

14) Guia de Replicação no PostgreSQL com Slony-I (Usando o PGAdmin, PostgreSQL-8.2.6)

14.1) Conceitos

14.2) No Windows XP

14.3) No Linux Ubuntu 7.10

14.1) Conceitos de replicação

Replicação de banco de dados é a **cópia dos dados de uma base de dados original para outra base**. Isto permite que possa **disponibilizar os dados em um ou mais sites**. Isto melhora a disponibilidade dos dados. A cópia destes dados podem ser parcial ou total.

A replicação poderá ser usada para Sistemas distribuídos, Sistema de alta disponibilidade, ou talvez em processo de ETL em Data Warehouse (*).

Nos conceitos atuais de replicação a primeira classificação das ferramentas é relativo a sua arquitetura é **síncrona ou assíncrona**.

Replicação **síncrona** é uma **cópia fiel em qualquer momento dos dados e transações**. No mecanismo **assíncrono**, o **servidor Master assume toda carga de escrita, atualização, deleção e leitura, com a diferença que o Slave apenas disponibiliza leitura da base** conforme a figura 2.

Nos servidores Master podem ocorrer transações de leitura, escrita, atualização e deleção, já no servidor Slave, poderá ocorrer apenas operações de leitura, conforme figura 1.

Entre servidores Multimaster você poderá ter síncrono ou assíncrono, já em servidores Master e Slave, assíncrono. Isto é determinado pela ferramenta utilizada, e o que determina qual arquitetura será utilizada é o objetivo de implementação. Segue abaixo um comparativo de ferramentas para replicação no PostgreSQL:

	Slony-I	Mamooth	Postgres-R	PG-Replicator
Tipo:	Assíncrono	Assíncrono	Síncrono	Assíncrono
Uso:	Master - multislave	Master - multislave	MultiMaster	Multimaster
Propaga DDL	Não	Não	Não	?
Sist. Operacional	Todos	Todos	Todos	Linux
BLOB	Não	?	?	Sim

Nestas três arquiteturas propostas acima, cada uma tem uma aplicabilidade específica preferencial.

- Síncrono Multimaster: Balanceamento de carga ou alta disponibilidade;
- Assíncrono Multimaster: Alta disponibilidade;

- Assíncrono Master-Slave: Relatórios diversos níveis.

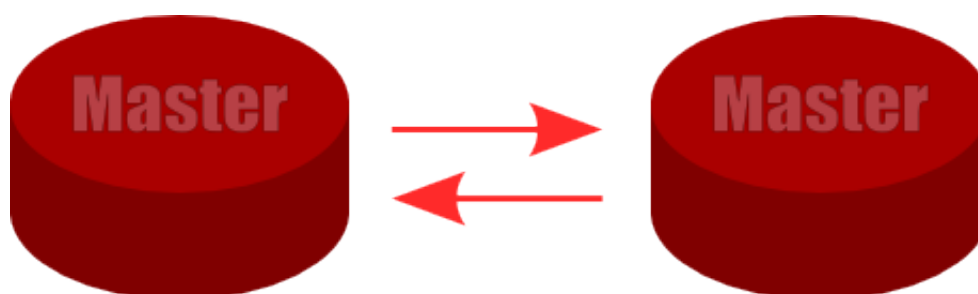


Fig1 – Servidores Master estão ambos sincronizados transmitindo escrita, atualizações e deleções em tempo real.

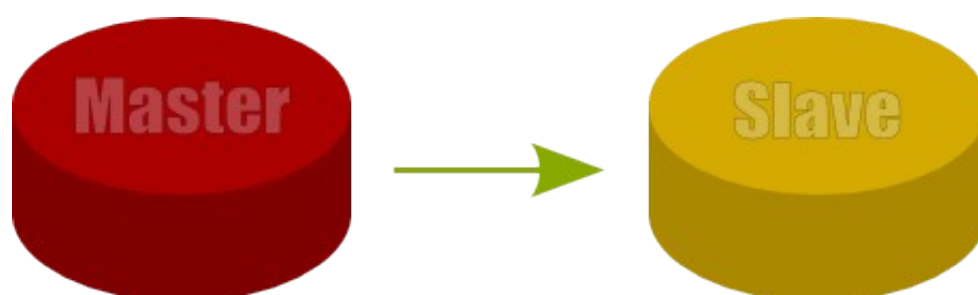


Fig2 - Servidores Master e Slave estão ambos sincronizados transmitindo escrita, atualizações e deleções em tempo real do Master para o Slave.

