CHINA POSTGRESQL ASSOCIATION

中国PostgreSQL培训认证课程之应用管理基础

中国PG分会 魏波
boluo_wei@126.com

1.PostgreSQL安装卸载

安装方式

◆ 在Linux安装PostgreSQL

二进制安装包安装
通过yum安装
通过源码编译安装

源码安装可根据需要制定多种选项

源码的安装一般由3个步骤组成：
配置(configure)
编译(make或 gmake)
安装(make install或 gmake install)

获得介质

程序类型	操作系统		获取方式
发行版	BSD	FreeBSD	参考：https://www.postgresql.org/download/freebsd/
		OpenBSD	参考：https://www.postgresql.org/download/openbsd/
	Linux	RedHat	如何配置软件源：https://www.postgresql.org/download/linux/redhat/
		Debian	如何配置软件源：https://www.postgresql.org/download/linux/debian/
		Ubuntu	如何配置软件源：https://www.postgresql.org/download/linux/ubuntu/
		SuSE	https://software.opensuse.org/search?utf8=%E2%9C%93&q=postgresql&search_devel=false&search_unsupported=false&baseproject=openSUSE%3ALeap%3A42.2
		Others	参考：https://www.postgresql.org/download/linux/
	macOS		BigSQL：https://www.openscg.com/bigsql/postgresql/installers.jsp/
			EDB：https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#linux
	Solaris		https://www.postgresql.org/ftp/binary/
	Windows		HGDB：http://www.highgo.com/content.php?catid=72#fl1
源码包			https://www.postgresql.org/ftp/source/

安装方式

◆ 在Linux安装PostgreSQL

1. 二进制安装包安装
2. 通过yum安装
3. 通过源码编译安装

源码的安装一般由3个步骤组成：

配置(configure)

编译(make)

安装(make install)

安装方式-源码安装

1）编译前依赖包，以RedHat/CentOS为例：

最小依赖：gcc、gcc-c++、zlib-devel、readline-devel

其他依赖：perl-ExtUtils-Embed、pam-devel、libxml2-devel、libxslt-devel、openldap-devel、python-devel、openssl-devel、cmake

2）下载源码解压后，通过configure、make、make install进行编译安装

安装方式-源码安装

- configure 选项

  命令./configure –help可以输出详细的选项列表

prefix：指定安装路径
with-openssl：对openssl进行扩展支持
with-python：对python进行扩展支持
with-perl：对perl进行扩展支持
with-libxml：对xml进行扩展支持

初始化（initdb）

- 数据库集簇“cluster”，是一个包含所有数据文件、配置文件的“data”目录
- 一台物理服务器上可以有多个cluster，通过以下方式进行区分：

  data路径

  port端口号
- 编译安装的数据库，需要initdb命令手动初始化该data目录，常用参数如下：

  -D：data目录的路径

  -U：数据库超级用户名

  -E：配置区域语言、字符集

编译安装如何卸载

- 编译安装后如何卸载
  - 一般的，清除安装目录，再次安装可以尝试重新指定路径（安装时指定安装目录）
  - 在安装的时候加 prefix ，不然的话虽然用find命令可以找出一些相关文件，但是对于修改了配置，比如加入启动项之类的操作就不好根除
  - 通常只能用find去手动清除。所以软件通常都会提供uninstall的程序来执行卸载操作。

示例-源码安装PG V10.3

源码目录

```
[postgres@localhost postgresql-10.3]$ ll
total 668
-rw-r--r--      1 postgres postgres    457    Nov  7 08:46 aclocal.m4

drwxrwxr-x   2 postgres postgres   4096   Nov  7 08:49 config

-rwxr-xr-x     1 postgres postgres 495587 Nov  7 08:46 configure          源码安装的配置脚本

-rw-r--r--      1 postgres postgres  76386  Nov  7 08:46 configure.in

drwxrwxr-x 55 postgres postgres   4096   Nov  7 08:49 contrib            已打包到PG源码中的第三方贡献的插件源码
-rw-r--r--      1 postgres postgres   1192   Nov  7 08:46 COPYRIGHT

drwxrwxr-x   3 postgres postgres    101    Nov  7 08:49 doc                文档
-rw-r--r--      1 postgres postgres   3638   Nov  7 08:46 GNUmakefile.in

-rw-r--r--      1 postgres postgres    284    Nov  7 08:46 HISTORY          版本变更的历史记录
-rw-r--r--      1 postgres postgres  71584  Nov  7 08:52 INSTALL           安装说明

-rw-r--r--      1 postgres postgres   1529   Nov  7 08:46 Makefile
-rw-r--r--      1 postgres postgres   1212   Nov  7 08:46 README
drwxrwxr-x 16 postgres postgres   4096   Nov  7 08:52 src                源代码
```

示例-源码安装PG V10.3

1）配置

```
./configure --prefix=/usr/local/pgl103 --with-pgport=1922  --with-openssl --with-perl --with-tcl --with-python --with-pam --without-ldap --with-libxml --with-libxslt --enable-thread-safety --with-wal-blocksize=16 --with-blocksize=8 --enable-dtrace --enable-debug
```

注意：

--with-blocksize

- 如果数据库需要经常做插入的操作，数据量增长非常快，尽量把此参数设大一点；
- 经常做小数据查询、更新且内存不是非常大的时候可以设小一点，默认8K即可。
- 生产环境不要加--enable-dtrace --enable-debug。

示例-源码安装PG V10.3

2）编译

```
gmake world          --包括第三方插件全部编译
gmake check-world    --这个需要使用普通用户执行，可选，耗时较长
```

3）安装

```
gmake install-world    -- 包括第三方插件全部安装
```

注意：
如果遇到依赖的动态库缺失，需要提前安装即可
gmake world安装包含了文档，所有的contirb

示例-源码安装PG V10.3

4）创建普通用户启动数据库，超级用户不能启动数据库，若使用超级用户启动，需要修改源码，对root用户检测的一些东西，并修改它的profile

```
useradd postgres
su - postgres
vi ~/.bashrc(bash_profile)
#add
export PGPORT=1922
export PG_HOME=/usr/local/pgl103
export PATH=$PG_HOME/bin:$PATH
export PGDATA=$PG_HOME/data
export LD_LIBRARY_PATH=$PG_HOME/lib
export LANG=en_US.utf8

source ~/.bashrc
```

示例-源码安装PG V10.3

5）初始化

```
initdb -D $PGDATA -E UTF8 --locale=C -U postgres -W
```

注意：
加-W参数会提示输入数据库的超级用户的密码，默认情况下密码为空。

6）启动数据库前修改配置文件
pg_hba.conf用于配置控制访问数据库的来源
postgresql.conf是数据库的主配置文件，最好也调整一下Linux内核参数

示例-源码安装PG V10.3

Linux系统内核调整

```
vi /etc/sysctl.conf
#add
kernel.shmmax = 68719476736 （默认）          #最大共享内存段大小
kernel.shmall = 4294967296 （默认）           #可以使用的共享内存的总量
kernel.shmmni = 4096                          #整个系统共享内存段的最大数目
kernel.sem = 50100 64128000 50100 1280        #每个信号对象集的最大信号对象数；系统范围内
                                              #最大信号对象数；每个信号对象支持的最大操作数；
                                              #系统范围内最大信号对象集数
fs.file-max = 7672460                         #文件句柄的最大数量。
                                              #文件句柄设置表示在linux系统中可以打开的文件数量
net.ipv4.ip_local_port_range = 9000 65000     #应用程序可使用的IPv4端口范围
net.core.rmem_default = 1048576               #套接字接收缓冲区大小的缺省值
net.core.wmem_default = 262144                #套接字发送缓冲区大小的缺省值
net.core.wmem_max = 1048576                   #套接字发送缓冲区大小的最大值

sysctl -p                    #使配置生效
```

示例-源码安装PG V10.3

Linux资源限制调整

```
vi /etc/security/limits.conf
#add
* soft nofile 131072                    #当前系统生效的打开文件的数目
* hard nofile 131072                    #系统中所能设定的打开文件的最大值
* soft nproc  131072                    #当前系统生效的进程的数目
* hard nproc  131072                    #系统中所能设定的进程数目的最大值
* soft core   unlimited                 #当前系统生效的内核文件的大小
* hard core   unlimited                 #系统中所能设定的内核文件的最大值
* soft memlock 50000000                 #当前系统生效的锁定内存地址空间的大小
* hard memlock 50000000                 #系统中所能设定的的锁定内存地址空间最大值
```

示例-源码安装PG V10.3

系统防火墙配置（或者直接关闭防火墙）

```
vi /etc/sysconfig/iptables-config
-A INPUT -i lo -j ACCEPT
#允许源IP
-A INPUT -s 192.168.0.0/16 -j ACCEPT
#允许源IP访问目标端口
-A INPUT -s 192.168.1.0/24 -m state --state NEW -m tcp -p tcp --dport 1922 -j ACCEPT
#允许任意IP访问目标端口
-A INPUT -p tcp -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT
```

