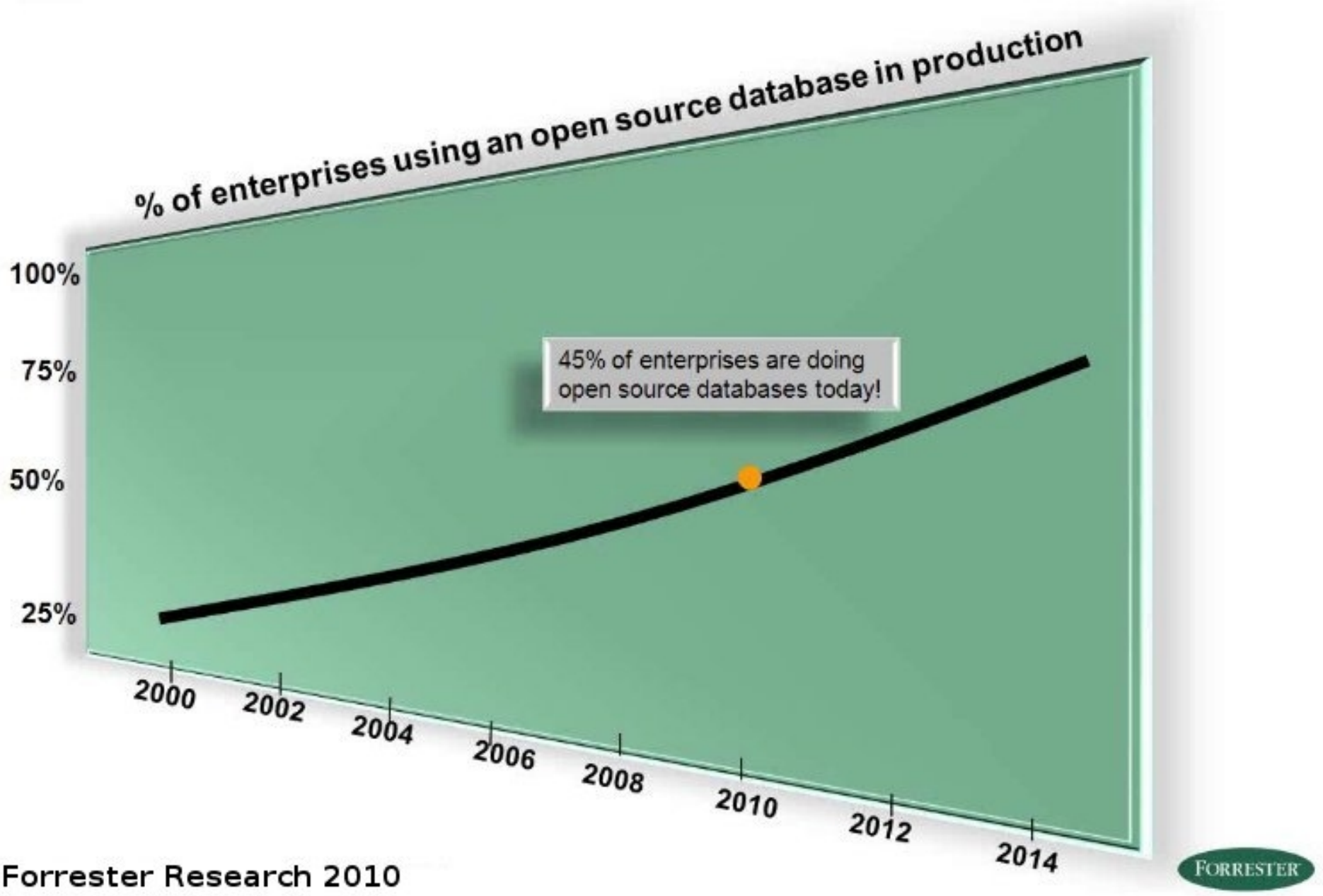
PostgreSQL: Novidades da Versão 9.0

PgDay SP - Outubro/2010

Mercado de BDs open source



Forrester Research 2010

Novidades do PostgreSQL 9.0

dextra
Coding your Business

Mais de 200 melhorias na versão 9.0

- Hot standby
- Streaming Replication
- Melhorias na administração de privilégios dos usuários
- Blocos anônimos em PL/pgSQL
- Triggers condicionais e por colunas
- Reimplementação do VACUUM FULL
- Contrib para migração de versões
- Melhorias de performance para consultas geradas por ORMs