Para quem ainda não está familiarizado com as particularidades do mecanismo síncrono ou assíncrono talvez se pergunte o que aconteceria se um “Servidor A” replicasse assincronamente ao “Servidor B” e vice versa, isto não seria parecido com o mecanismo síncrono? A resposta disso é imediata, o servidor Slave é atualizado apenas e exclusivamente pelo mecanismo de sincronização. Isto provocaria um “lock”, travamento, de entrada de dados em ambos os servidores, ficando ambos apenas disponível para leitura, o que é logicamente um erro. A maior aplicabilidade dos mecanismos síncronos são para proporcionar alta disponibilidade, enquanto o mecanismo assíncrono pode ser usado para melhoria de performance e disponibilidade de relatórios complexos, e por que não dizer parte constituinte de um *Data Warehouse*, no processo de Extração, Transformação e Carga(ETL). **A alta disponibilidade é desejável em sistemas críticos que não podem parar.** Em caso de crash de um servidor, outro servidor assume de forma transparente para o usuário final do sistema. Para nosso primeiro exemplo de replicação imaginem uma empresa matriz no Distrito Federal e com filiais nos estados do Rio de Janeiro e São Paulo. A estrutura de rede necessariamente uma máquina deve “enxergar” a outra, seja por IP fixo, VPN ou outra. A matriz possui a gerência de material e recursos humanos da empresa toda e as filiais estão on-line podendo acompanhar atualizações na gestão de material e dos recursos humanos. Se houver uma queda temporária da rede a(s) filial(ais) estarão com o último estado consistente da base de dados, podendo manter seus serviços com menor risco, como por exemplo de vender algo que já foi vendido pela matriz ou outra filial, ou de não vender por não saber que o estoque já foi renovado e está disponível.

(*) Um *data warehouse* (ou **armazém de dados**, ou **depósito de dados** no Brasil) é um [sistema](#) de

computação utilizado para armazenar informações relativas às atividades de uma organização em bancos de dados, de forma consolidada. O desenho da base de dados favorece os relatórios, a análise de grandes volumes de dados e a obtenção de informações estratégicas que podem facilitar a tomada de decisão.

O *data warehouse* possibilita a análise de grandes volumes de dados, coletados dos sistemas transacionais (OLTP). São as chamadas séries históricas que possibilitam uma melhor análise de eventos passados, oferecendo suporte às tomadas de decisões presentes e a previsão de eventos futuros. Por definição, os dados em um *data warehouse* não são voláteis, ou seja, eles não mudam, salvo quando é necessário fazer correções de dados previamente carregados. Os dados estão disponíveis somente para leitura e não podem ser alterados.

A ferramenta mais popular para exploração de um *data warehouse* é a *Online Analytical Processing* OLAP ou Processo Analítico em Tempo Real, mas muitas outras podem ser usadas.

Fonte: http://pt.wikipedia.org/wiki/Data_warehouse

Dentre os mecanismos de replicação iremos no capítulo seguinte mostrar o Slony, atualmente é o mais conhecido pela internet e aparentemente é o projeto que ganha mais adeptos.

Algumas características do Slony são:

- Replicação Master Multislave;
- Assíncrono;
- Multiplataforma;
- A definição da tabela a ser replicada na base “Master” deve ser idêntica a “Slave”;
- Replicação de tabelas e seqüências. Os demais objetos devem ser importados na base Slave de acordo com a necessidade do usuário;
- Em caso de falha o servidor slave não para, e quando retomada a comunicação, o banco slave retorna automaticamente a forma do master em “pouco tempo”;
- Slony deve ser instalado em cada servidor que desejar ser master ou slave;
- Baixo tráfego pela rede;
- As tabelas e seqüências slave ficam permanentemente com lock para escrita, atualização ou deleção. Qualquer tratamento deve ser feito pelo banco através de functions ou triggers;
- Inadequado para clusters multimaster;

Conceitos no Slony

O Slony é um replicador Master multi-Slave, assíncrono, e é multi-plataforma, ou seja, podemos instalá-lo no Windows, Linux, UNIX, Free BSD, e outros.