2.PostgreSQL服务管理

相关概念

- 数据库服务（集簇服务\集群服务\实例）
  - 多数操作系统上，PostgreSQL 是作为一种服务（或者叫守护进程）安装的
  - 多个 PostgreSQL 服务可以运行于同一台物理服务器上
  - 它们的侦听端口不能重复，也不能共享同一个数据存储目录
  - 多数情况下，在一台物理服务器上只会安装一个服务

- DATABASE
  - 每个 PostgreSQL 服务可以包含多个独立的 database

自动服务管理（Linux）

- 发行版安装的数据库，在安装完成之后会自动添加数据库服务
- 编译安装的数据库添加自启动服务

  1）将“源码包/contrib/start-scripts”目录下的脚本重命名后放到/etc/init.d目录下

  2）通过下面的命令添加到服务中

  chkconfig --add ***ScriptName***

  相应的，可通过下述命令删除

  chkconfig --del ***ScriptName***

命令行服务管理

- 通过pg_ctl命令来实现数据库的启动、关闭、加载管理
- 在线帮助 pg_ctl --help
- 主要选项
  - start：启动
  - stop：关闭
  - reload：重新加载
  - restart：重启

命令行服务管理

- pg_ctl stop关闭数据库时，可以通过“-m”参数定义关闭方式
  - smart：缺省模式，等待所有客户端断开
  - fast：　强制模式，对未断开的客户端进行回滚
  - immediate：强制模式，但不回滚，重启时需要自动恢复

命令行服务管理

- 关于pg_ctl命令的几个常用参数解释如下：

  -D：数据库data目录的路径

  -w：等待启动或关闭完成，等待是关闭的默认选项，但不是启动的默认选项

  -t： 等待时间

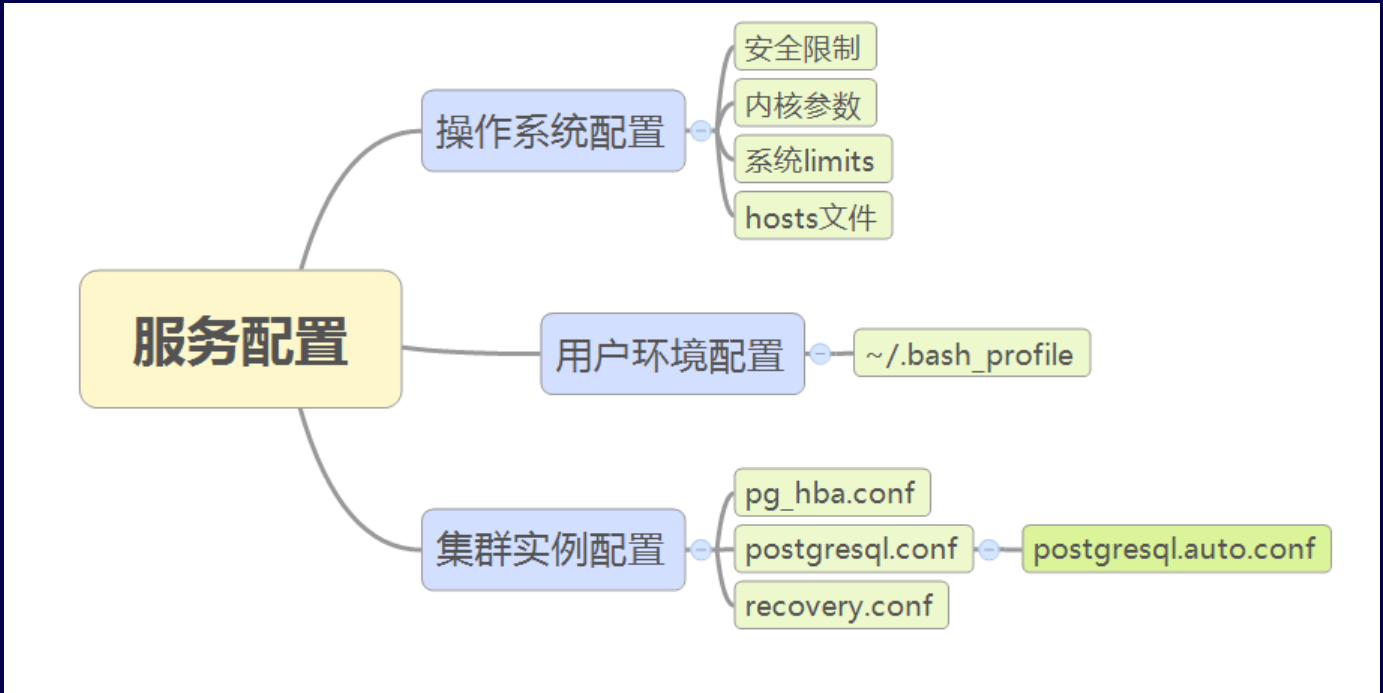
  -W：不等待命令成功完成

  -l：指定服务器日志记录文件

命令行服务管理

```
pg_ctl start   [-w] [-t SECS] [-D DATADIR] [-s] [-l FILENAME] [-o "OPTIONS"]
pg_ctl stop    [-W] [-t SECS] [-D DATADIR] [-s] [-m SHUTDOWN-MODE]
pg_ctl restart [-w] [-t SECS] [-D DATADIR] [-s] [-m SHUTDOWN-MODE] [-o
"OPTIONS"]
pg_ctl reload  [-D DATADIR] [-s]
pg_ctl status  [-D DATADIR]
pg_ctl promote [-D DATADIR] [-s]
pg_ctl kill  SIGNALNAME PID
```

服务配置


服务配置
操作系统配置
安全限制
内核参数
系统limits
hosts文件
用户环境配置
~/.bash_profile
集群实例配置
pg_hba.conf
postgresql.conf
postgresql.auto.conf
recovery.conf

用户环境配置（.bash_profile）

- 当安装完成后，用户如果想直接用命令行进行操作，需要进行环境变量的设置
- 否则直接执行psql、pg_ctl等操作，会提示命令无法找到
- 以下操作过程结束后，就可以直接使用psql连接数据库了

```
# su -postgres
vi  ~/.bash_profile
export PGHOME=/data/postgres/9.5.7
export PATH=$PGHOME/bin:$PATH
export PGDATA=$PGHOME/data
export LD_LIBRARY_PATH=$PGHOME/lib
source ~/.bash_profile
```

服务配置文件

文件名	主要作用
pg_hba.conf	客户端访问认证文件
postgresql.auto.conf	保存ALTER SYSTEM修改后的参数 不要手动修改它 优先级较高
postgresql.conf	主要配置文件
recovery.conf	基于wal日志恢复的配置文件
pg_ident.conf	配置哪些操作系统用户可以映射为数据库用户

客户端认证配置（pg_hba.conf）

- 配置文件pg_hba.conf作用
  - 哪些主机可以连接数据库实例
  - 哪个数据库用户可以使用它
  - 允许这个用户使用哪些数据库
  - 客户端使用什么连接方式和认证方式

pg_hba.conf-格式

# TYPE	DATABASE	USER	ADDRESS	METHOD
local	database	user	auth-method	[auth-options]
host	database	user	address	auth-method [auth-options]
hostssl	database	user	address	auth-method [auth-options]
hostnossl	database	user	address	auth-method [auth-options]
host	database	user	IP-address IP-mask	auth-method [auth-options]
hostssl	database	user	IP-address IP-mask	auth-method [auth-options]
hostnossl	database	user	IP-address IP-mask	auth-method [auth-options]

pg_hba.conf-连接方式

- local匹配使用Unix域套接字的连接
  - 如果没有TYPE为local的条目则不允许通过Unix域套接字连接
- host匹配使用 TCP/IP建立的连接，同时匹配SSL和非SSL连接
  - 缺省安装只监听本地环回地址localhost的连接，不允许使用TCP/IP远程连接
  - 启用远程连接需要修改postgresql.conf中的listen_addresses参数
- hostssl匹配必须是使用SSL的TCP/IP连接
  - 客户端和服务器端都安装OpenSSL
  - 编译PostgreSQL的时候指定configure参数--with-openssl打开SSL支持
  - 在postgresql.conf中配置ssl = on
- hostnossl只匹配使用非SSL的TCP/IP连接

pg_hba.conf-认证方式

- trust
  - 无条件地允许连接。
  - 允许任何可以与PostgreSQL数据库服务器连接的用户身份登入
  - 不需要口令或者其他任何认证。
- reject
  - 无条件拒绝连接。常用于从一个组中“过滤出”特定主机
  - 例如一个reject行可以阻塞特定的主机连接,而后面一行允许特定网络中的其余主机进行连接
- md5和password口令认证
  - md5认证方式为双重md5加密，password指明文密码
  - 不能在非信任网络使用password方式。
- peer
  - 从操作系统获得客户端的操作系统用户,并且检查它是否匹配被请求的数据库用户名
  - 只对本地连接可用
- 其它认证方式
  - https://www.postgresql.org/docs/current/static/auth-methods.html

pg_hba.conf-配置示例

vi $PGDATA/pg_hba.conf

```
# TYPE  DATABASE USER    ADDRESS  METHOD
  local     all   all                          trust    # 服务端本地用户可信登录
  host      replication replica  0.0.0.0/0     md5    # 流复制用户密码验证登录
  host      all postgres         0.0.0.0/0     reject   # 拒绝超级用户从网络登录
  host      all all              0.0.0.0/0     md5      # 其它用户密码验证登陆（不安全）
```

local 备注:
Linux下进程通讯方式有很多,比较典型的有套接字,平时比较常用的套接字是基于TCP/IP协议的,适用于两台不同主机上两个进程间通信, 通信之前需要指定IP地址.
同一台主机上两个进程间通信用套接字,还需要指定ip地址,有点过于繁琐. 这个时候就需要用到UNIX Domain Socket, 简称UDS,

