PostgreSQL Administração

Rodrigo Hjort
SERPRO/CETEC

v3.0

http://www.serpro.gov.br

Sobre o SERPRO

O **Serviço Federal de Processamento de Dados - SERPRO** é uma empresa pública vinculada ao Ministério da Fazenda.

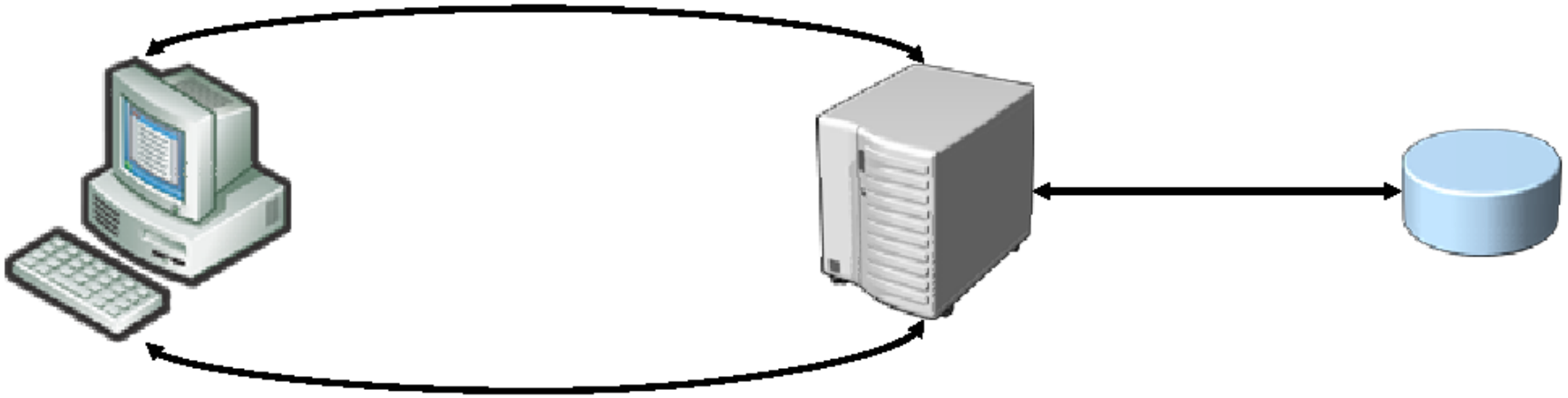
Foi criada em 1964 com o objetivo de modernizar e dar agilidade a setores estratégicos da Administração Pública brasileira prestando serviços em Tecnologia da Informação e Comunicações.