Fonte: <http://www.insphired.com/content/view/57/1/>

Alerta: muito importante verificar se a versão do PostgreSQL é compatível com a do Slony. Veja neste site detalhes: <http://developer.pgadmin.org/~hiroshi/Slony-I/>

O PGAdmin tem capacidade de criar e administrar replicação de clusters Slony-I já há algum tempo, porém ele foi projetado para permitir que o usuário trabalhe diretamente com os níveis mais baixos de conceitos do Slony, como listens (escutas) e paths (caminhos). Isto é muito flexível mas não é muito amigável. Nós estamos esperando incluir alguns assistentes no futuro para tornar mais fácil de configurar e modificar clusters, mas enquanto isso, aqui está um bom atalho para criar um cluster de replicação com 3 nós.

14.2) No Windows XP

Neste exemplo um servidor master é configurado com dois slaves diretos. Este exemplo foi escrito e testado originalmente usando Slony-I v1.2.11 e PostgreSQL-8.2.5, rodando numa única máquina com o Windows XP. Repetido em Windows XP, PostgreSQL-8.2.6 e Slony-I 1.2.12.

Obs.: os três servidores, do master, do slave1 e do slave2, neste exemplo, estão numa mesma máquina (localhost). Numa situação real, geralmente cada um tem um IP diferente.

O utilitário (contrib) pgbench é utilizado para gerar o schema de teste e a carga de trabalho.

Replicando um banco de dados para dois slaves

1 – Crie três bancos: master, slave1 e slave2 e instale plpgsql em todos.

2 – Criar um schema do pgbench no banco master:

```
C:\Program Files\PostgreSQL\8.2\bin>pgbench -i -U postgres master -P postgres -p 5432
```

3 – Adicione na tabela history uma chave primária com nome history_pkey nos campos tid, bid e aid.

4 – Gerar um dump, somente do schema do banco master e importar nos bancos slave1 e slave2.

```
pg_dump -s -p 5432 -U postgres master > schema.sql
psql -p 5432 -U postgres slave1 < schema.sql
psql -p 5432 -U postgres slave2 < schema.sql
```

5 – Criar um script de configuração para cada engine do Slony. Os arquivos deverão conter somente duas linhas como as seguintes:

```
c:\slony\master.conf
```

```
cluster_name='pgbench'
conn_info='host=127.0.0.1 port=5432 user=postgres dbname=master
password=postgres'
```

Crie um arquivo para cada banco, ajuste o dbname e faça outros ajustes se precisar.

6 – Instalar o serviço do Slony

```
slon -regservice Slony-I
```

7 – Registrar cada um dos engines

Uso do comando slon:

```
slon [options] clustername conninfo
```

```
slon -addengine Slony-I C:\slony\master.conf
```

```
slon -addengine Slony-I C:\slony\slave1.conf
```

```
slon -addengine Slony-I C:\slony\slave2.conf
```

8 – No PGAdmin

Antes indicar em Arquivo – Opções

Em caminho do slony indique - [C:\Arquivos](#) de Programas\PostgreSQL\8.2\share

Sob o Nó Replicação do banco master criar um novo cluster Slony-I usando as seguintes opções:

```
Join existing cluster: Unchecked
```

```
Cluster name: pgbench
```

```
Local node: 1 Master node
```

```
Admin node: 99 Admin node
```

9 – Em cada um dos bancos slave1 e slave2 criar um cluster de Replicação com as opções:

slave1:

```
Join existing cluster: Checked
```

```
Server: <Select the server containing the master database>
```

```
Database: master
```

```
Cluster name: pgbench
```

```
Local node: 10 Slave node 1
```

```
Admin node: 99 - Admin node
```

e slave2:

```
Join existing cluster: Checked
```

```
Server: <Select the server containing the master database>
```

```
Database: master
```

```
Cluster name: pgbench
```

```
Local node: 20 Slave node 2
```

Admin node: 99 - Admin node

10 – Criar no master, paths para ambos os slaves e em cada slave voltar para o master. Criar os paths sob cada nó no master usando a string de conexão do script de configuração. Observe que futuras reconstruções do cluster devem exigir que novos paths sejam definidos. Todos os paths devem ser preenchidos, tanto os do master quanto os dos slaves. Usar a string de conexão dos arquivos de configuração adequadamente:

```
host=127.0.0.1 port=5432 user=postgres dbname=master password=postgres
host=127.0.0.1 port=5432 user=postgres dbname=slave1 password=postgres
host=127.0.0.1 port=5432 user=postgres dbname=slave2 password=postgres
```

Veja na próxima página os detalhes

Exemplo de preenchimento dos Paths

Tomemos como exemplo os nós do banco master:

Ao expandir Replication – pgbench - Nodes.

