pgAdmin Documentation

Release 1

LANXE

May 20, 2013

CONTENTS

1 Manual de Administración de PostgreSQL **2**
- 1.1 Tutorial Introductorio . . . 2

2 Generación de Logs **9**
- 2.1 Parámetros . . . 9

3 Controles de Acceso a Objetos de la base de datos **10**
- 3.1 Usuarios, grupos y roles . . . 10
- 3.2 Grant. Otorgar derechos a usuarios. . . . 11
- 3.3 El archivo pg_hba.conf . . . 12
- 3.4 Métodos de autentificación . . . 14

4 Tipos de Datos **15**
- 4.1 Booleano . . . 15
- 4.2 Caracter . . . 15
- 4.3 Numéricos: . . . 16
- 4.4 Tipos de datos temporales . . . 17
- 4.5 Tipos de datos especiales . . . 18
- 4.6 Manipulación de datos . . . 19

5 Tablas **21**
- 5.1 Principales constraints . . . 21

6 Indices and tables **23**

Contents:

CHAPTER

ONE

MANUAL DE ADMINISTRACIÓN DE POSTGRESQL

1.1 Tutorial Introductorio

1.1.1 El proceso de instalación

- Instalamos el repositorio oficial

```
wget http://yum.postgresql.org/9.2/fedora/fedora-18-x86_64/pgdg-fedora92-9.2-6.noarch.rpm
sudo yum localinstall pgdg-fedora92-9.2-6.noarch.rpm
sudo yum check-update
```

- Configuramos los repositorios oficiales de fedora para que dejen de proveer postgres

```
sudo -i
vi /etc/yum.respos.d/fedora.repo
```

y añadimos la siguiente línea en la sección [fedora]

```
exclude=postgres*
```

- Procedemos de igual manera con el archivo /etc/yum.repos.d/fedora-updates.repo

Características de PostgreSQL

1. Manejador de bases de datos objeto-relacional
2. Arquitectura cliente / servidor
3. Transaccional
4. Un proceso por conexión
5. Los procesos pueden estar distribuidos en distintos procesadores
6. **Procesos auxiliares**
    - Escritura en segundo plano
    - Escritor de WAL
    - Autovacuum
    - Collector de estadísticas

    - Collector de la bitácora de eventos

7. El tamaño máximo de una base de datos es ilimitado.
8. Tamaño máximo de una tabla 32TB (almacenadas como múltiples archivos de 1GB).
9. Tamaño máximo de una fila: 400 GB
10. Tamaño máximo de una columna: 1GB
11. Número máximo de índices en una tabla: ilimitado
12. Tamaño máximo de un identificador (nombres de tablas,columnas, etc) 63 bytes.
13. **Puerto por defecto 5432 (TCP).** Puede ser instalado sin ser administrador
14. Los usuarios de la base de datos son distintos a los usuarios del sistema operativo.
15. Los usuarios son compartidos en todas las bases de datos.
16. El usuario por defecto se llama postgress y es superusuario.
17. Herencia (de campos) y polimorfismo en funciones
18. Mayúsculas y minúsculas
19. Dos archivos principales de configuración: - pg_hba.conf - postgresql.conf - Un archivo pg_ident.conf (no muy usado)
20. Las bases de datos viven en un cluster.

Arquitectura de PostgreSQL

- Vista General
- El proceso postmaster es un proceso supervisor de segundo plano que atiende solicitudes de conexión, luego de autentificar y autorizar crea copias o forks de sí mismo (nuevos procesos servidor) para atender de manera dedicada a cada nueva conexión. De ese punto en adelante el nuevo proceso servidor atiende al cliente sin la intervención del postmaster.
- El proceso postmaster está siempre corriendo, esperando por solicitudes de conexion, mientras que los procesos servidores que las atienden, van y vienen.