Dupla Dinâmica

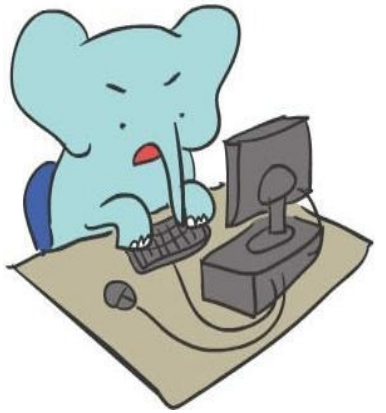
- **Hot standby & Streaming Replication**

Santa Disponibilidade, Batman!!!

Comparativo entre versões

- **Desde a versão 8.2: Warm Standby**
  - Alta disponibilidade com servidor standby
  - Replicação por arquivos do WAL
  - Não disponibiliza o servidor standby para consultas

- **PostgreSQL 9.0: Hot Standby e Streming Replication**
  - Alta disponibilidade com servidor standby
  - Replicação por fragmentos do WAL
  - Servidor standby disponível para consultas



Warm Standby 9.0

- **Não é necessário instalar a contrib pg_standby**
- **wal_level:** Novo parâmetro que define o nível de informação a ser gravada nos segmentos de log de transação:
  - **minimal:** Registra o mínimo de informação necessária para recuperação do servidor após crash.
  - **archive:** Registra informações para recuperação do backup via arquivamento do WAL ou streaming replication.
  - **hot_standby:** Registra informações sobre transações que estão ativas, possibilitando a restauração em modo read-only.
- **A configuração com pg_standby ainda é possível**

Warm Standby 9.0

- ***postgresql.conf* no servidor master:**

```
wal_level = archive
archive_mode = on
```

  - ***Inicia o processo archiver***

```
archive_command = 'scp %p dextra02:/archives/%f'
```

  - ***Copia o log de transação para os servidores de contingência***

```
archive_timeout = 120
```

  - ***Força a cópia do log de transação a cada 120 segundos***
  - ***O objetivo é diminuir o tempo máximo de atraso das réplicas em relação ao servidor de produção***

Warm Standby 9.0

dextra
Coding your Business

```
postgres@dextra02:~$ ps -ef | grep postgres
postgres  1444     1    07:50  00:00:00 /usr/local/bin/postgres
postgres  1446  1444 07:50  00:00:00 postgres: writer process
postgres  1447  1444 07:50  00:00:00 postgres: wal writer process
postgres  1448  1444 07:50  00:00:00 postgres: autovacuum launcher process
postgres  1449  1444 07:50  00:00:00 postgres: archiver process
postgres  1450  1444 07:50  00:00:00 postgres: stats collector process
```

Warm Standby 9.0

- ***recovery.conf* no servidor slave ($PGDATA):**

```
standby_mode = 'true'
```

  - ***Indica se o PostgreSQL permanecerá em modo de restauração, seja através do restore_command ou WalReceiver***

```
restore_command = 'cp /archives/%f %p'
```

  - ***Comando que copia os logs de transação arquivados para o diretório $PGDATA/pg_xlog***

```
trigger_file = '/tmp/arquivo_gatilho.pgsql'
```

  - ***Arquivo necessário para indicar o fim do processo de restauração contínua***