服务主配置（postgresql.conf）

配置文件结构

#注释

key = value

支持的参数值类型

布尔、整数、浮点数、字符串、枚举

include指令（允许嵌套）

备注：
如果配置值中包含数字，则需要用单引号括起。
如果参数值本身包含单引号，我们可以写两个单引号(推荐方法)或用反斜扛包围

配置参数项-连接

listen_addresses = '*'　　#（关联配置文件pg_hba.conf）

#指定服务器在哪些 TCP/IP 地址上监听客户端连接。
值的形式是一个逗号分隔的主机名和/或数字 IP 地址列表。

特殊项*对应监听所有可用 IP 接口

0.0.0.0允许监听所有 IPv4 地址
并且::允许监听所有 IPv6 地址
如果列表为空，服务器将根本不会监听任何 IP 接口，在这种情况中只能使用 Unix 域套接字来连接它。
默认值是localhost，它只允许建立本地 TCP/IP "环回"连接。
这能帮助在不安全网络接口上阻止重复的恶意连接请求。
这个参数只能在服务器启动时设置。

配置参数项-连接

port = 5866

#服务器监听的 TCP 端口；默认是 5866 。
请注意服务器会同一个端口号监听所有的 IP 地址。
这个参数只能在服务器启动时设置。

max_connections = 100

#决定数据库的最大并发连接数。
默认值通常是 100 个连接，但是如果内核设置不支持（initdb时决定），可能会比这个 数少。
这个参数只能在服务器启动时设置。（cpu 1核 50个链接）

配置参数项-内存

shared_buffers

它表示数据缓冲区中的数据块的个数，每个数据块的大小是8KB。
数据缓冲区位于数据库的共享内存中，它越大越好，不能小于128KB。
这个参数只有在启动数据库时，才能被设置。
默认值是128MB。
推荐值：1/4 主机物理内存

wal_buffers

用于还未写入磁盘的 WAL 数据的共享内存量。
默认值 -1 表示将该参数值设置为 shared_buffers 的 1/32 的大小 （大约 3%），但是不小于64kB 也不大于一个WAL段的大小（通常为 16MB）。
如果自动的选择太大或太小可以手工设置该值，但是任何小于 32kB 的正值都将被当作 32kB。
这个参数只能在服务器启动时设置。
事务日志缓冲区位于数据库的共享内存中。
推荐值：min(2047MB, shared_buffers/32) = 512MB

配置参数项-内存

work_mem

指定在写到临时磁盘文件之前用于内部排序操作和哈希表的内存量。
ORDER BY， DISTINCT 和合并连接（ merge joins） 都会用到排序操作。
默认值为 4 兆字节（ 4MB）。
推荐值：work_mem = (输入内存数量- shared_buffers)/(连接数 * 3） * 1024 (单位是 KB);

maintenance_work_mem

它决定数据库的维护操作使用的内存空间的大小。
数据库的维护操作包括VACUUM、CREATE INDEX和ALTER TABLE ADD FOREIGN KEY等操作。
值如果比较大，通常可以缩短VACUUM数据库和从dump文件中恢复数据库需要的时间。
maintenance_work_mem存放在每个数据库进程的私有内存中，而不是存放在数据库的共享内存中。这个参数可以在任何时候被设置。

配置参数项-优化建议

random_page_cost = 2.5

规划器对一次非顺序获取磁盘页面的代价估计。 默认值是 4.0。 高端存储或者ssd可以适当调小该参数。

autovacuum_max_workers = 10

指定能同时运行的autovacuum进程的最大数量， 适当调大， 避免vacuum不及时导致表膨胀

checkpoint_completion_target = 0.7

增加checkpoint_completion_target来降低检查点的I/O负载， 默认0.5

archive_timeout =1800

强制服务器来周期性地切换到一个新的 WAL 段文件

archive_command = 'test ! -f /paic/dba/pgbackup/${PGNAME}/archlog/%f && pxz -2 < %p >/paic/dba/pgbackup/${PGNAME}/archlog/%f'

pxz压缩归档日志， 64M的归档， 压缩时间可以在0.5s内， 压缩比一般可在1:3左右

配置参数级别

- 系统级别 /集群服务级别/实例级别（全局）
- 用户/角色级别
- 用户/角色 + 数据库级别
- 会话级别

参数设置级别	参数存储位置
cluster	postgresql.conf or postgresql.auto.conf
db	pg_db_role_setting
role	pg_db_role_setting
db 和 role的组合	pg_db_role_setting

配置修改-全局

- 以下几种方式可实现全局配置
  - 通过linux命令(vim,echo,sed)修改配置文件
  - 启动时设置（不推荐，除非进入单用户模式）
  - psql -c configparameter=newvalue
  - 通过ALTER SYSTEM命令修改全局配置（针对postgresql.auto.conf）

```
\h alter system
命令:       ALTER SYSTEM
描述:       更改服务器的配置参数

语法:
ALTER SYSTEM SET 配置参数 { TO | = } { 值 | '值' | DEFAULT }
ALTER SYSTEM RESET 配置参数
ALTER SYSTEM RESET ALL
```

配置修改-非全局

配置生效级别	修改方式
Database	ALTER DATABASE name SET configparameter { TO | = } { value | DEFAULT } ALTER DATABASE name RESET configuration
Session	通过SET命令设置当前Session的配置 SET configparameter { TO | = } { value | 'value' | DEFAULT } SET configparameter TO DEFAULT;
	更新pg_settings视图 UPDATE pg_settings SET setting = new_value WHERE name = 'configparameter';
	使用set_config函数更新会话配置 SELECT set_config('configparameter',new_value,false);
user/Role	ALTER ROLE name [IN DATABASE database_name] SET configparameter { TO | = } { value | DEFAULT } 重置/取消这些参数的设置 ALTER ROLE name [IN DATABASE database_name] RESET configparameter ALTER ROLE name [IN DATABASE database_name] RESET ALL;

配置生效说明

select name,distinct(context) from pg_settings;

Context（上下文）	生效方式
sihup	给服务器发送 HUP信号会使服务器重新加载 postgresql.conf配置， 可以立即生效
postmaster	只有服务重启才能生效
internal	编译期间的设置，只有重新编译才能生效。
backend	与sighup类似，但是不影响正在运行的会话，只在新会话中生效
superuser	使用superuser(如postgres)才能更改， 不用重新加载所有配置即可生效
user	单个会话用户可以在任意时间做修改，只会影响该会话

配置生效方法

- 使配置生效的几种方法

1）用超级用户运行

SELECT pg_reload_conf();

2）使用pg_ctl命令触发SIGHUP信号

pg_ctl reload

3）用UNIX的kill手动发起HUP信号

ps -ef|grep -i postmaster|grep -v grep|xargs kill -HUP

4）重启数据库服务

pg_ctl restart

备注：
主服务器进程每次收到 SIGHUP 信号后都会重新读取这个配置文件。
同时主服务器进程也将这个信号广播给所有正在运行的子服务器进程，这样现有会话也能得到新值。

配置查看方法

- 通过 SHOW 命令查看特定系统设置
  - SHOW 命令的输出结果会自动根据数值大小选择合适的单位的所有设置
  - SHOW ALL 命令，其输出结果会针对每个设置选用合适的单位

```
--配置查看示例
show all;                    #查看所有数据库参数的值
show shared_buffers;   #查看某个参数的当前值(可查看当前会话值)
select current_setting('shared_buffers');
```

配置查看方法

- 查询 pg_settings 视图可以很方便地检查当前设置
  - 它本质上是SHOW和SET命令的可替换接口
  - 它还提供了SHOW不能提供的关于每一个参数的一些实现，例如最大值和最小值

```
--配置查看参考SELECT name, setting FROM pg_settings WHERE category = 'File Locations';
select name,context,unit,boot_val,setting,reset_val from pg_settings where name
in('listen_addresses','max_connections',
'shared_buffers','effective_cache_size','work_mem','maintenance_work_mem')
order by context, name;
SELECT name,setting FROM pg_settings where name ~ ‘xxx’ ;
SELECT current_setting(name);
```

