 ago. 2009.

POSTGRESQLBRASIL. Disponível em: <http://www.postgresql.org.br/> Acesso em: 12ago. 2009.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **Documentação do PostgreSQL 8.0.0**. Tradução: Haley Pacheco de Oliveira. Rio de Janeiro, 2005. 1205 p.

Embrapa

Informática Agropecuária

Ministério da
Agricultura, Pecuária
e Abastecimento

BRASIL
UM PAÍS DE TODOS
GOVERNO FEDERAL