Então vemos: Master node, Slave node 1, Slave node 2 e Admin node. Ao expandir cada um destes, encontraremos Paths(0), que deverá ser preenchido com todas as possibilidades, como por exemplo, o do Master node: clique sobre Paths(0) com o botão direito e New Path, então aparece um diálogo para entrarmos com informações do "10 – Slave node 1", apenas com Connect info vazio, onde devemos entrar com as informações de conexão do Slave1:

```
host=127.0.0.1 port=5432 user=postgres dbname=slave1 password=postgres
```

E Ok.

Então clicamos novamente com o botão direito do mouse sobre Paths(1) e New Path. Agora ele nos solicita as informações para o Slave2 e entramos em Connect info com:

```
host=127.0.0.1 port=5432 user=postgres dbname=slave2 password=postgres e Ok.
```

Clique novamente com o botão direito sobre Paths(2) e New Path. Observe que agora não aparece nenhum servidor. Mesmo que procuremos na combo, nada aí. Ou seja, ele nos solicita somente os que precisa, o que ajuda e muito a quem está iniciando.

De forma semelhante preencha todos os paths do master e dos slaves, de forma que ao final fique como o que aparece na próxima página.

Banco Master

Replication

pgbench

Nodes (4)

Master node

Paths(2)

```

Slave node 1
Slave node 2
listens(0)
Slave node 1
Paths(2)
Master node
Slave node 2
listens(0)
Slave node 2
Paths(2)
Master node
Slave node 1
listens(0)
Admin node
Paths(3)
Master node
Slave node 1
Slave node 2
listens(9)
Replication Sets(1)

```

Banco Slave1

```

Replication
pgbench
Nodes (3)
Master node
Paths(1)
Slave node 1
listens(1)
Slave node 1
Paths(1)
Master node
listens(0)
Admin node
Paths(2)
Master node
Slave node 1
listens(2)
Replication Sets(0)

```

Banco Slave2

```

Replication
pgbench
Nodes (4)
Master node
Paths(2)
Slave node 1
Slave node 2

```

```
listens(0)
Slave node 1
  Paths(2)
    Master node
    Slave node 2
  listens(0)
Slave node 2
  Paths(2)
    Master node
    Slave node 1
  listens(4)
Admin node
  Paths(3)
    Master node
    Slave node 1
    Slave node 2
  listens(6)
Replication Sets(0)
```


11 – Criar um Replication Set no master usando as seguintes configurações:

ID: 1

Comment: pgbench set

12 – Adicionar as tabelas para o Replication set (master) com as configurações:

(master – Replication – pebench – Replication Sets – pgbench set - Tables)

Table: public.accounts

ID: 1

Index: accounts_pkey

Table: public.branches

ID: 2

Index: branches_pkey

Table: public.history

ID: 3

Index: history_pkey

Table: public.tellers

ID: 4

Index: tellers_pkey

13 – No master node criar uma nova subscrição para cada slave usando as seguintes opções:

(pgbench set – botão direito – new object – new subscription)