Warm Standby 9.0

postgres@dextra02:~$ pg_ctl start

LOG: entering standby mode

LOG: restored log file "000000010000000000000005" from archive

LOG: redo starts at 0/5000070

LOG: consistent recovery state reached at 0/6000000

LOG: restored log file "000000010000000000000006" from archive

LOG: restored log file "000000010000000000000007" from archive

Warm Standby 9.0

- **O servidor standby permanece em modo de restauração**
  - Verifica por logs de transações enviados pelo servidor de produção

cp: cannot stat `/archives/000000010000000000000008': No such file or directory

cp: cannot stat `/archives/000000010000000000000008': No such file or directory

LOG: restored log file "000000010000000000000008" from archive

cp: cannot stat `/archives/000000010000000000000009': No such file or directory

cp: cannot stat `/archives/000000010000000000000009': No such file or directory

Warm Standby 9.0

- **Não é possível acessar o servidor durante a restauração:**

  postgres@dextra02:~$ psql

  FATAL: the database system is starting up

- **A criação do arquivo de gatilho possibilita o failover:**

  postgres@dextra02:~$ touch /tmp/arquivo_gatilho.pgsql

  LOG: trigger file found: /tmp/arquivo_gatilho.pgsql

  LOG: restored log file "000000010000000000000008" from archive

  LOG: selected new timeline ID: 2

  LOG: archive recovery complete

  LOG: autovacuum launcher started

  LOG: database system is ready to accept connections

Hot Standby

- **Permite a criação de instâncias secundárias (standby), que podem ser atualizadas com as transações da instância principal**
  - Atualização assíncrona
- **Permite consultas de leitura no servidor standby**
- **Conflito: atualizações x consultas em execução**
- **Resolução de conflito configurável**

Hot Standby

- ***postgresql.conf* no servidor master:**

```
wal_level = hot_standby
archive_mode = on
archive_command = 'scp %p dextra02:/archives/%f'
archive_timeout = 120
```

- ***postgresql.conf* no servidor slave:**

```
hot_standby = on
```

- ***recovery.conf* no servidor slave ($PGDATA):**

```
standby_mode = 'true'
restore_command = 'cp /archives/%f %p'
trigger_file = '/tmp/arquivo_gatilho.pgsql'
```

Hot Standby

postgres@dextra02:~$ pg_ctl start

LOG: entering standby mode

LOG: restored log file "000000010000000000000000D" from archive

LOG: redo starts at 0/D000020

LOG: consistent recovery state reached at 0/E000000

LOG: database system is ready to accept read only connections

LOG: restored log file "000000010000000000000000E" from archive

cp: cannot stat `/archives/000000010000000000000000F': No such file or directory

LOG: restored log file "000000010000000000000000F" from archive

cp: cannot stat `/archives/000000010000000000000010': No such file or directory

Hot Standby

- **Servidor standby disponível em modo read-only:**

  postgres@dextra02:~$ psql pagila

  pagila=# UPDATE actor SET last_update = now();

  ERROR: cannot execute UPDATE in a read-only transaction

  STATEMENT: UPDATE actor SET last_update = now();

- **Restauração ocorre paralelamente às consultas**
- **Em caso de *failover*:**
  - Conexões ativas são mantidas e autorizadas a alterar a base de dados sem a necessidade de reconexão

Hot Standby: Limitações

- **Consultas nos servidores em modo standby podem causar conflitos com as operações de restauração que ocorrem em paralelo.**
- **Exemplo:**
  - Uma operação de VACUUM, ao ser replicada pode remover um registro “morto”, que esteja sendo utilizado por uma transação *read-only;*
  - Ao remover o registro, a consulta pode trazer resultados incorretos.
- **Há duas maneiras de resolver este problema:**
  - Parar a replicação até que a consulta finalize
  - Matar a consulta no servidor standby