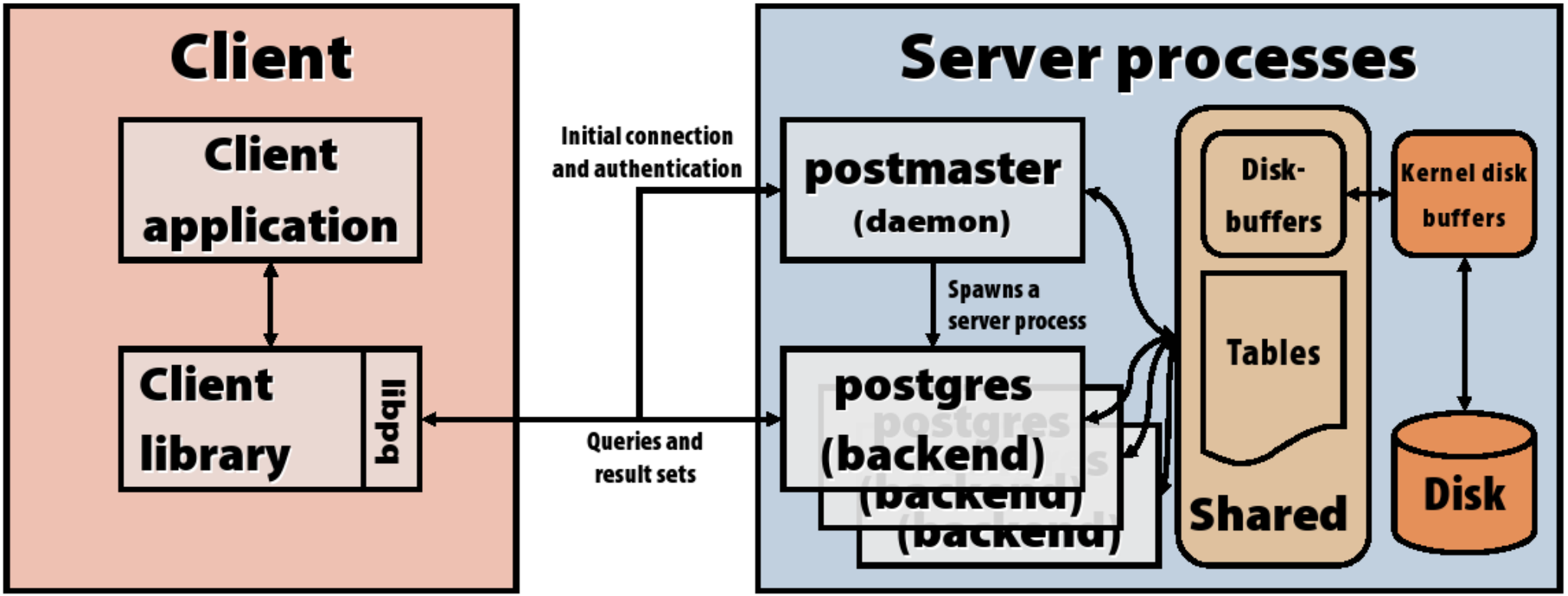
Agenda

1. A arquitetura do SGBD PostgreSQL
2. Preparando uma instância do PostgreSQL
3. Fixando parâmetros de configuração
4. Gerenciando bancos de dados e objetos
5. Explorando o catálogo (metadados)
6. Segurança Lógica: Autenticação e Autorização
7. Segurança Física: Backup e Restore