配置查看方法

pg_settings参数查看示例：

```
postgres=# select name,context,unit,boot_val,reset_val from pg_settings where name in('listen_addresses','max_c
onnections', 'shared_buffers','effective_cache_size','work_mem','maintenance_work_mem')
postgres-# ORDER BY context, name;
         name          |  context   | unit | boot_val  | reset_val
-----------------------+------------+------+-----------+-----------
 listen_addresses      | postmaster |      | localhost | *
 max_connections       | postmaster |      | 100       | 100
 shared_buffers        | postmaster | 8kB  | 1024      | 16384
 effective_cache_size  | user       | 8kB  | 524288    | 524288
 maintenance_work_mem  | user       | kB   | 65536     | 65536
 work_mem              | user       | kB   | 4096      | 4096
(6 rows)
```

备注：
setting 是指当前设置；boot_val 是指默认设置；reset_val 是指重新启动服务器或重新加载设置之后的新设置。
在 postgresql.conf 中修改了设置后，一定要记得查看一下 setting 和 reset_val 并确保二者是一致的，否则说明设置并未生效，需要重新启动服务器或者重新加载设置。

配置示例-用户角色级修改参数

```
--查询某参数在某用户级别的设置
create user test password 'test';
alter role test set log_min_duration_statement = 100;

--方法一：查询pg_user表
select * from pg_user where usename='test';

--方法二：查询pg_db_role_setting表
select * from pg_db_role_setting where setrole in (select usesysid from pg_user where usename in ('test'))
 order by setrole,setdatabase;

--参数在某用户下针对数据库级别的设置
alter role test in database postgres set client_min_messages='warning';

select * from pg_db_role_setting where setrole in (select usesysid from pg_user where usename in ('test'))
order by setrole,setdatabase;
```

配置示例-postgresql.auto.conf

```
#运行日志相关配置变更
alter system set logging_collector = on;
alter system set log_destination = 'csvlog';
alter system set log_directory = '/data/pg968/data/pg_log';
alter system set log_filename ='postgresql-%Y-%m-%d_%H%M%S.log';
alter system set log_rotation_age = '1d';                    #每天生成一个新的日志文件
alter system set log_rotation_size = 0;                      #不限制单个日志文件大小
alter system set log_truncate_on_rotation = on;              #覆盖同名文件
                                                             #只保留7天日志，循环覆盖

alter system set log_hostname = on;
alter system set log_line_prefix = '%m';          #控制每条日志信息的前缀格式，默认值是空串
alter system set log_statement = 'ddl';
```

配置示例-postgresql.auto.conf

```
#WAL和归档参数变更
alter system set wal_level = archive;
alter system set archive_mode = on;
alter system set archive_directory = '/ssd/archive';
```

建议：此处/ssd/archive建议将归档日志放到数据库的定时备份的路径内，便于数据库异机恢复时最大可能的恢复数据

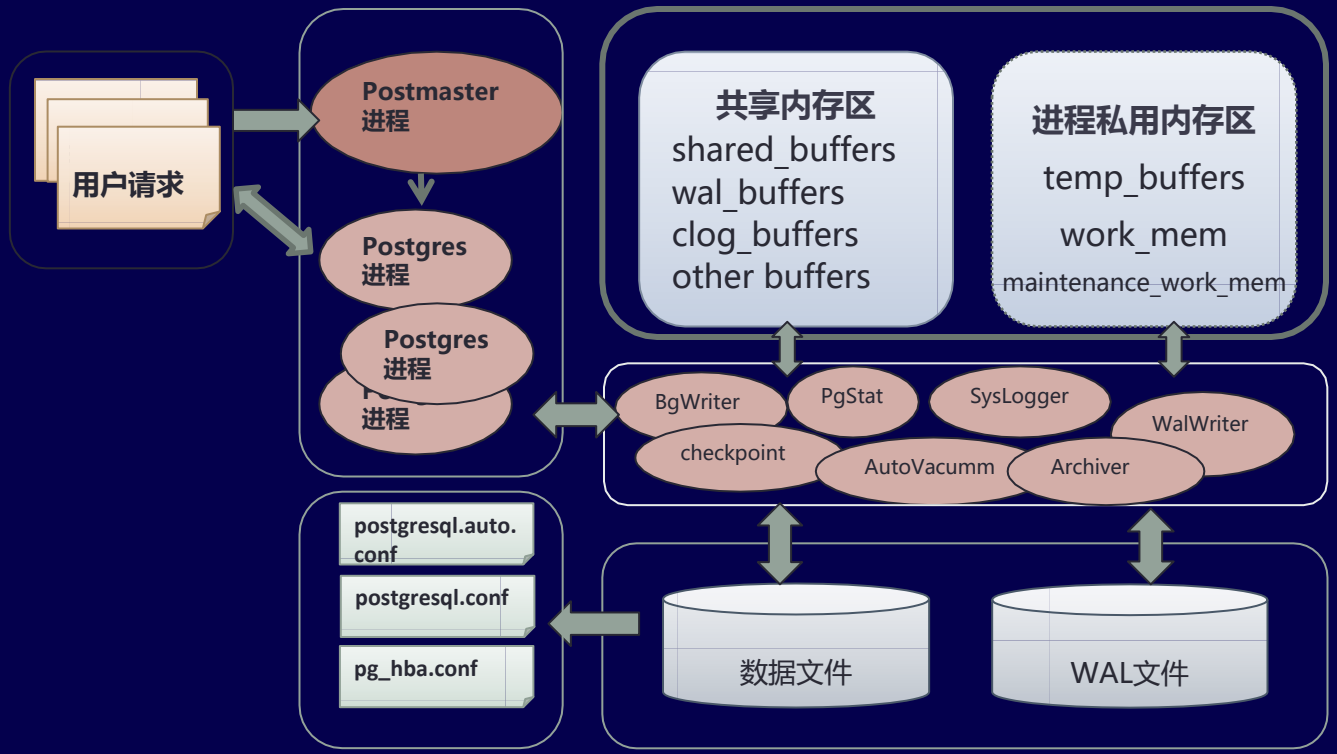
```
#参数修改完成后重新启动数据库
pg_ctl -m fast stop
pg_ctl start
```

3.PostgreSQL体系结构

概述

- PostgreSQL使用一种客户端/服务器，进程架构模型
- 整体上包括以下几个部分：
  - 内存区域
  - 配置文件
  - 服务进程
  - 磁盘存储

整体结构


用户请求
Postmaster 进程
Postgres 进程
Postgres 进程
进程
共享内存区
shared_buffers
wal_buffers
clog_buffers
other buffers
进程私用内存区
temp_buffers
work_mem
maintenance_work_mem
BgWriter
PgStat
SysLogger
WalWriter
checkpoint
AutoVacumm
Archiver
postgresql.auto.conf
postgresql.conf
pg_hba.conf
数据文件
WAL文件