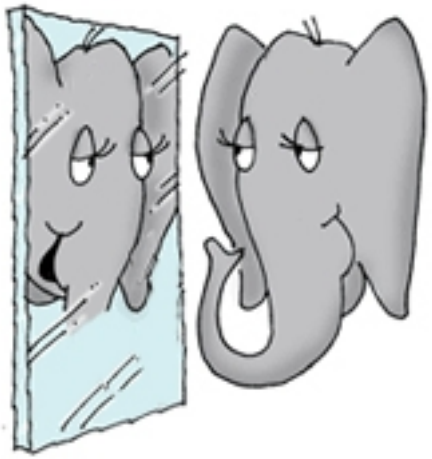
Hot Standby: Parâmetros

- **vacuum_defer_cleanup_age**
  - Especifica o número de transações de VACUUM que podem ser postergadas no servidor de produção, caso ocorra algum conflito nos servidores standby
  - Números altos podem:
    - *Eliminar os conflitos no servidor standby*
    - *Reduzir a eficiência do VACUUM no servidor de produção*
- **max_standby_archive_delay**
  - Especifica o tempo máximo para que as consultas que causam conflito no servidor standby sejam mortas
  - Este parâmetro é específico para a restauração baseada em arquivos do WAL. Padrão: 30 segundos

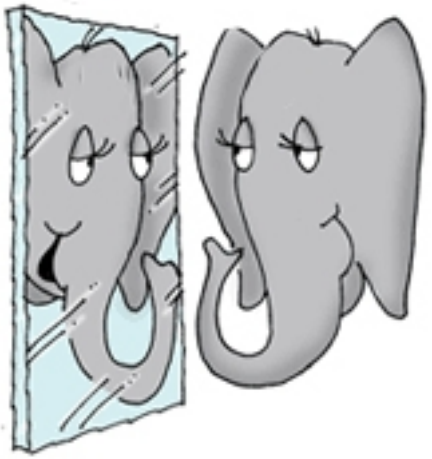
Streaming Replication

- Criação de um canal de comunicação via TCP/IP
- Comunicação constante entre os servidores master e slaves, agilizando a transferência de fragmentos de segmentos de log de transação
- Criação de processos **WalSender** e **WalReceiver**, iniciados nos servidores master e slaves respectivamente