A arquitetura do SGBD PostgreSQL

Como funciona a conexão?





Os processos do PostgreSQL

SERPRO

- no Linux, cada componente do servidor PostgreSQL é representado por um processo
- no Windows, o PostgreSQL possui um aplicativo e diversas threads

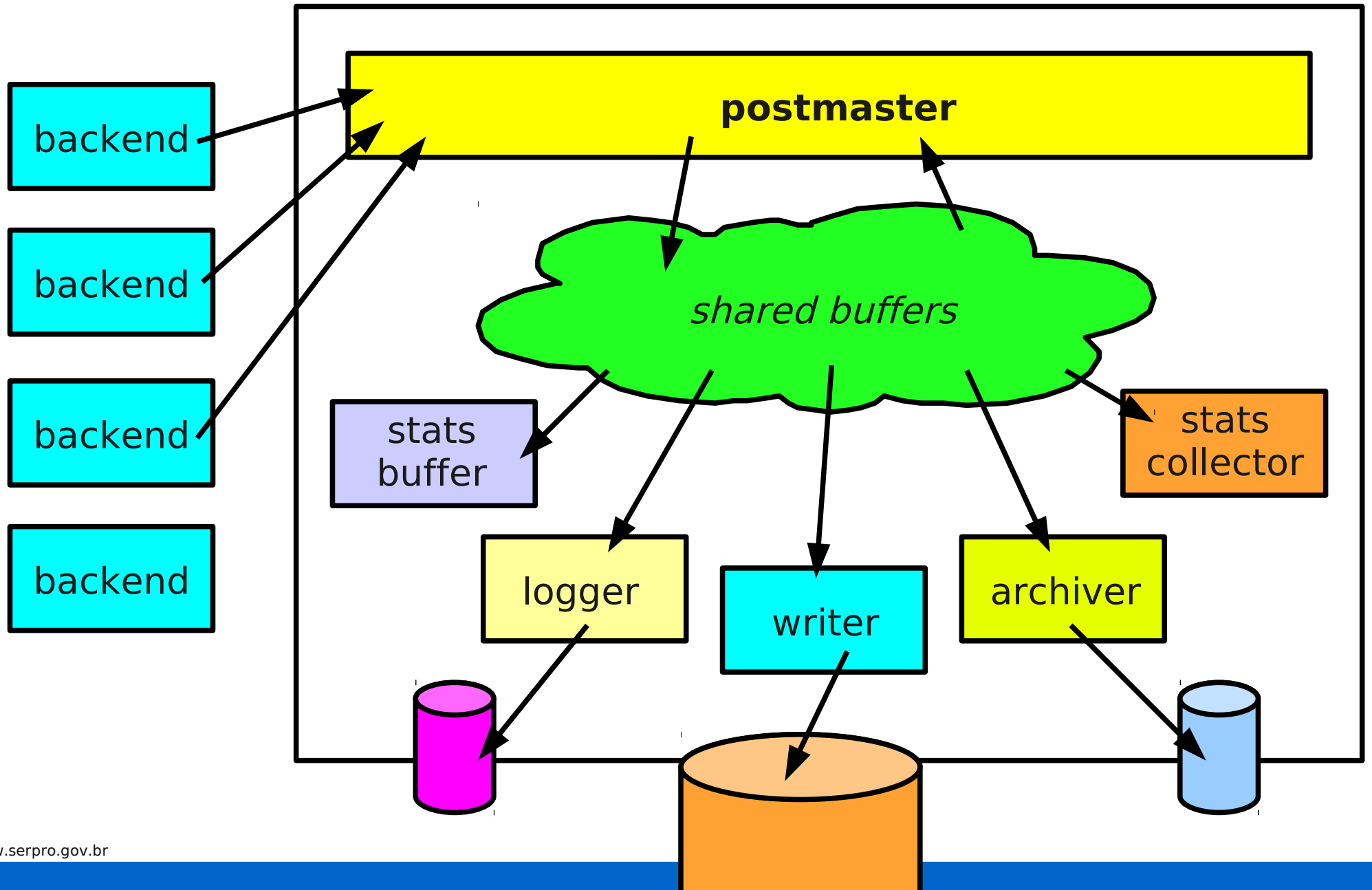
- visualizando pelo comando **top**:

```
$ top -u postgres
```