共享内存区

shared_buffers ---共享缓冲区

它表示数据缓冲区中的数据块的个数，每个数据块的大小是8KB。

数据缓冲区位于数据库的共享内存中，它越大越好，不能小于128KB。

这个参数只有在启动数据库时，才能被设置。

默认值是128MB。

推荐值：1/4 主机物理内存

wal_buffers --- 日志缓存区的大小

可以降低IO，如果遇上比较多的并发短事务，应该和commit_delay一起用

存放WAL数据的内存空间大小，系统默认值是64K

进程私有内存区

temp_buffers---临时缓冲区

用于存放数据库会话访问临时表数据，系统默认值为8M。

可以在单独的session中对该参数进行设置，尤其是需要访问比较大的临时表时，将会有显著的性能提升

work_mem --- 工作内存或者操作内存。

其负责内部的sort和hash操作，合适的work_mem大小能够保证这些操作在内存中进行。

maintenance_work_mem ---维护工作内存

主要是针对数据库的维护操作或者语句。

主要针对VACUUM，CREATE INDEX，ALTER TABLE ADD FOREIGN KEY等操作。

在对整个数据库进行VACUUM或者较大的index进行重建时，适当的调整该参数非常必要

postresql文档提示在启用了autoacuum功能的情况下，该参数不能配置的过大

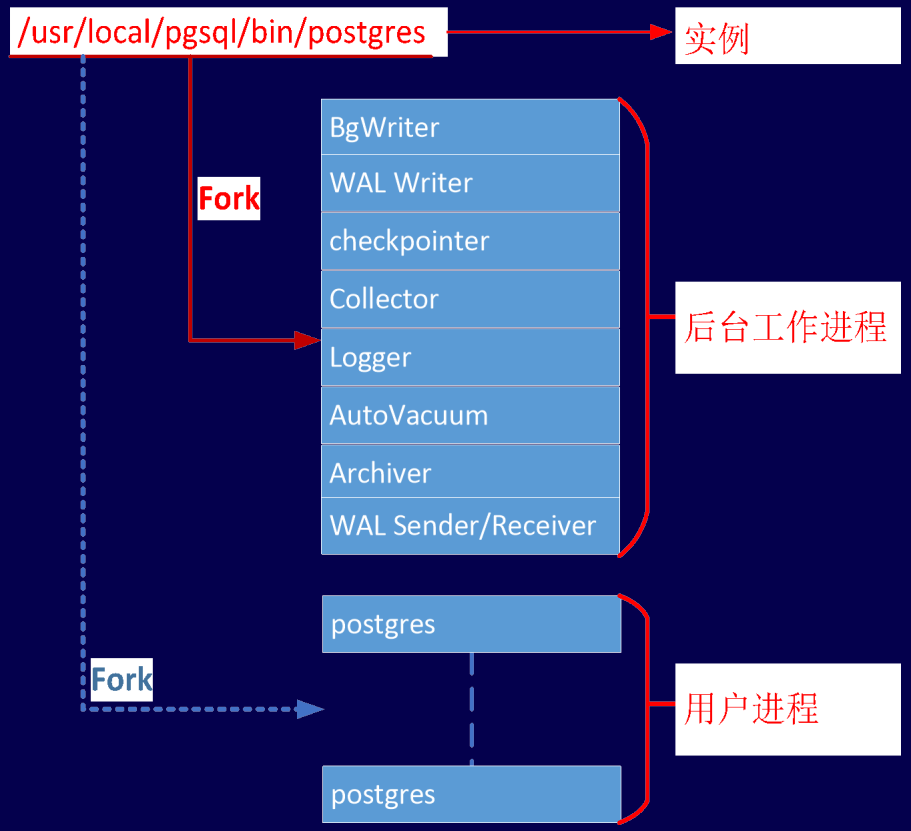
主要配置文件

存储着和连接相关的权限配置，文件存放在Data目录下。文件中每一行为一条配置，每条配置中包括连接类型、数据库、用户、客户端地址和认证方式几个字段

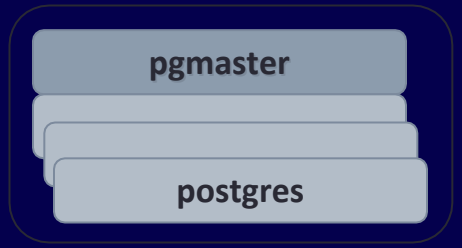
Pgsql的配置参数是在这个文件中集中管理的，文件存放在Data目录下。
每个参数配置项的格式都是“参数名 = 参数值”

它和postgresql.conf 的格式相同，但不允许手动编辑。这个文件保存了通过ALTER SYSTEM 命令设置的参数。在读取postgresql.conf时自动读取该文件，并且它的设置以相同的方式生效。
设置会覆盖postgresql.conf 中的设置。

进程结构


/usr/local/pgsql/bin/postgres
实例
Fork
BgWriter
WAL Writer
checkpointer
Collector
Logger
AutoVacuum
Archiver
WAL Sender/Receiver
后台工作进程
Fork
postgres
postgres
用户进程

主要进程



Postmaster进程
PG数据库的总控制进程，负责启动和关闭数据库实例，是一个指向postgres命令的链接，算是第一个postgres进程。
Postmaster的主进程，还会fork出一些辅助子进程。

辅助子进程

- BGWriter
- Wal Writer
- AutoVAcuum
- archiver
- stats collector
- checkpoint

BgWriter（后台写）进程

WaLWriter（预写式日志）进程

AutoVAcuum（系统自动清理）

SysLogger（系统日志）进程

PgArch（归档）进程

PgStat（统计数据收集）进程

CheckPoint（检查点）进程

进程结构

```
[postgres@hgdb01 13295]$ ps -ef|grep -i postgres
postgres   1549      1  0 May31 ?        00:00:00 /usr/local/pgsql957/bin/postmaster -D /pgdata/data
postgres   1559   1549  0 May31 ?        00:00:00 postgres: logger process
postgres   1561   1549  0 May31 ?        00:00:00 postgres: checkpointer process
postgres   1562   1549  0 May31 ?        00:00:00 postgres: writer process
postgres   1563   1549  0 May31 ?        00:00:00 postgres: wal writer process
postgres   1564   1549  0 May31 ?        00:00:00 postgres: autovacuum launcher process
postgres   1565   1549  0 May31 ?        00:00:00 postgres: archiver process
postgres   1566   1549  0 May31 ?        00:00:01 postgres: stats collector process
root      12034  11736  0 10:19 pts/0    00:00:00 su - postgres
postgres  12035  12034  0 10:19 pts/0    00:00:00 -bash
root      12521  12482  0 10:54 pts/1    00:00:00 su - postgres
postgres  12522  12521  0 10:54 pts/1    00:00:00 -bash
postgres  14877  12035  0 14:39 pts/0    00:00:00 ps -ef
postgres  14878  12035  0 14:39 pts/0    00:00:00 grep --color=auto -i postgres
[postgres@hgdb01 13295]$ pstree -p 1549
postmaster(1549)─┬─postmaster(1559)
                 ├─postmaster(1561)
                 ├─postmaster(1562)
                 ├─postmaster(1563)
                 ├─postmaster(1564)
                 ├─postmaster(1565)
                 └─postmaster(1566)
[postgres@hgdb01 13295]$
```

进程通信-建立会话

建立会话的过程

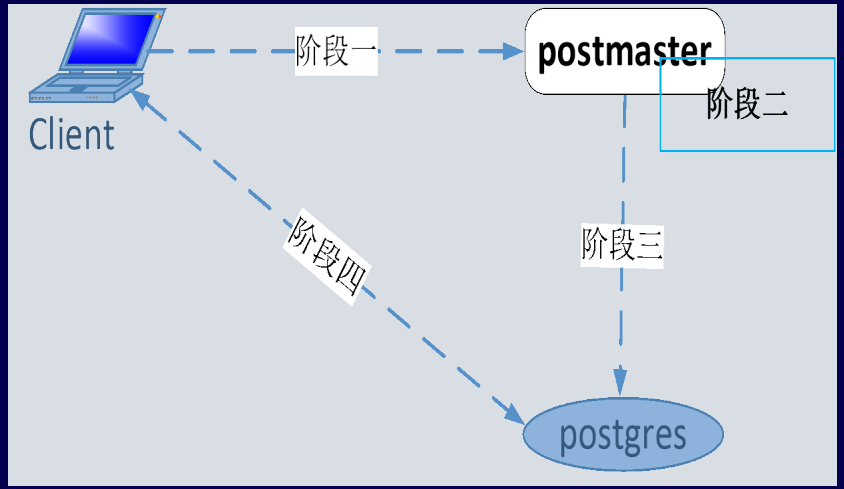
阶段一：客户端发起请求

阶段二：该阶段由主服务postmaster进程负责

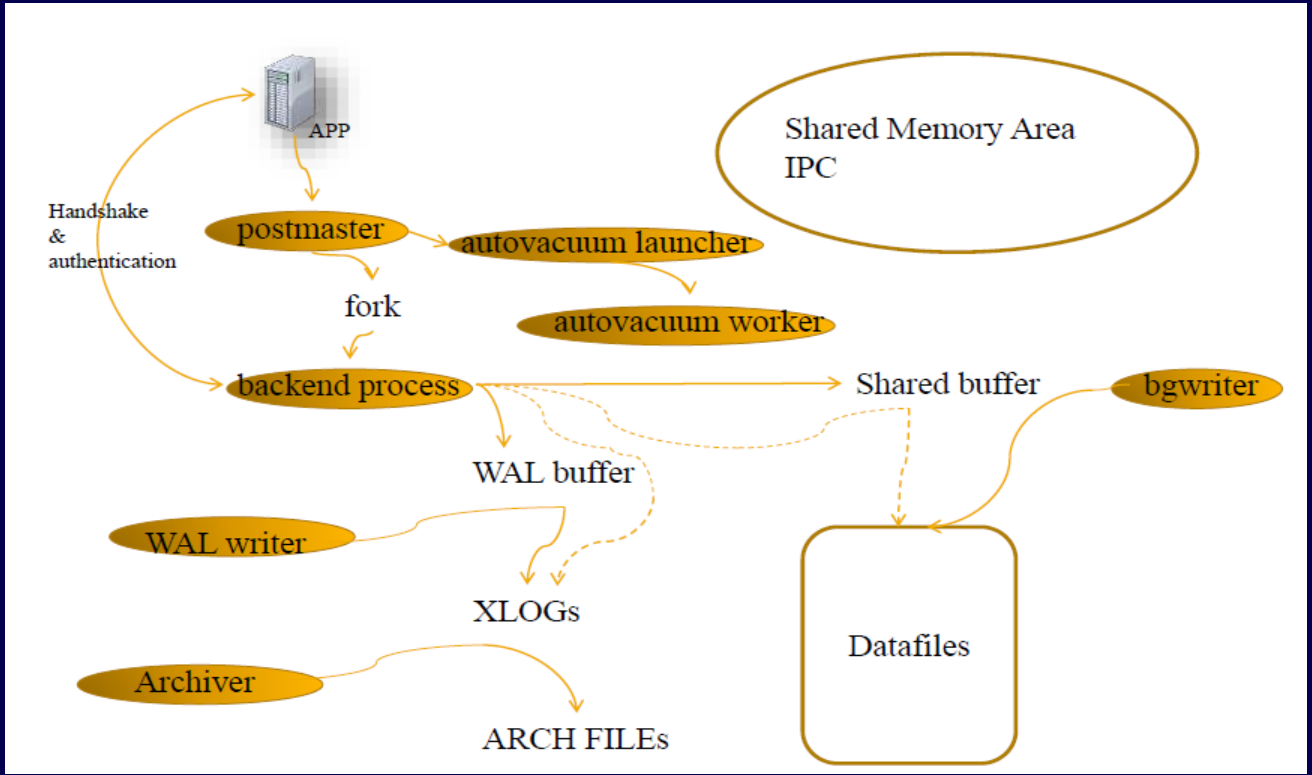
- ✓ 服务器是否接受客户端的host通信认证
- ✓ 服务器对客户端进行身份鉴别