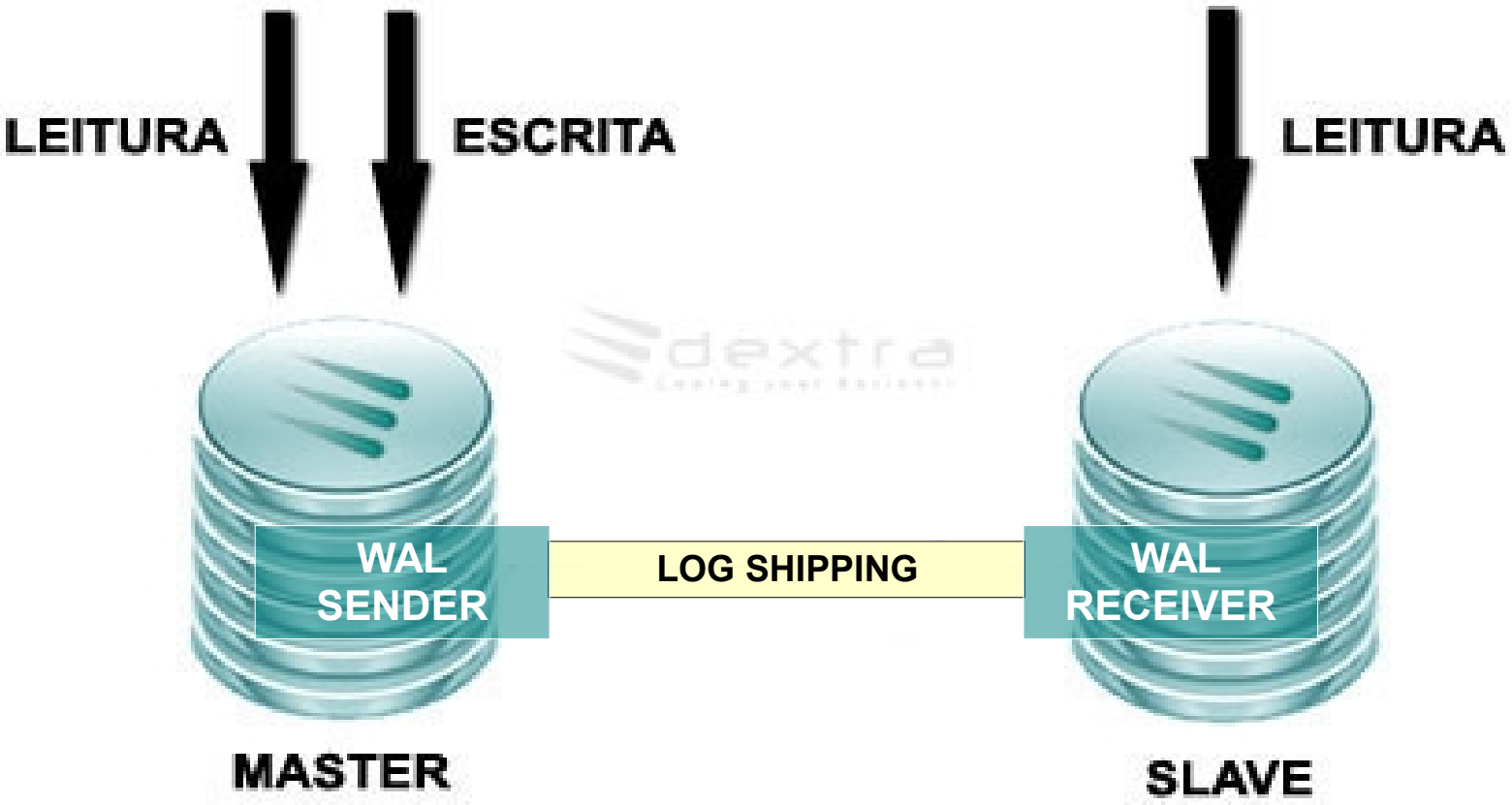
File-based Replication



Record-based Replication



Streaming Replication



Streaming Replication

- ***postgresql.conf* no servidor master:**

```
wal_level = archive
```

```
max_wal_senders = 3
```

  - *Define o número conexões concorrentes recebidas de servidores standby. Cada servidor standby utiliza uma conexão.*

```
#wal_sender_delay = 200ms
```

  - *Intervalo entre a operações de envio de fragmentos (stream) pelo processo WalSender. Padrão: 200 ms*

```
wal_keep_segments = 20
```

  - *Números de arquivos de logs de transação que devem ser mantidos para a recuperação, em caso de problemas com a restauração automática*

Streaming Replication

- ***pg_hba.conf* no servidor master:**

```
host    replication    all    192.168.40.128/32  trust
```

  - *Diretiva especial para que possa ser aceita a conexão do servidor standby*

- ***recovery.conf* no servidor slave ($PGDATA):**

```
standby_mode = 'true'
primary_conninfo = 'host=dextra01'
```

  - *String de conexão com o servidor de produção*

```
trigger_file = '/tmp/arquivo_gatilho.pgsql'
```

Streaming Replication

postgres@dextra02:~$ pg_ctl start

LOG: entering standby mode

LOG: consistent recovery state reached at 0/2002FFF0

LOG: unexpected pageaddr 0/1030000 in log file 0, segment 32, offset 196608

LOG: streaming replication successfully connected to primary

- ***Streaming Replication* isoladamente não disponibiliza o servidor para consultas, apenas sincroniza os servidores com os fragmentos (*streams*) dos logs de transação**

postgres@dextra02:~$ psql

FATAL: the database system is starting up

Streaming Replication

- **No servidor de produção, é criado o processo *WalSender*:**

  postgres 2568 2566 08:39 00:00:00 postgres: writer process

  postgres 2569 2566 08:39 00:00:00 postgres: wal writer process

  postgres 2572 2566 08:39 00:00:00 postgres: stats collector process

  postgres 3071 2566 0 09:01 ? 00:00:00 postgres: wal sender process postgres 192.168.40.128(39593) streaming 0/19000000