- visualizando pelo comando **ps**:

```
$ ps aux | grep ^postgres
```

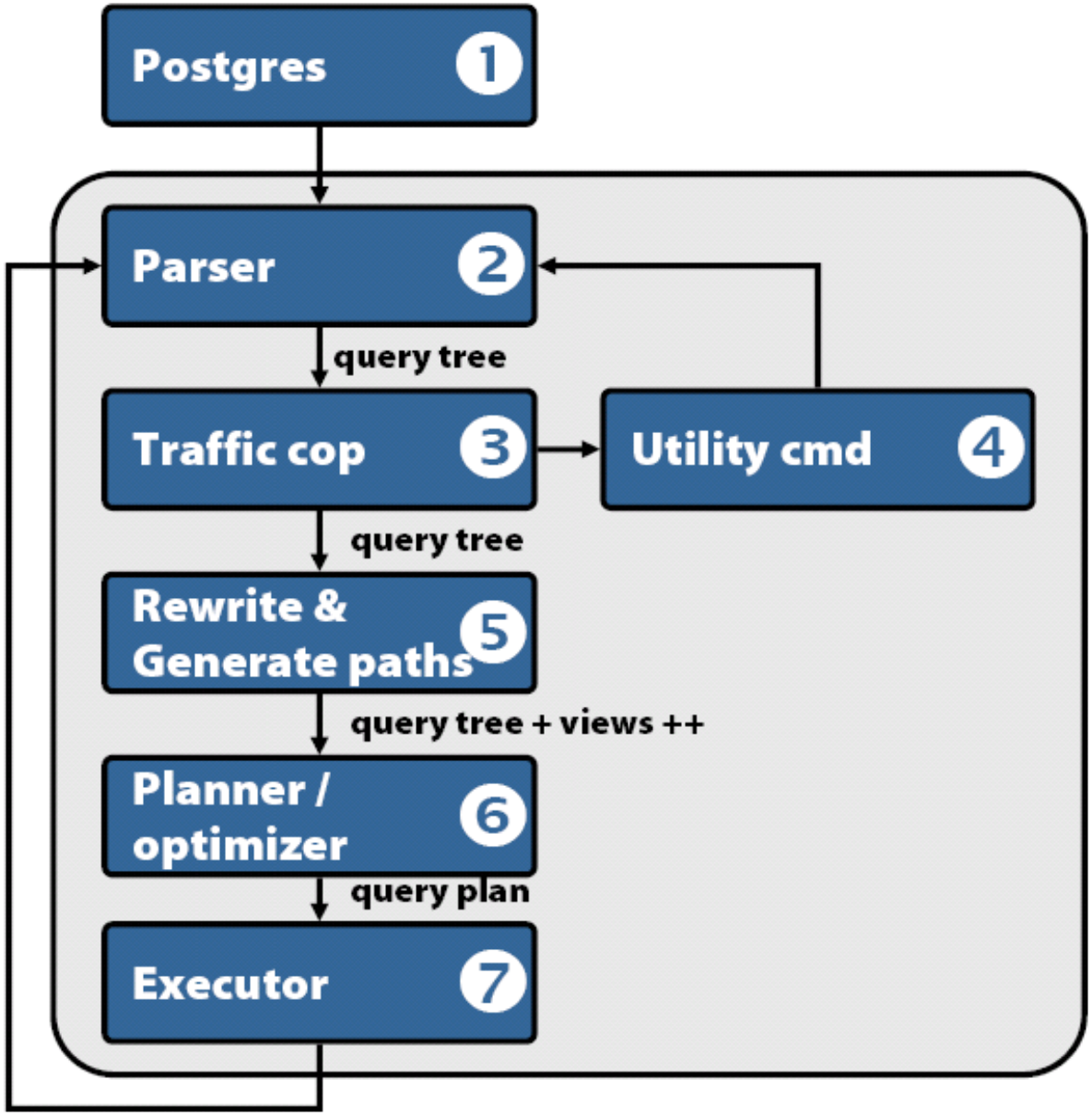
Os processos do PostgreSQL



Os processos auxiliares

- postmaster: processo principal do servidor
- logger: grava logs de transação *[LGWR]*
- writer: realiza a escrita de páginas sujas definitivamente no disco *[DBWR]*
- archiver: arquivador de logs *[ARCn]*
- stats buffer: buffer de estatísticas
- stats collector: coletor de estatísticas
- postgres (backend): conexões clientes

Qual o caminho de uma query?



1. A consulta chega por um soquete e é jogada numa string
2. Lex/Yacc corta a string e a consulta é analisada e devidamente identificada
3. Verifica se a consulta é complexa e utiliza comando auxiliar
4. Executa comando e volta
5. Aplica regras, views, etc
6. Escolhe o melhor plano de execução e envia ao *executor*
7. Executa a consulta, coleta dados, ordena, qualifica e retorna as linhas

SERPRO

Preparando uma instância do PostgreSQL

www.serpro.gov.br

Gerenciando o servidor

SERPRO

- inicializando o PostgreSQL:

```
/etc/init.d/postgresql-8.3 start
```

- finalizando o PostgreSQL:

```
/etc/init.d/postgresql-8.3 stop
```

- outros comandos:

```
/usr/lib/postgresql/8.3/bin/postmaster
/usr/lib/postgresql/8.3/bin/pg_ctl
```

Criando o diretório de dados

- o conjunto de diretórios e arquivos de dados do PostgreSQL é chamado *cluster*

- apesar de gerado automaticamente na instalação, podemos criar o cluster de forma manual

- criando um cluster PostgreSQL:

```
$ initdb -E codificação -D diretório
$ initdb -E LATIN1 -D /tmp/pgsql
```

Definindo a codificação

- diversas codificações são suportadas, permitindo fácil internacionalização
- mais adequada: UTF8 (Unicode)

- especificando na criação do cluster

```
$ initdb -E LATIN1
```

- especificando na criação do banco

```
$ createdb -E UTF8 nova_base
```

Por que definir corretamente?

- somente serão aceitos caracteres que estejam contemplados na codificação configurada
- a codificação influencia na ordenação de nomes e textos em geral
- outras codificações possíveis:
  - SQL_ASCII
  - LATIN1
  - ISO_8859_5

Fixando parâmetros de configuração

O arquivo de configurações

- as configurações do servidor PostgreSQL estão definidas e armazenadas no arquivo:

  `$PGDATA/postgresql.conf`

- os parâmetros mais importantes somente podem ser definidos no momento de inicialização do banco

- para visualizar a configuração via SQL:

  `SHOW ALL;`

  `SHOW` *`parâmetro;`*

O formato do arquivo .conf

SERPRO

- o arquivo consiste de linhas do tipo
  - nome_do_parâmetro = valor
- comentários são definidos com “#” no início da linha
- o arquivo é subdividido em áreas comuns:
  - conexões e autenticação, utilização de recursos, WAL, query tuning, erro e log, estatísticas, autovacuum, opções do cliente

Como alterar parâmetros?