Procesos auxiliares

- Procesos mandatorios (que no se tiene la opción de deshabilitar)
  - BGWriter
  - WAL Writer
- Procesos opcionales
  - Stats-collector
  - Autovacuum launcher
  - Archiver
  - Syslogger
  - Wal Sender
  - Wal Receiver

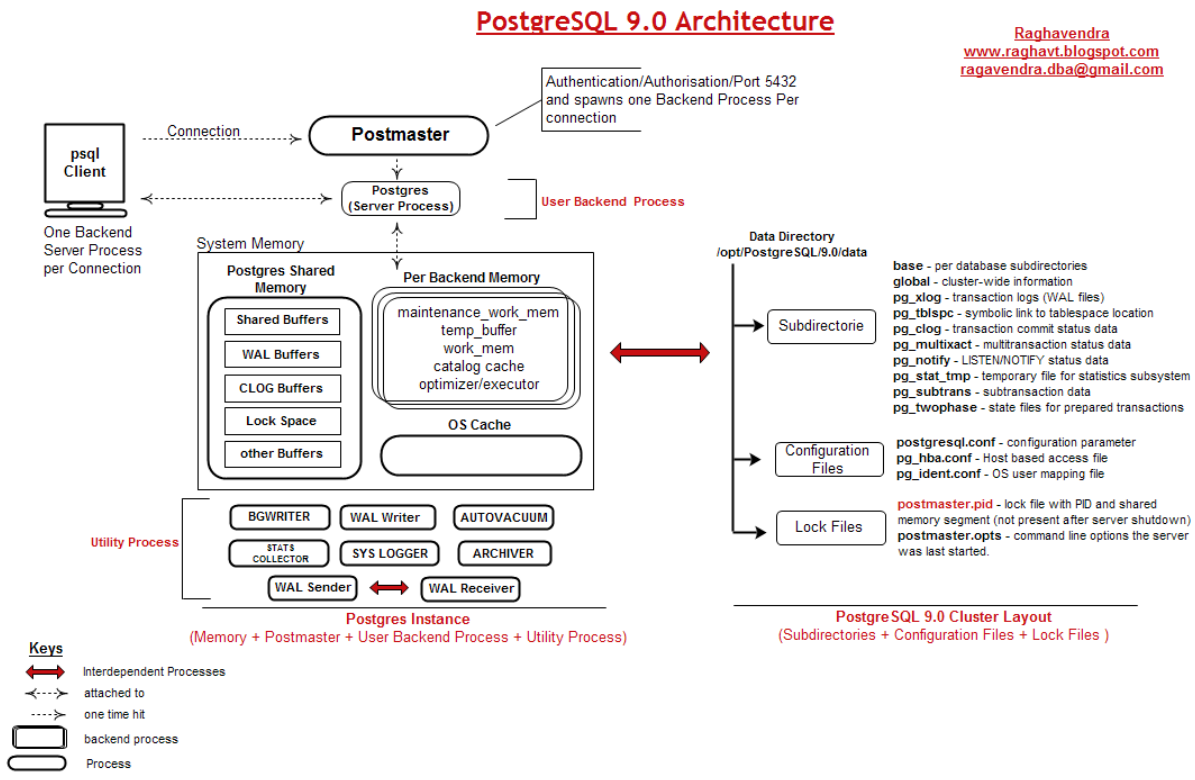

Figure 1.1: Arquitectura de PostgreSQL

- Despliegue de procesos desde línea de comandos

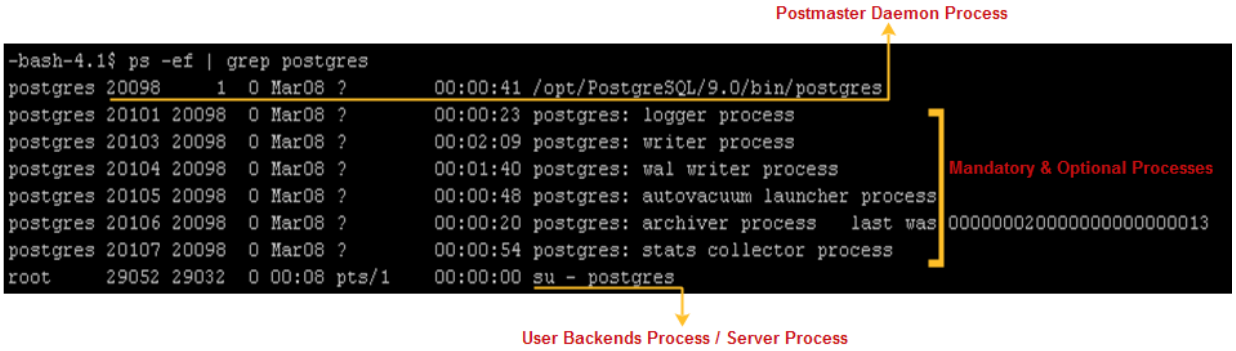

Figure 1.2: Vista de Procesos de Postgres

- Procesos y memoria compartida
- El collector de estadísticas
- Syslogger
- Proceso de archivado (archiver)