Origin: 1

Provider: 1 – Master node

Receiver: 10 – Slave node 1

Origin: 1

Provider: 1 – Master node

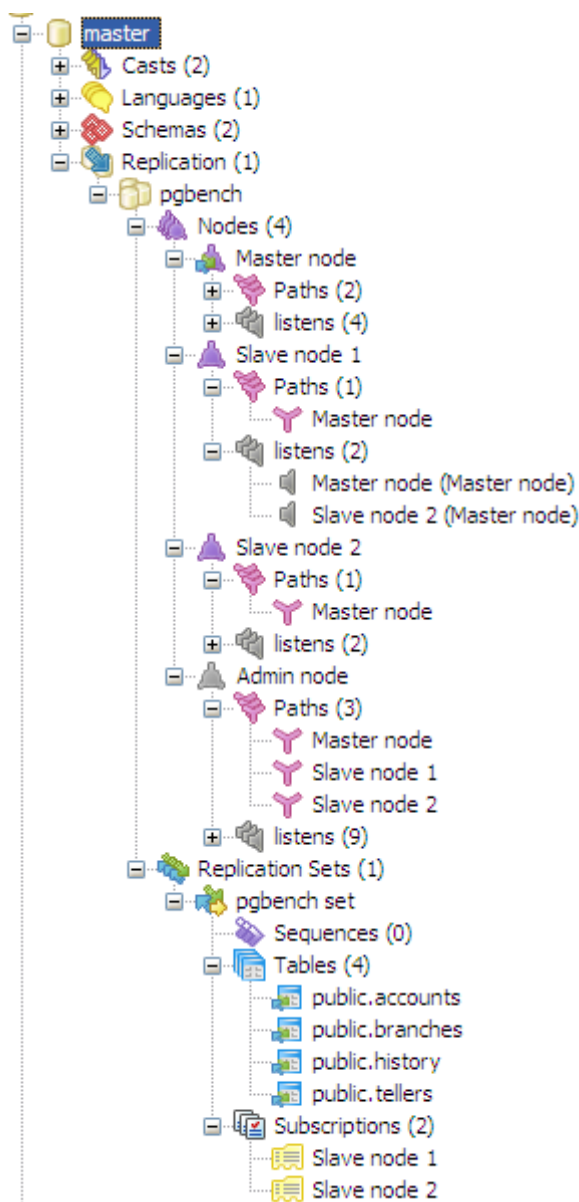
Receiver: 20 – Slave node 2

14 - Iniciar o serviço slon:

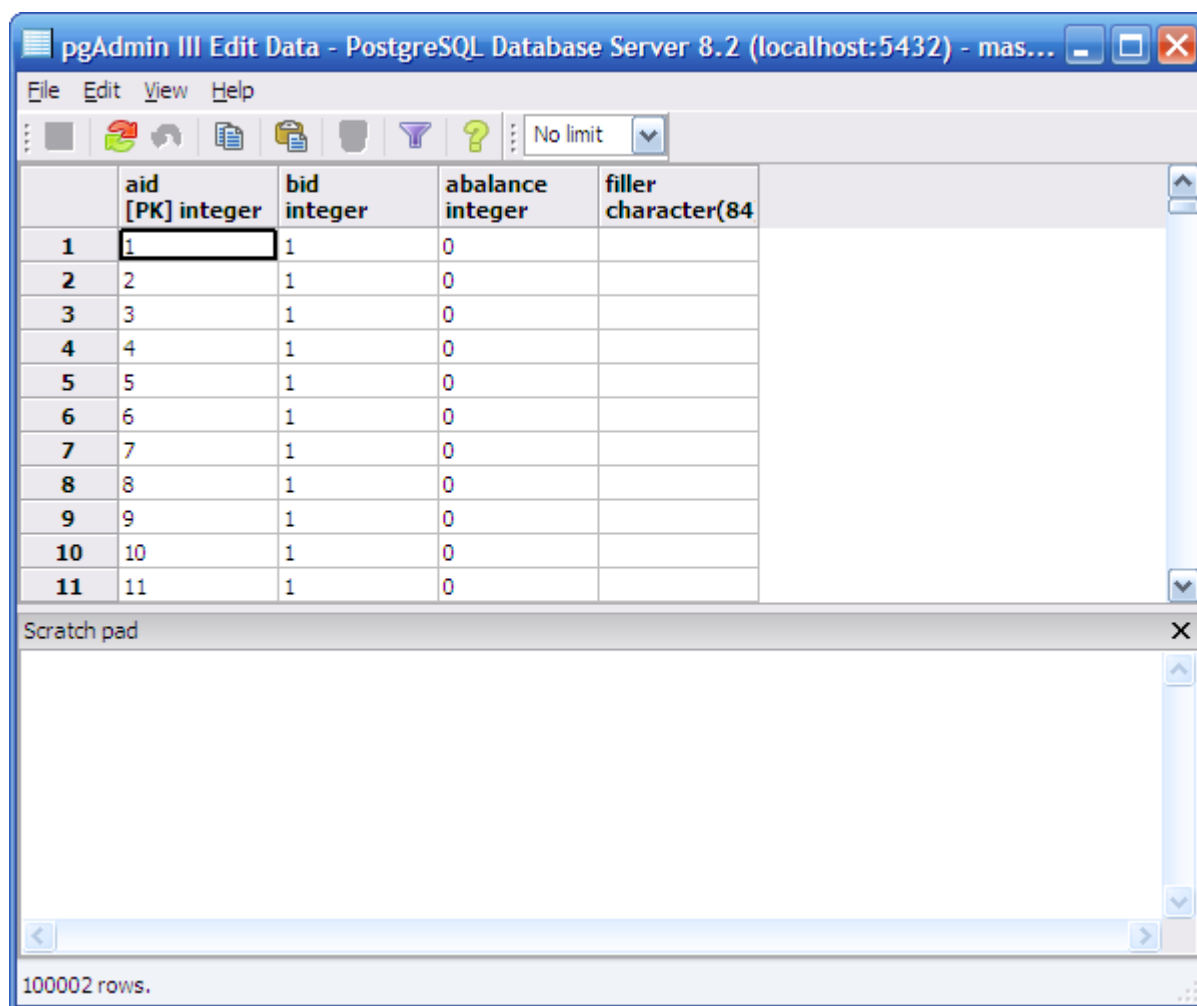
```
net start Slony-I
```

A replicação inicial deve começar e pode ser monitorada na aba Estatística do PGAdmin para cada nó. Basta selecionar o cluster pgbench, expandir os 4 nós e selecionar Estatística.

Veja o resultado no banco master após a conclusão:



Veja agora a tabela account no master:



pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - mas...

File Edit View Help

No limit

	aid [PK] integer	bid integer	abalance integer	filler character(84)
1	1	1	0	
2	2	1	0	
3	3	1	0	
4	4	1	0	
5	5	1	0	
6	6	1	0	
7	7	1	0	
8	8	1	0	
9	9	1	0	
10	10	1	0	
11	11	1	0	

Scratch pad

100002 rows.

Agora no Slave 1:

The screenshot shows the pgAdmin III 'Edit Data' window for a PostgreSQL database. The window title is 'pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - slave1 ...'. The interface includes a menu bar (File, Edit, View, Help), a toolbar with various icons, and a data table. The table has five columns: 'aid [PK] integer', 'bid integer', 'abalance integer', and 'filler character(84)'. The first column is highlighted. The table contains 11 rows of data, with the first row having 'aid' 1, 'bid' 1, 'abalance' 0, and 'filler' empty. Below the table is a 'Scratch pad' area. At the bottom, a status bar indicates '100002 rows.'.

	aid [PK] integer	bid integer	abalance integer	filler character(84)
1	1	1	0	
2	2	1	0	
3	3	1	0	
4	4	1	0	
5	5	1	0	
6	6	1	0	
7	7	1	0	
8	8	1	0	
9	9	1	0	
10	10	1	0	
11	11	1	0	

Scratch pad

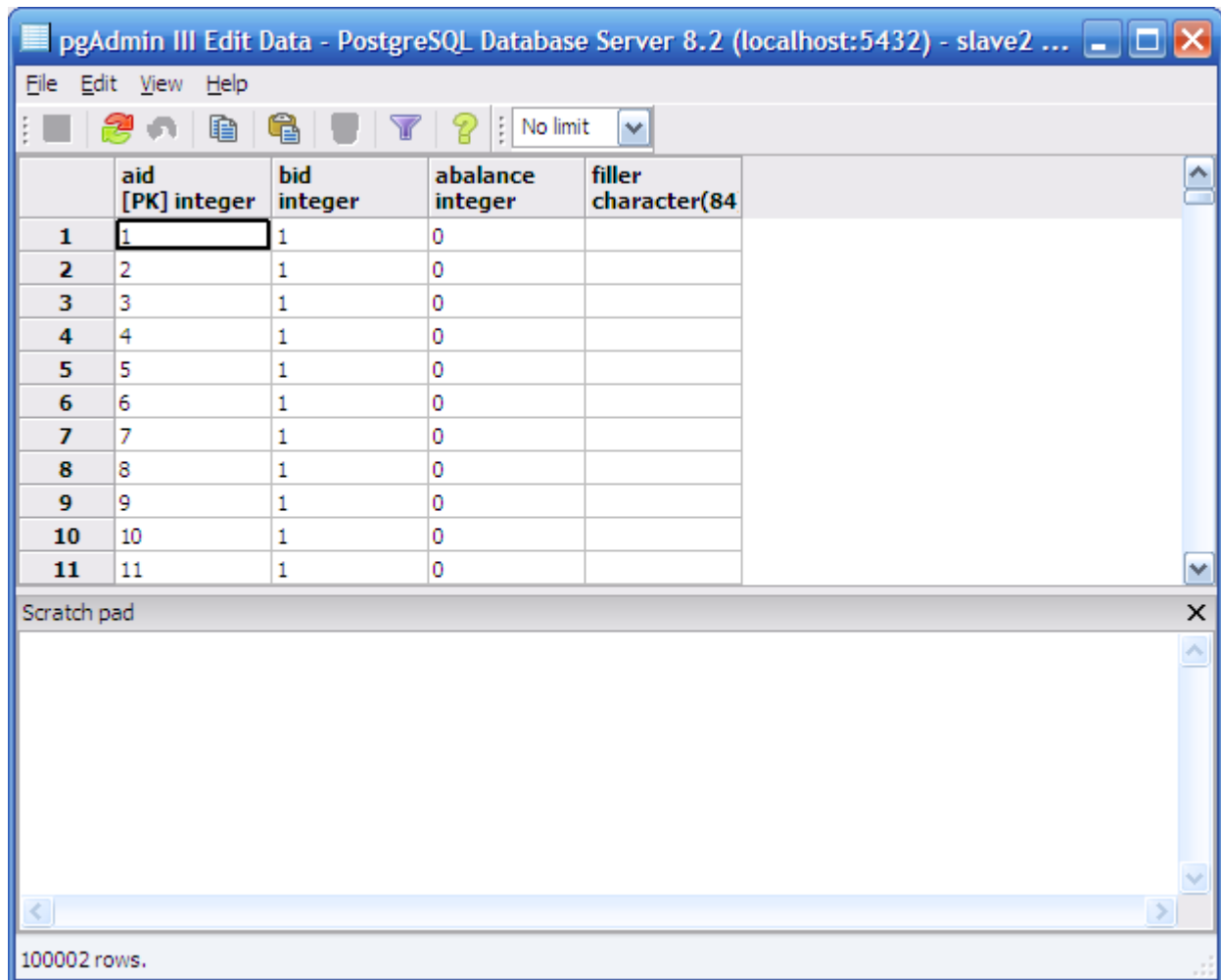
100002 rows.