- reinicializando o PostgreSQL:

```
/etc/init.d/postgresql-8.3 restart
```

- enviando um sinal ao postmaster:

```
/etc/init.d/postgresql-8.3 reload
```

- alterando em tempo de sessão:

```
SHOW parâmetro;
SET parâmetro TO valor;
SET parâmetro TO DEFAULT;
```

Usuário ou banco específico

- Fixar parâmetros para um usuário:

```
ALTER USER usuário
  SET parâmetro TO valor;

ALTER USER usuário RESET parâmetro;

ALTER USER usuário RESET ALL;
```

- Fixar parâmetros para um banco:

```
ALTER DATABASE banco
  SET parâmetro TO valor;

ALTER DATABASE banco RESET parâmetro;

ALTER DATABASE banco RESET ALL;
```

Gerenciando bancos de dados e seus objetos

Criando um banco de dados

SERPRO

- somente determinados usuários podem criar um banco de dados

- criando um banco de dados via Shell:

```
$ createdb -E codificação -T modelo nome
$ createdb -T template0 -O dono novobd
```

- criando um banco de dados via SQL:

```
CREATE DATABASE nome OWNER dono
  TEMPLATE modelo ENCODING codificação;
```

Removendo um banco de dados

- um banco de dados não pode ser removido quando há conexões abertas nele

- removendo um banco de dados via Shell:

```
$ dropdb nome
```

- removendo um banco de dados via SQL:

```
DROP DATABASE nome;
```

Os bancos template

- na criação do banco é especificado o modelo (*template*), do qual são copiados estrutura e dados
- se adicionarmos objetos a um banco de dados de modelo, nos bancos criados posteriormente existirão esses objetos
- modelo padrão: *template1*
- modelo zerado: *template0*
- não pode haver conexão no modelo!

Os esquemas

- uma forma de separar os objetos dentro do banco de dados, tal como diretórios em uma hierarquia de arquivos
- esquemas são usados para:
  - agrupar de modo lógico os objetos de um banco de dados
  - implantar a segurança dos objetos de acordo com os usuários ou papéis (*roles*)
  - evitar colisões de nomes em bancos de dados grandes

Manipulando schemas

SERPRO

- criando um esquema:

```
CREATE SCHEMA historico;
```

- criando objetos dentro do esquema:

```
CREATE TABLE historico.log (
  id serial, descricao text);
```

- usando o caminho de busca:

```
SHOW search_path;
```

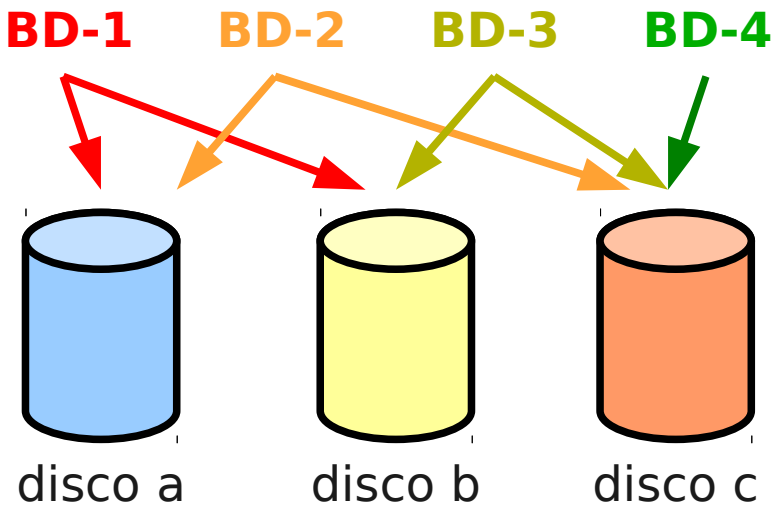
```
SET search_path TO historico, public;
```

- excluindo o esquema (em cascata):

```
DROP SCHEMA historico CASCADE;
```

Os espaços de tabela

- um *tablespace* nos permite especificar onde serão armazenados fisicamente:
  - bancos de dados
  - tabelas
  - índices



- vantagens:
  - granularidade a nível de objeto
  - aumenta o desempenho e controle sobre a utilização de cada disco rígido
  - maior flexibilidade para gerenciar espaço

Manipulando tablespaces

SERPRO

- criando um tablespace:

```
CREATE TABLESPACE tsfast LOCATION
  '/var/data/postgres/tsfast';
```

- associando objetos ao tablespace:

```
CREATE TABLE foo (id int)
  TABLESPACE tsfast;

ALTER TABLE foo
  SET TABLESPACE pg_default;
```

- excluindo o tablespace (se vazio):

```
DROP TABLESPACE tsfast;
```

Explorando o catálogo (metadados)

As tabelas de sistema

- o PostgreSQL utiliza intensamente o recurso de dicionário de dados (ou catálogo) para armazenar e controlar as atividades sobre o banco de dados
- a maioria das tabelas possui o campo OID que armazena o identificador único do objeto em questão
- as tabelas e visões do dicionário de dados são prefixadas com “pg_” e, em sua maior parte, são acessíveis a qualquer usuário