- **No standby, o processo *WalReceiver*:**

  postgres 4295 1 09:01 pts/0 00:00:00 /usr/local/bin/postgres

  postgres 4296 4295 09:01 00:00:00 postgres: startup process recovering 000000010000000000000020

  postgres 4298 4295 09:01 00:00:00 postgres: writer process

  postgres 4565 4295 09:09 00:00:00 postgres: wal receiver process streaming 0/2000E758

Streaming Replication

- **As alterações que ocorrem no servidor de produção são enviadas continuamente para os servidores standby.**
- **A cada atualização, o *stream* do arquivo de log é incrementado:**

postgres: startup process recovering 000000010000000000000020

postgres: wal receiver process ***streaming 0/2000E758***

postgres: startup process recovering 000000010000000000000020

postgres: wal receiver process ***streaming 0/20017BF4***

postgres: startup process recovering 000000010000000000000020

postgres: wal receiver process ***streaming 0/20026684***

Streaming Replication: Monitoramento

- **Informações sobre o servidor standby:**

pagila=# SELECT pg_last_xlog_receive_location();

0/23AC4360

  - Último fragmento recebido do servidor de produção

pagila=# SELECT pg_last_xlog_replay_location();

0/23AC4360

  - Último fragmento aplicado durante a recuperação
  - Valores idênticos para as duas funções indicam que não há informações pendentes a serem aplicadas

pagila=# SELECT pg_is_in_recovery();

true

  - Indica se o servidor PostgreSQL está em restauração

Streaming Replication: Monitoramento

- **O aplicativo *pg_controldata* exibe várias informações de controle do WAL:**
- **As seguintes informações podem ser comparadas entre os servidores master e slaves**

postgres@dextra02:~$ pg_controldata

...

Latest checkpoint location: *0/23AC6658*

...

Latest checkpoint's REDO location: *0/23AC6624*

Streaming Replication + Hot Standby

- ***postgresql.conf* no servidor master:**

```
wal_level = hot_standby
max_wal_senders = 3
wal_keep_segments = 20
```

- ***pg_hba.conf* no servidor master:**

```
host    replication    all    192.168.40.128/32   trust
```

- ***postgresql.conf* no servidor slave:**

```
hot_standby = on
```

- ***recovery.conf* no servidor slave ($PGDATA):**

```
standby_mode = 'true'
primary_conninfo = 'host=dextra01'
trigger_file = '/tmp/arquivo_gatilho.pgsql'
```

Streaming Replication + Hot Standby

- **max_standby_streaming_delay**
  - Idêntico ao parâmetro max_standby_archive_delay
  - Específico para Streaming Replication
- ***max_standby_delay = -1***
  - Aguarda a finalização das transações que causam conflitos no servidor standby
  - Ideal para ambientes onde as consultas são mais importantes do que a sincronização dos servidores
- ***max_standby_delay = 0***
  - Mata as consultas que causam conflito imediatamente
  - Ideal para ambientes onde a sincronização é mais importante do que as consultas do servidor standby

Cenários possíveis

- **Backup WAL + Hot Standby + Streaming Replication**
  - Possibilidade de Point In Time Recovery (PITR)
  - Geração de relatórios nos servidores standby
  - Utilização de ferramenta de balanceamento de carga para consultas
- **Streaming Replication**
  - Servidor standby atualizado a cada alteração
- **Utilização de ferramentas que automatizem o processo de failover:**
  - Heartbeat

Administração de privilégios de usuários

- **Facilidade para alterações de privilégios em massa:**

  *pagila=> SELECT * FROM dextra.actor;*

  ERROR: permission denied for relation actor

  *STATEMENT: SELECT * FROM dextra.actor;*

- **Concedendo permissão de consulta para todas as tabelas do schema *dextra* ao usuário *foobar*:**

  pagila=# GRANT SELECT ON ALL TABLES IN SCHEMA dextra TO foobar;

- **Até a versão 8.4, era possível apenas conceder/revogar privilégios especificando os nomes de todas as tabelas:**

  - Necessidade de gerar scripts SQL
  - Assistente de GRANT do PgAdmin

Administração de privilégios de usuários

- **Definição de privilégios padrões para futuros objetos**

pagila=> CREATE TABLE clientes();

pagila=> \c pagila foobar

You are now connected to database "pagila" as user "foobar".