Após inserir um registro na tabela account do banco master e dar um refresh, este mesmo registro foi visto nos slaves.

Lembrando que somente as tabelas do master permitem escrita. As dos slaves são somente leitura.

Orinalmente eram 100.000 registros. Inseri mais dois e os mesmos foram também vistos nos slaves.

Veja o slave2



pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - slave2 ...

File Edit View Help

No limit

	aid [PK] integer	bid integer	abalance integer	filler character(84)
1	1	1	0	
2	2	1	0	
3	3	1	0	
4	4	1	0	
5	5	1	0	
6	6	1	0	
7	7	1	0	
8	8	1	0	
9	9	1	0	
10	10	1	0	
11	11	1	0	

Scratch pad

100002 rows.

Original em inglês: <http://people.planetpostgresql.org/dpage/index.php?/archives/51-Setting-up-Slony-I-with-pgAdmin.html>

Tradução de Ribamar FS.

14.3) No Linux Ubuntu 7.10

Replicando um banco de dados para dois slaves

Instalar o PostgreSQL-8.2.7, PostgreSQL-8.2-slony1, PostgreSQL-contrib-8.2 (que traz o pgbench), slony1-bin e slony1-doc

1 – Crie três bancos: master, slave1 e slave2 e instale plpgsql em todos.

2 – Criar um schema do pgbench no banco master:

```
su - postgres
/usr/lib/postgresql/8.2/bin/pgbench -i -U postgres master -P postgres -p 5432
```

3 – Adicione na tabela history uma chave primária com nome history_pkey nos campos tid, bid e aid.

4 – Gerar um dump, somente do schema do banco master e importar nos bancos slave1 e slave2.

```
su - postgres

/usr/lib/postgresql/8.2/bin/pg_dump -s -U postgres master > schema.sql -p 5432
/usr/lib/postgresql/8.2/bin/psql -U postgres slave1 < schema.sql -p 5432
/usr/lib/postgresql/8.2/bin/psql -U postgres slave2 < schema.sql -p 5432
```

5 – Criar um script de configuração para cada engine do Slony (daemon em *nix). Os arquivos deverão conter somente duas linhas como as seguintes:

/var/lib/postgresql/slony/master.conf e slave1.conf e slave2.conf, com o conteúdo:

```
cluster_name='pgbench'
conn_info='host=127.0.0.1 port=5432 user=postgres dbname=master password=postgres'
```

Alerta: Cuidado para não inserir ; ao final das linhas. Isso aconteceu comigo e deu trabalho para descobrir. Quando editei o master.log que fica no /var/lib/postgresql/slony/ eu percebi pela mensagem de erro.