Tabelas de sistema 1/4

SERPRO

- pg_shadow e pg_user
  - informações sobre os usuários
- pg_group
  - informações sobre os grupos de usuários
- pg_database
  - informações sobre todos os bancos de dados
- pg_class
  - definições de relações (tabelas, índices, visões e seqüências)

Tabelas de sistema 2/4

SERPRO

- pg_attribute
  - informações sobre as colunas das tabelas
- pg_attrdef
  - valores padrões das colunas das tabelas
- pg_constraint
  - definições das restrições de tabelas e colunas
- pg_proc
  - informações das funções

Tabelas de sistema 3/4

SERPRO

- pg_trigger
  - informações dos disparadores (triggers)
- pg_depend
  - informações sobre dependências entre os objetos
- pg_namespace
  - informações sobre os esquemas
- pg_rewrite
  - informações sobre as regras (rules)

Tabelas de sistema 4/4

- pg_indexes
  - definições dos índices
- pg_views
  - definições das visões (views)
- pg_locks
  - informações sobre os bloqueios (locks) existentes
  - um lock pode aparecer múltiplas vezes, pois contém uma linha para cada objeto, modo de lock e transações envolvidas

SERPRO

Segurança Lógica: Autenticação e Autorização

Autenticação no PostgreSQL

- **autenticação** é o processo pelo qual o servidor de banco de dados estabelece a identidade do cliente e determina onde a aplicação cliente pode se conectar com o usuário informado
- o PostgreSQL oferece diferentes métodos de autenticação de clientes, sendo que estes baseiam-se no IP do cliente, no banco de dados e no usuário e senha informados

O arquivo pg_hba.conf

SERPRO

- no PostgreSQL a autenticação de clientes é configurada no arquivo:

  `$PGDATA/pg_hba.conf`

- importante: o arquivo é lido de cima para baixo, e o primeiro registro que se enquadra na requisição é considerado!
- existem diversas formas de registro:

  local *banco usuário método*

  host *banco endereço usuário método*

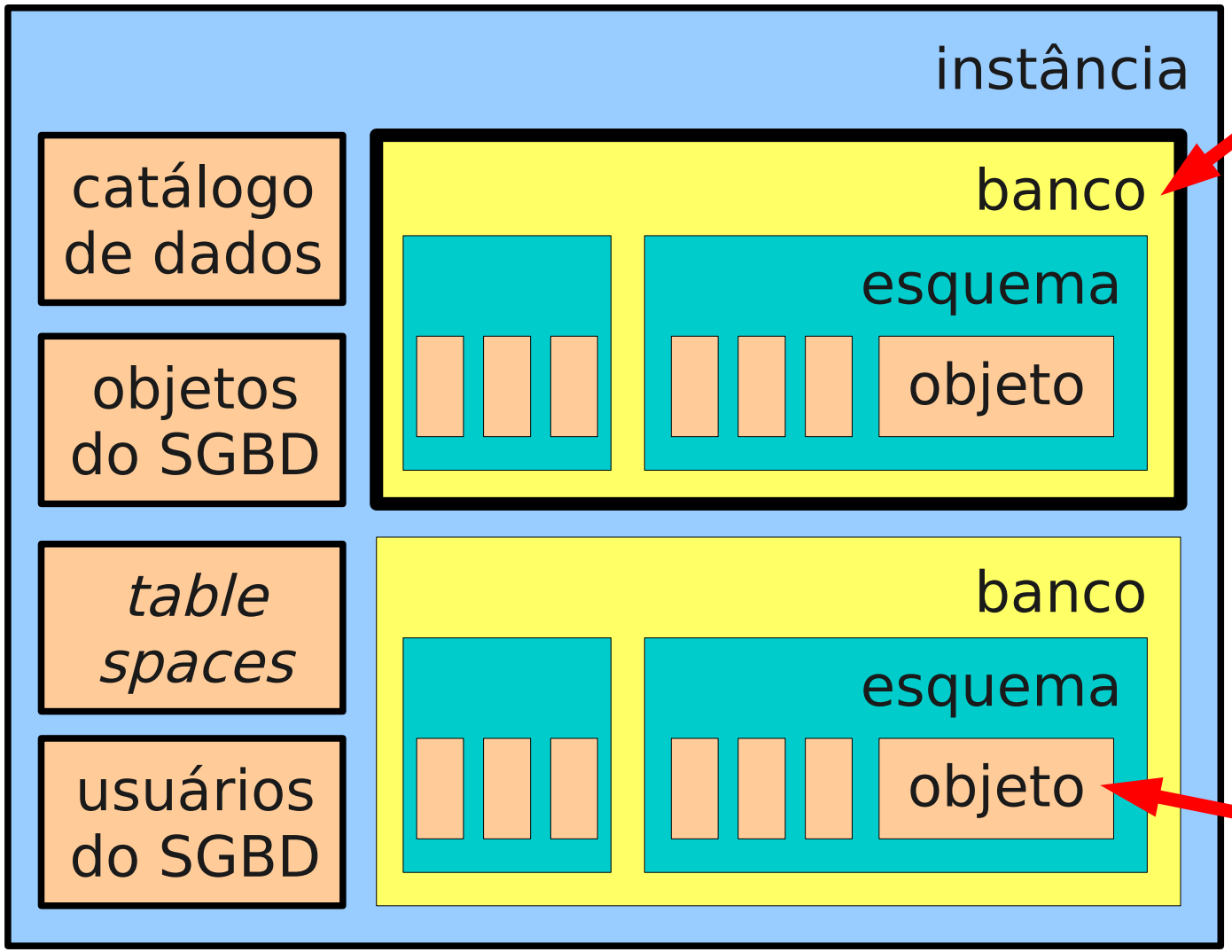
  hostssl *banco endereço usuário método*