*pagila=> SELECT * FROM dextra.clientes;*

ERROR: ***permission denied for relation clientes***

*STATEMENT: SELECT * FROM dextra.clientes;*

Concedendo permissão de consulta para todas as futuras tabelas do schema *dextra* ao usuário *foobar*:

pagila=# ALTER DEFAULT PRIVILEGES FOR ROLE dextra IN SCHEMA dextra GRANT SELECT ON TABLES TO foobar;

Blocos anônimos em PL/pgSQL

- **Capacidade de executar funções sem a necessidade de criá-las**
- **Todas as linguagens procedurais podem ser utilizadas em linha de comando**
- **Sintaxe:**

```
DO [ LANGUAGE nome_linguagem ] código
```

- **Facilita tarefas de administração**
  - Não há necessidade de CREATE/DROP FUNCTION
- **A estrutura das funções são mantidas:**

```
[ DECLARE ]
BEGIN
END
```

Blocos anônimos em PL/pgSQL

```
pagila=> DO $$
DECLARE
stmt text;
BEGIN
FOR stmt IN SELECT 'ALTER TABLE ' ||tablename|| ' ADD COLUMN
ultima_modificacao timestamp;' FROM pg_tables WHERE schemaname = 'dextra'
AND tablename NOT LIKE 'payment_%'
LOOP
    EXECUTE stmt;
END LOOP;
END
$$ LANGUAGE plpgsql;
DO
pagila=>
```

Triggers por colunas

- **Triggers disparadas com eventos de UPDATE em colunas**
  - Evita condições lógicas e comparação de valores no código da função
  - Não dispara em caso de atualização da coluna para o valor *default*

*CREATE TRIGGER tg_log_ativo BEFORE **UPDATE OF activebool** ON customer FOR EACH ROW EXECUTE PROCEDURE log_ativo();*

UPDATE customer SET activebool = false WHERE customer_id = 599;

```
pagila=# SELECT * FROM log_ativo WHERE customer_id = 599;
 id     | customer_id | first_name | activebool | last_modified
 600  |       599       | AUSTIN      | f                | 2010-08-28
```

Triggers condicionais

- **Comparação que define se a trigger será executada**
  - Reduz o número de execuções das funções de trigger
  - Elimina estruturas de condição do código da função

CREATE TRIGGER tg_log_ativo BEFORE UPDATE OF activebool ON customer FOR EACH ROW ***WHEN (OLD.activebool IS DISTINCT FROM NEW.activebool)*** *EXECUTE PROCEDURE log_ativo();*

```
pagila=# UPDATE customer SET activebool = true WHERE customer_id = 599;
pagila=# UPDATE customer SET activebool = true WHERE customer_id = 599;
pagila=# SELECT * FROM log_ativo WHERE customer_id = 599;
 id      | customer_id | first_name | activebool | last_modified
 606  |       599       | AUSTIN       | t                | 2010-08-28
```

Reimplementação do VACUUM FULL

- **O processo de VACUUM FULL é mais rápido, pois duplica a tabela, eliminando a original e recriando os índices**
  - Não é mais necessária a execução do REINDEX após o VACUUM FULL
  - Mais espaço em disco
- **VACUUM FULL 8.4**

  pagila=# VACUUM FULL actor;

  Time: ***217 613.377 ms***

  pagila=# REINDEX TABLE actor;

  Time: ***81 567.277 ms***

- **VACUUM FULL 9.0**

  pagila=# VACUUM FULL actor;

  Time: ***98 295.479 ms***

Contrib para migração de versões

- **A ferramenta pg_migrator está presente no contrib com o nome de pg_upgrade**
  - Permite a migração através dos *datafiles*, sem a necessida de dump/restore
  - Torna o processo de migração muito rápido
  - Somente para migração de 8.4 para 9.0

cd /usr/local/src/postgresql-9.0beta4/contrib/pg_upgrade_support
make; make install
cd ../pg_upgrade
make; make install

pg_upgrade -d /dados/postgresql8.4/ -D /dados/postgresql9.0/ -b /usr/local/pgsql/bin/ -B /usr/local/bin/ > upgrade.log