Crie um arquivo para cada banco, ajuste o dbname e faça outros ajustes se precisar.

6 – Registrar cada um dos engines

Uso do comando slon - slon [options] clustername conninfo

```
su - postgres
slon -f /var/lib/postgresql/slony/master.conf > master.log 2>& 1 &
```

```
slon -f /var/lib/postgresql/slony/slave1.conf > slave1.log 2>& 1 &  
slon -f /var/lib/postgresql/slony/slave2.conf > slave2.log 2>& 1 &
```

7 – No PGAdmin

Antes indicar em Arquivo – Opções - Caminho do slony - /usr/share/slony1

Sob o Nó Replicação do banco master criar um novo cluster Slony-I usando as seguintes opções:

```
Join existing cluster: Unchecked  
Cluster name: pgbench  
Local node: 1 Master node  
Admin node: 99 Admin node
```

8 – Em cada um dos bancos slave1 e slave2 criar um cluster de Replicação com as opções:

```
slave1:  
Join existing cluster: Checked  
Server: <Select the server containing the master database>  
Database: master  
Cluster name: pgbench  
Local node: 10 Slave node 1  
Admin node: 99 - Admin node
```

```
e slave2:  
Join existing cluster: Checked  
Server: <Select the server containing the master database>  
Database: master  
Cluster name: pgbench  
Local node: 20 Slave node 2  
Admin node: 99 - Admin node
```

9 – Criar no master, paths para ambos os slaves e em cada slave voltar para o master. Criar os paths sob cada nó no master usando a string de conexão do script de configuração. Observe que futuras reconstruções do cluster devem exigir que novos paths sejam definidos. Todos os paths devem ser preenchidos, tanto os do master quanto os dos slaves. Usar a string de conexão dos arquivos de configuração adequadamente:

```
host=127.0.0.1 port=5432 user=postgres dbname=master password=postgres  
host=127.0.0.1 port=5432 user=postgres dbname=slave1 password=postgres  
host=127.0.0.1 port=5432 user=postgres dbname=slave2 password=postgres
```

Observação: No banco master, no Nó master criar dois pats, um para cada slave.
Em cada Nó Slave, criar um path para o master.

Repetir o procedimento para os bancos slaves.

10 – Criar um Replication Set no master usando as seguintes configurações:

ID: 1
Comment: pgbench set

11 – Adicionar as tabelas para o Replication set (master) com as configurações:

(master – Replication – pebench – Replication Sets – pgbench set - Tables)

Table: public.accounts
ID: 1
Index: accounts_pkey

Table: public.branches
ID: 2
Index: branches_pkey

Table: public.history
ID: 3
Index: history_pkey

Table: public.tellers
ID: 4
Index: tellers_pkey

12 – No master node criar uma nova subscrição para cada slave usando as seguintes opções:

Origin: 1
Provider: 1 – Master node
Receiver: 10 – Slave node 1

Origin: 1
Provider: 1 – Master node
Receiver: 20 – Slave node 2

13 - Concluído

A replicação inicial deve começar e pode ser monitorada na aba Estatística do PGAdmin para cada nó. Basta selecionar o cluster pgbench, expandir os 4 nós e selecionar Estatística.

Fontes consultadas:

- Artigo do pontapé inicial - <http://people.planetpostgresql.org/dpage/index.php?/archives/51-Setting-up-Slony-I-with-pgAdmin.html>
- <http://www.insphired.com/content/view/57/1/>
- Lista internacional do PostgreSQL - <http://archives.postgresql.org/>
- Página oficial do Slony - <http://www.slony.info/>
- Documentação online do Slony - <http://linuxfinances.info/info/slonyintro.html>
- Documentação para download - <http://main.slony.info/downloads/1.2/source/slony1-1.2.13-docs.tar.bz2>
- Em pdf - <http://www.postgresql.org.br/Documenta%C3%A7%C3%A3o?action=AttachFile&do=get&target=slony.pdf>
- Bons artigos sobre Replicação com slony no Linux:
<http://www.linuxjournal.com/article/7834>
<http://www.vivaolinux.com.br/artigos/impressora.php?codigo=4536>
- Wikipédia – <http://pt.wikipedia.org>
- <http://www.google.com.br>
- Capítulo 24 do livro
The comprehensive guide to building, programming, and administering PostgreSQL databases,
De Korry Douglas e Susan Douglas, SAMS editora.