Editando o pg_hba.conf

SERPRO

- forma de autenticação
  - local (Unix Sockets), host e hostssl (TCP/IP)
- banco / usuário
  - banco de dados e usuário a se conectar (podem utilizar o valor “all” para todos)
- endereço
  - endereço IP e máscara de rede (CIDR address)
- método
  - trust, reject, password, md5, crypt, ident

Objetos no PostgreSQL



1. conexão ocorre em um único banco de dados e para um usuário
2. não existem sinônimos!
3. acesso a objetos de outro esquema com o recurso de *search path*
4. nativamente, não há como acessar objetos de outro banco

Usuários e permissões

- apesar das informações de usuários e grupos poderem existir dentro do sistema operacional, o PostgreSQL tem a sua própria estrutura para armazenar estas informações
- os usuários do PostgreSQL são globais ao *cluster* e não para cada banco de dados
- conceito de papéis (*roles*): mesmo tratamento para usuários e grupos

Usuários e grupos

SERPRO

- criando um usuário e grupo:

```
CREATE USER sa_curso PASSWORD 'senha';
CREATE GROUP admin WITH USER sa_curso;
```

- alterando um usuário:

```
ALTER USER sa_curso SUPERUSER;
```

- excluindo um usuário e grupo:

```
DROP USER sa_curso;
DROP GROUP admin;
```

Controle de acesso aos objetos

- quando um objeto é criado o usuário atual passa a ser o seu proprietário, tendo acesso irrestrito a ele
- para que outros usuários possam acessar estes objetos, é preciso conceder certos privilégios através dos comandos GRANT e REVOKE
- alguns tipos de privilégios concebíveis:
  - SELECT, INSERT, UPDATE, DELETE
  - CREATE, EXECUTE, USAGE

Manipulando privilégios

SERPRO

- concedendo privilégios:

```
GRANT [operação] ON [objeto] TO [papel];
GRANT ALL ON tabela TO usuario;
GRANT SELECT ON tabela TO PUBLIC;
GRANT CREATE ON schema_log TO admins;
```

- removendo privilégios:

```
REVOKE [oper] ON [objeto] FROM [papel];
REVOKE ALL ON tabela FROM usuario;
REVOKE DELETE ON tabela FROM PUBLIC;
```

Segurança Física: Backup e Restore

Backup no PostgreSQL

SERPRO

- existem três maneiras de se realizar o backup de um servidor de banco de dados PostgreSQL:
  - backup lógico online: via SQL Dumps (restauração portável, porém mais lenta)
  - backup físico offline: via Sistema de Arquivos e comandos como tar, gzip, bzip2 (servidor precisa ser desativado)
  - backup físico online: via Archiving e PITR (restauração mais rápida, mas não-portável)

Backup e restore com dumps

- efetuando e analisando o backup lógico:

```
$ pg_dump curso | less
$ pg_dump curso > curso.sql
$ pg_dump curso | gzip -9 > curso.gz
```

- visualizando e restaurando o dump:

```
$ zless curso.gz
$ createdb novocurso -T template0
$ zcat curso.gz | psql novocurso
$ psql novocurso -f curso.sql
```