Melhorias de performance para queries geradas por ORMs

- **Dentre as melhorias no plano de execução de consultas, destaca-se a possibilidade de otimização com base na reescrita**
- **Consultas podem ser reescritas e eliminarem junções (JOINs) ineficientes**
- **Para ferramentas ORM (*Object Relational Mapping*) há uma grande melhoria de performance na geração de relatórios**
    - Este tipo de ferramentas possuem uma tendência para escrever junções desnecessárias

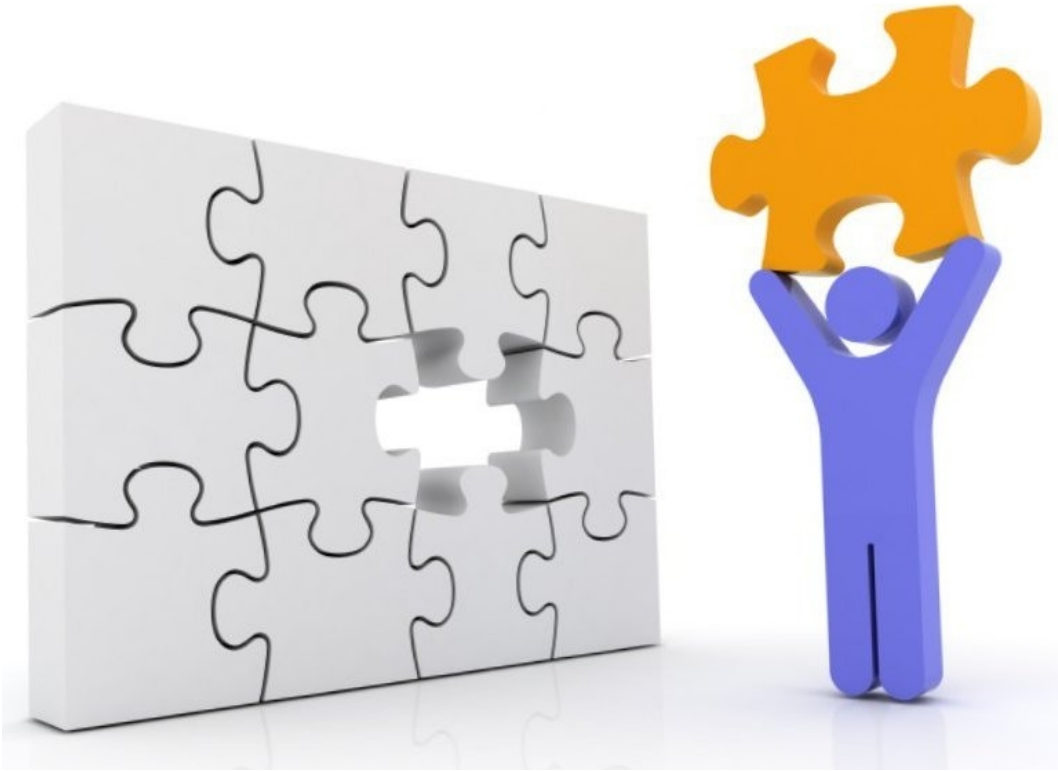
O que vem por aí...

PostgreSQL 9.1

- Acesso a fontes de dados externas via SQL
- Geração de backup base através da Streaming Replication
- Replicação síncrona
- Replicação em cascata

Cursos PostgreSQL

- PostgreSQL Essencial
- PL/pgSQL
- Administração
- Performance Tuning
- Alta Disponibilidade
- Postgis



www.dextra.com.br/treinamento

Matheus Espanhol

matheusespanhol@gmail.com

Dextra Sistemas

matheus.espanhol@dextra.com.br

www.dextra.com.br

São Paulo | Campinas | Campo Grande

(11) 2824-6722 | (19) 3256-6722