阶段三：阶段二通过之后，主服务进程为该客户端单独fork一个客户端工作进程postgres

阶段四：客户端与postgres进程建立通信连接，由postgres进程负责后续所有的客户端请求操作，直至客户端退出后，该postgres进程消失

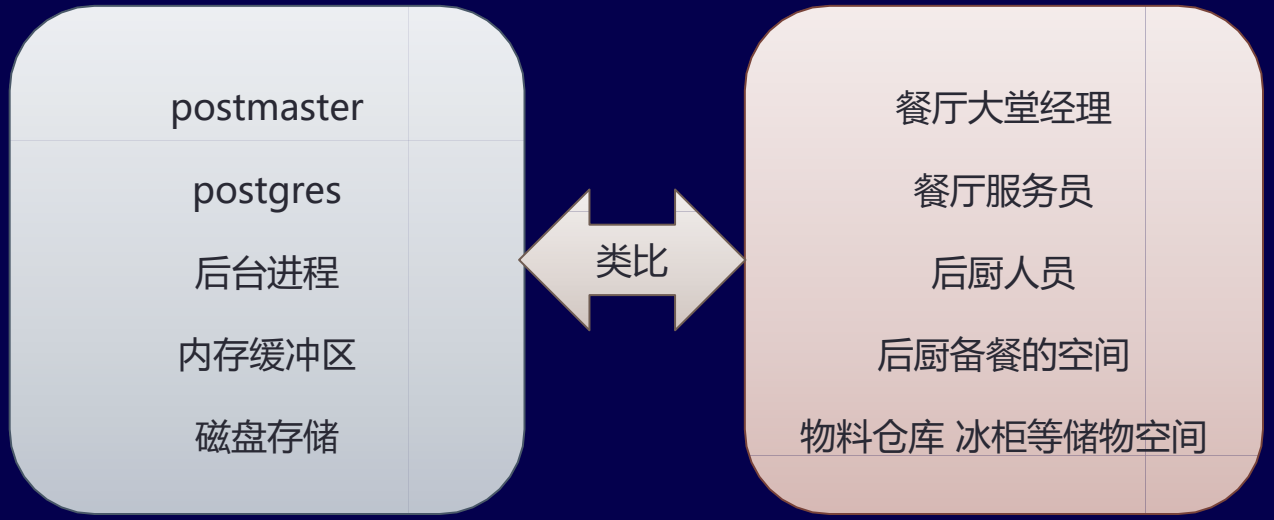


进程通信-建立会话


APP
Shared Memory Area
IPC
Handshake
&
authentication
postmaster
autovacuum launcher
fork
autovacuum worker
backend process
Shared buffer
bgwriter
WAL buffer
WAL writer
XLOGs
Datafiles
Archiver
ARCH FILEs

客户端请求理解

类比顾客去餐厅用餐



物理结构

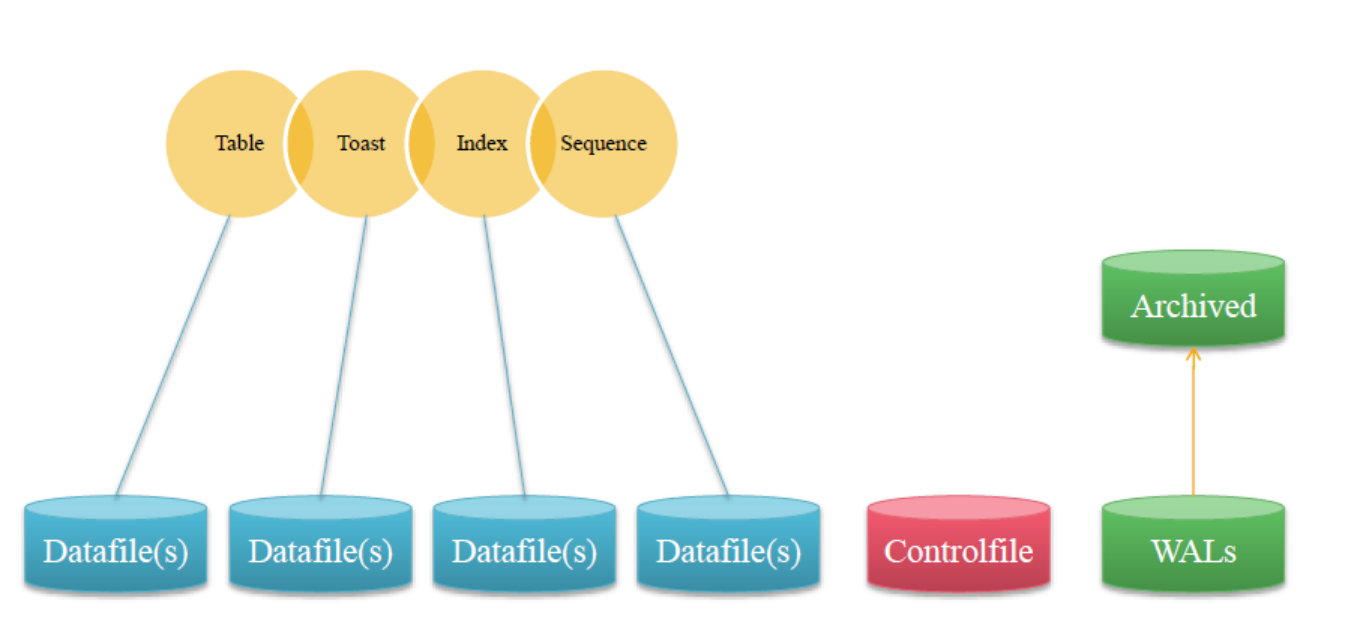


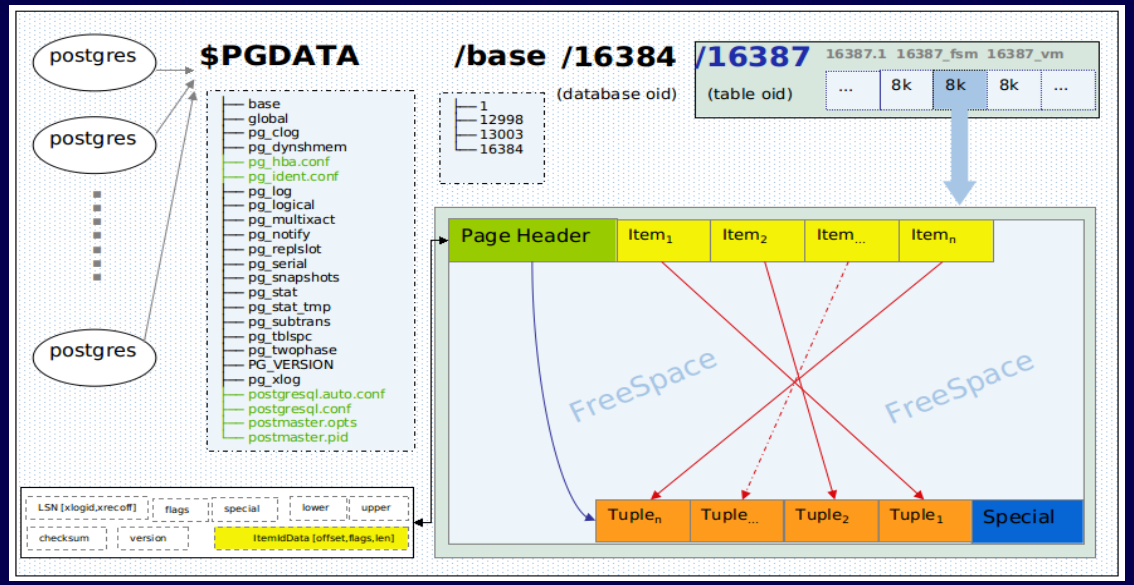
Table
Toast
Index
Sequence
Archived
Datafile(s)
Datafile(s)
Datafile(s)
Datafile(s)
Controlfile
WALs