Write Ahead Log (WAL)

SERPRO

- técnica padrão para log transacional
  - um COMMIT é gravado instantaneamente no log de transações (WAL)
  - alterações nos arquivos de dados são feitas de modo assíncrono por outro processo
- vantagens:
  - o arquivo de log é gravado sequencialmente
  - a consistência dos arquivos de dados é garantida
  - possibilita backup full online e point-in-time recovery (PITR)

Point-in-Time Recovery

SERPRO

- backup contínuo dos dados
  - possibilita backup completo dos dados binários sem desligar ou desativar o servidor
  - cada log de transação gerado (WAL) é copiado para outro lugar via archiving (LOG shipping)
  - restauração consiste em utilizar esses arquivos, processados sequencialmente até a data/hora desejada

Backup físico online

SERPRO

- habilitar parâmetro de configuração:

```
archive_command = 'cp -i %p
  /mnt/server/archivedir/%f </dev/null'
```

- iniciar o backup:

```
SELECT pg_start_backup('label');
```

- fazer as cópias físicas dos arquivos
- parar o backup:

```
SELECT pg_stop_backup();
```

Restore de backup físico

SERPRO

- parar o servidor PostgreSQL
- extrair backup para o diretório de dados
- copiar archives pendentes para pg_xlog/
- criar arquivo recovery.conf:

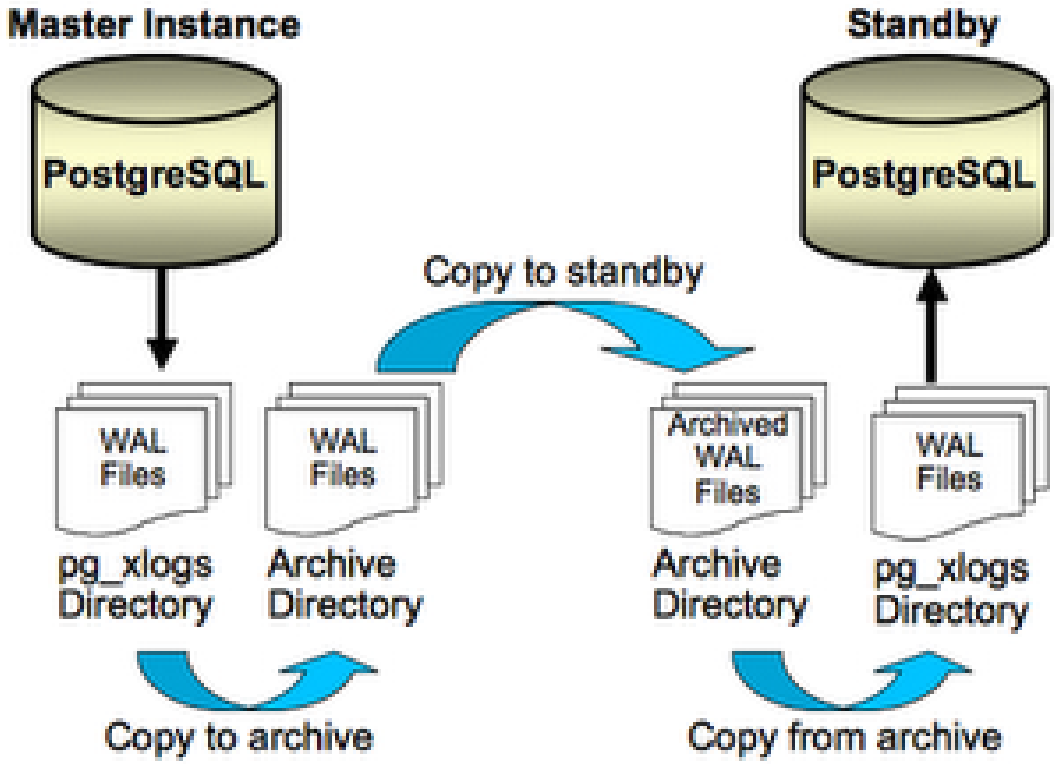
```
restore_command = 'cp
  /mnt/server/archivedir/%f %p'

#recovery_target_time = '…'
```

- rever as permissões dos arquivos
- iniciar o servidor PostgreSQL

Warm Standby (Log Shipping)

- cluster de alta disponibilidade com servidores standby prontos para retomar as operações se o servidor primário falhar



Bibliografia

PostgreSQL 8.3 Documentation

http://postgresql.org/docs/8.3/

Rodrigo Hjort
rodrigo@hjort.co