物理结构-数据存储方式

- 目录结构
  - 默认表空间的数据文件存放在base目录
  - 在pg_xlog保存WAL日志，只能通过软链改变它的位置
- 段(Segments)
  - 每一个表和索引都存放到单独的数据文件中
  - 文件名是表或索引的文件结点(filenode)编号
  - 如果表或索引超过 1 GB 就会被分割为多个段
  - 第一个段以文件结点(filenode)编号命名，第二个以及之后的段以filenode.1, filenode.2形式命名
- Page(Block)和Buffer
  - 在磁盘中称为page,内存中称为buffer
  - 默认为8k,可以在编译时指定block_size参数改变大小
- Relation：表示表或索引
- Tuple（row）表中的行

物理结构-目录结构

- 查看命令：
  show data_directory;
  select name,setting from pg_settings where category='File Locations';



物理结构-目录含义

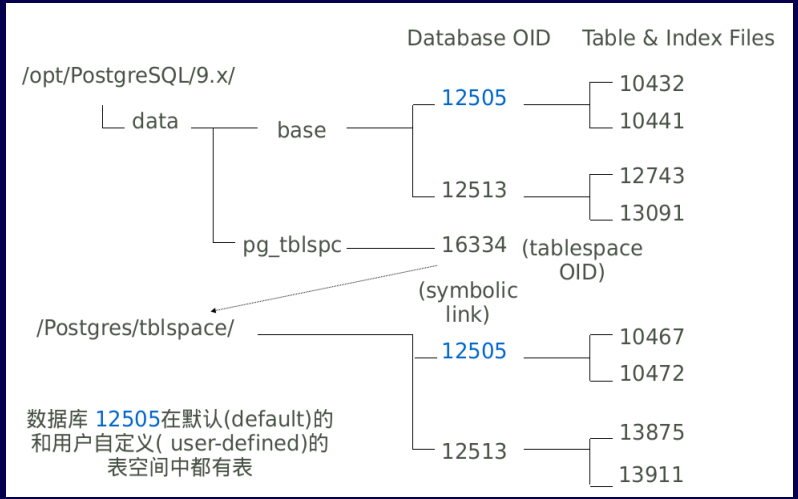
对象	类型	说 明
base	目录	存储数据库及数据文件
global	目录	存储cluster级的系统表
pg_clog	目录	存储事务提交状态
pg_commit_ts	目录	存储事务提交时间
pg_dynshmem	目录	存储动态共享内存子系统的相关文件
pg_log	目录	缺省的数据库日志存储目录
pg_logical	目录	存储逻辑解码的状态
pg_multixact	目录	存储并发事务的状态（用于共享行锁）
pg_notify	目录	存储listen/notify的状态
pg_replslot	目录	存储复制槽/位数据
pg_serial	目录	存储关于序列化事务的信息
pg_snapshots	目录	存储导出快照

物理结构-目录含义

对象	类型	说　明
pg_stat	目录	存储用于统计子系统的持久性文件
pg_stat_tmp	目录	存储用于统计子系统的临时文件
pg_subtrans	目录	存储子事务的状态
pg_tblspc	目录	存储指向实际表空间的软连接
pg_twophase	目录	存储关于预提交事务的状态
pg_xlog	目录	存储联机事务日志
PG_VERSION	文件	该cluster所属的数据库版本文件
postgresql.conf	文件	数据库参数配置文件
postgresql.auto.conf	文件	数据库参数配置文件
pg_hba.conf	文件	访问配置文件，侧重访问方法
pg_ident.conf	文件	访问配置文件，侧重身份鉴别、用户映射
postmaster.opts	文件	该cluster最后一次启动时使用的命令行选项
postmaster.pid	文件	该cluster启动后，生成的锁文件
recovery.conf	文件	恢复或复制中需要的配置文件

物理结构-目录关联

```
/opt/PostgreSQL/9.x
    |______bin
    |______data
        |______base
        |______global
        |______pg_clog
        |______pg_xlog
        |______pg_tblspc
        |______postgresql.conf  pg_hba.conf
    |______doc
    |______include
    |______installer
    |______lib
    |______share
```



物理结构-相关概念

- **OID**
  - 对象标识符类型Object Identifier Types，内部使用，并作为系统表的主键
  - 别名类型对查找一个对象的OID非常方便
  - 可以在用户定义的表中使用，不推荐使用
  - oid不会添加到 用户自己创建的表里，除非指定 WITH OIDS 或者 default_with_oids 打开
  - oid2name可以获取数据库、对象的OID
- **relfilenode**
  - 标识对象物理位置的数字标号，会随数据存放的位置变化而变化
  - 函数pg_relation_filenode() 可以获得对象的relfilenode

物理结构-相关概念

- **OID 与relfilenode的关系**
  - **oid - 身份证号**
  - **refilenode - 户口本上户号**
  - 只要人的住址变了，**户号**就会变
  - truncate、vacuum full、删除重建表操作都会导致refilenode变化

物理结构-相关概念

```
postgres=# create table a(a int);
CREATE TABLE

postgres=# select oid,relfilenode from pg_class where relname='a';
  oid   | relfilenode
--------+-------------
 123128 |      123128
(1 row)

postgres=# select pg_relation_filenode('a');
 pg_relation_filenode
----------------------
               123128
(1 row)
```

物理结构-相关概念

◆ 空闲空间映射(FSM)

- 每一个表和索引（除了哈希索引）都有一个空闲空间映射（FSM）来保持对关系中可用空间的跟踪
- 伴随主关系数据被存储在一个独立的关系分支中，以关系的文件节点号加上一个_fsm后缀命名
- FSM文件是执行VACUUM操作时，或者是为了插入行而第一次查询FSM文件时才会创建
- PostgreSQL使用了树形结构组织FSM文件
- FSM可以在数据插入时快速找到满足大小要求的空闲空间，从而复用空闲空间

物理结构-相关概念

◆ 可见性映射(VM)

- 为了能加快VACUUM清理的速度和降低对系统I/O性能的影响，V8.4版本以后为每个数据文件加了一个后缀为“__vm “的文件
- 每一个表都有一个可见性映射（VM）用来跟踪哪些页面只包含已知对所有活动事务可见的元组，它也跟踪哪些页面只包含未被冻结的元组。
- 它随着主关系数据被存储在一个独立的关系分支中，以该关系的文件节点号加上一个_vm后缀命名
- 有了这个文件后，通过VACUUM命令扫描这个 文件时，如果发现VM文件中这个数据块上的位表示该数据块没有需要清理的行，则会跳过对这个数据块的扫描，从而加快VACUUM清理的速度。
- pg_visibility模块可以被用来检查存储在可见性映射中的信息

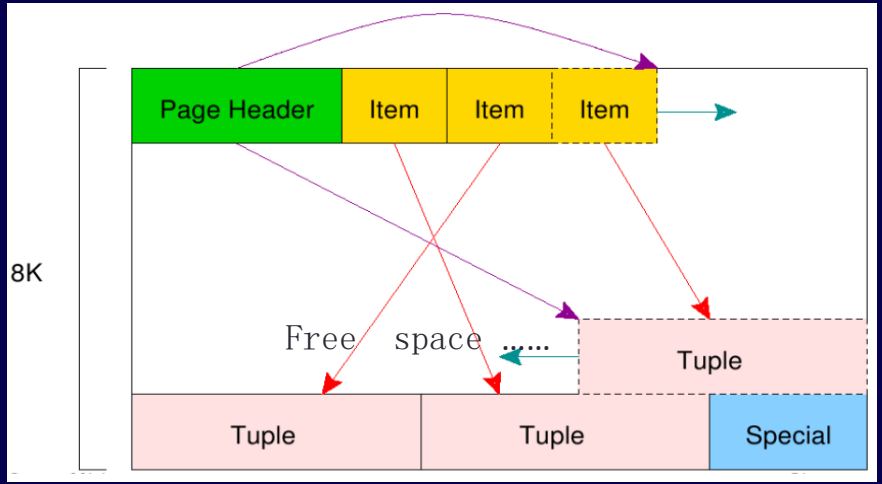
物理结构-相关概念

```
postgres=#  select pg_relation_filepath('a'::regclass);
 pg_relation_filepath
----------------------
 base/13295/123128
(1 row)

postgres=# \! ls -l $PGDATA/base/13295/123128*
-rw-------. 1 postgres postgres  8192 May 29 15:12 /pgdata/data/base/13295/123128
-rw-------. 1 postgres postgres 16384 May 29 14:41 /pgdata/data/base/13295/123128_fsm
-rw-------. 1 postgres postgres     0 May 29 14:41 /pgdata/data/base/13295/123128_vm
```

物理结构-页结构

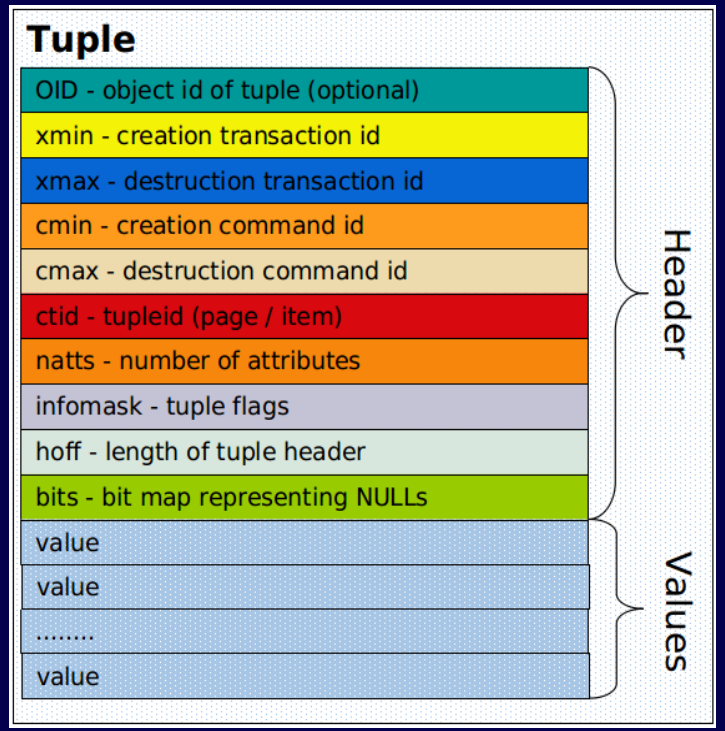
- Page Header
  - Page的基本信息
  - 指向空闲空间(free space)
  - 长为 24 bytes
- ItemIdData（Row/Index Pointers）
  - 一个记录偏移量/长度(offset/length)的数组，
  - 指向实际的记录(rows/index entries)
  - 每一个 item 4字节
- Free Space
  - 未分配的空间
  - 新指针(pointers)从这个区域的开头开始分配
  - 新的记录(rows/index entries)从结尾开始分配
- Items（Row/Index Entry）
  - 实际的记录本身
- Special
  - 不同的索引访问方式相关的数据
  - 在普通表中为空



物理结构-PageHeaderData布局

域	类型	长度	描述
pd_lsn	PageXLogRecPtr	8 bytes	LSN: 最后修改这个页面的xlog记录最后一个字节后面的第一个字节
pd_checksum	uint16	2 bytes	页面校验码
pd_flags	uint16	2 bytes	标志位
pd_lower	LocationIndex	2 bytes	到空闲空间开头的偏移量
pd_upper	LocationIndex	2 bytes	到空闲空间结尾的偏移量
pd_special	LocationIndex	2 bytes	到特殊空间开头的偏移量
pd_pagesize_version	uint16	2 bytes	页面大小和布局版本号信息
pd_prune_xid	TransactionId	4 bytes	页面上最老未删除XMAX，如果没有则为0

物理结构-TUPLE数据结构

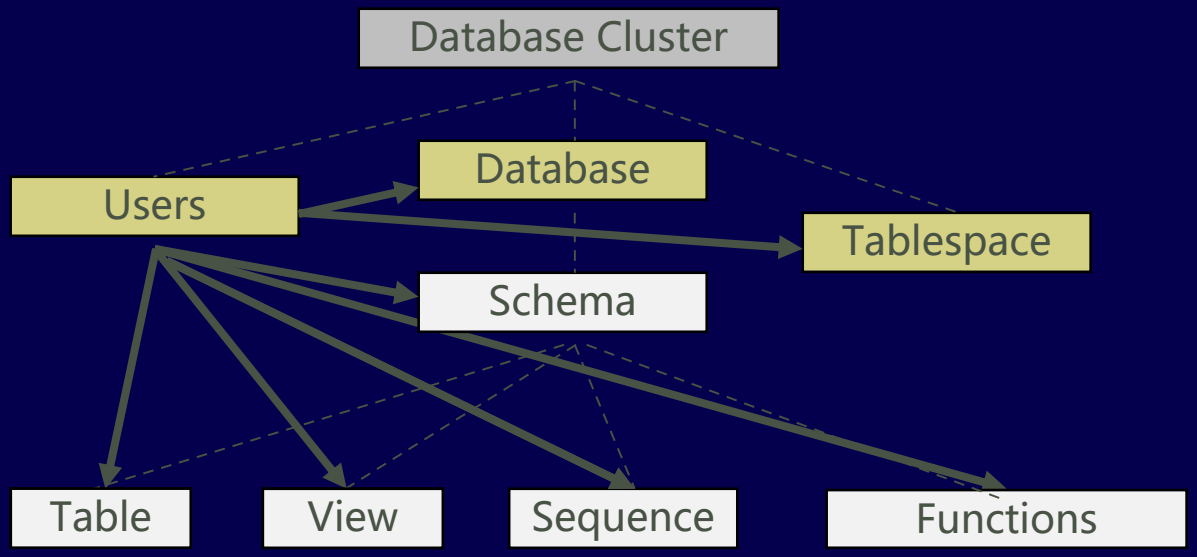



```
postgres=# \d a
      Table "public.a"
 Column |  Type   | Modifiers
--------+---------+-----------
 a      | integer |
postgres=# select txid_current();
 txid_current
--------------
         8473
(1 row)
postgres=# insert into a values(1);
INSERT 0 1
postgres=# select xmin,xmax,ctid,a from a;
 xmin | xmax | ctid  | a
------+------+-------+---
 8473 |    0 | (0,1) | 1
(1 row)
```

物理结构-TUPLE头部数据结构

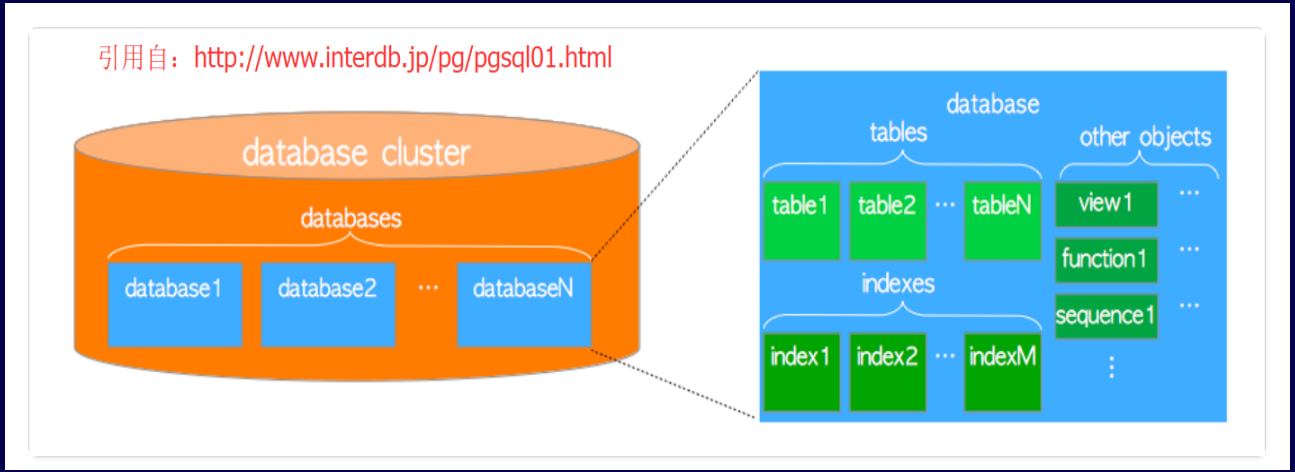
域	类型	长度	描述
t_xmin	TransactionId	4 bytes	插入XID标志
t_xmax	TransactionId	4 bytes	删除XID标志
t_cid	CommandId	4 bytes	插入和/或删除CID标志（覆盖t_xvac）
t_xvac	TransactionId	4 bytes	VACUUM操作移动一个行版本的XID
t_ctid	ItemPointerData	6 bytes	当前版本的TID或者指向更新的行版本
t_infomask2	uint16	2 bytes	一些属性，加上多个标志位
t_infomask	uint16	2 bytes	多个标志位
t_hoff	uint8	1 byte	到用户数据的偏移量

逻辑结构图


Database Cluster
Database
Users
Tablespace
Schema
Table
View
Sequence
Functions

逻辑结构

- 一个PG Cluster是在同一个os上的一个或者多个PG database的集合
- 在一个操作系统（OS）中，可以有一个或者多个PG Cluster
- 不同的PG Cluster是以port （端口号）作为区分的
- 这两个PG Cluster的软件版本甚至可以是不同的软件版本
- 假设同一个OS上有两个PG Cluster，每个PG Cluster中均可以有一个名为database1的PG database



逻辑结构-主要概念

- Cluster
  - Cluster,物理盘上的数据库存储区，在SQL标准中的术语为catalog cluster在PostgreSQL中称之为database cluster，它在PostgreSQL中从下述两个方面区分：
  - 物理存储的绝对路径　　$PGDATA
  - 各自的端口号　　5432
- database
  - 每个 PostgreSQL 服务可以包含多个独立的 database
- 表空间
  - PostgreSQL中的表空间实际是物理盘上的一个存储目录，它从下述两个方面进行区分：
  - 表空间名
  - 定义表空间时使用的路径（同一个路径，不能同时用于不同的表空间）

CHINA
POSTGRESQL
ASSOCIATION

THANKS

中国PG分会 魏波
boluo_wei@126.com