El ambiente del Sistema Operativo

Utilidades binarias

initdb

Inicializa un cluster (directorio de datos) de datos de PostgreSQL

pg_ctl

Es una utilidad para inicializar, arrancar, detener o controlar un servidor PostgreSQL

createdb

Crea bases de datos en un cluster PostgreSQL

createuser

Crea un nuevo rol en el servidor PostgreSQL

psql

psql es la terminal interactiva de PostgreSQL

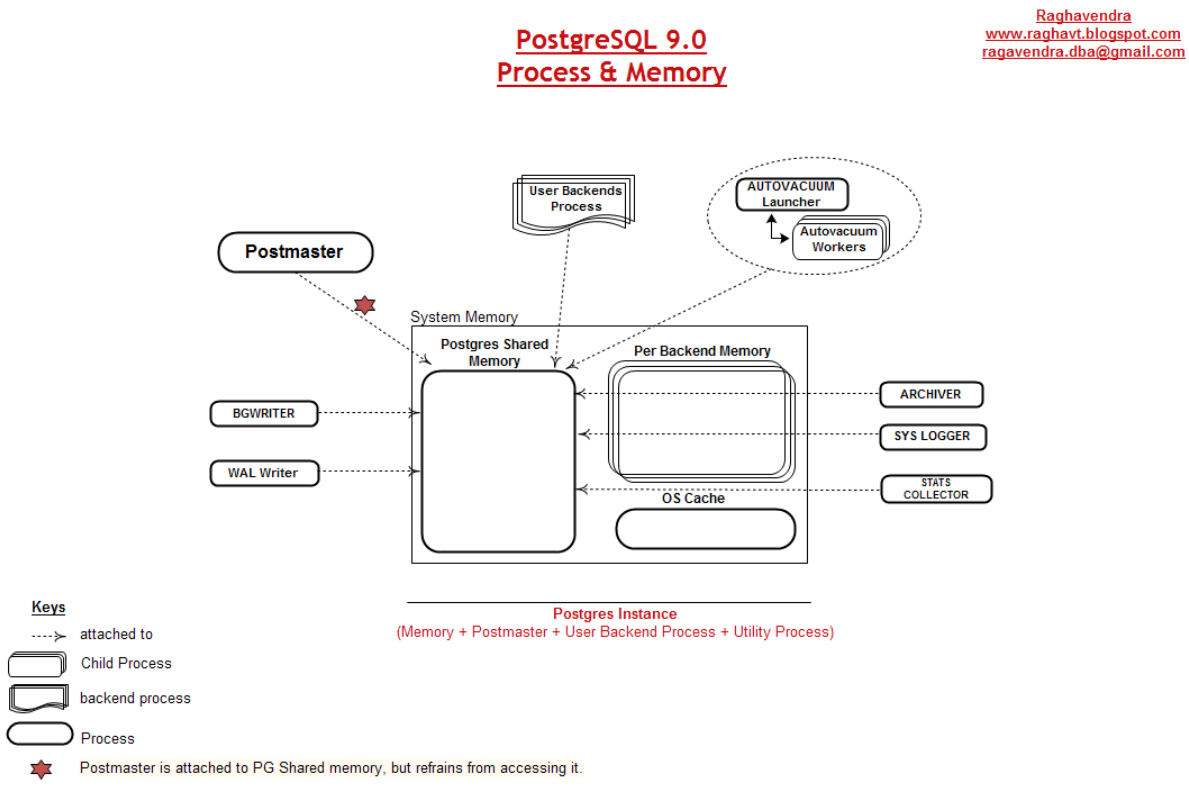

Figure 1.3: Procesos y memoria compartida

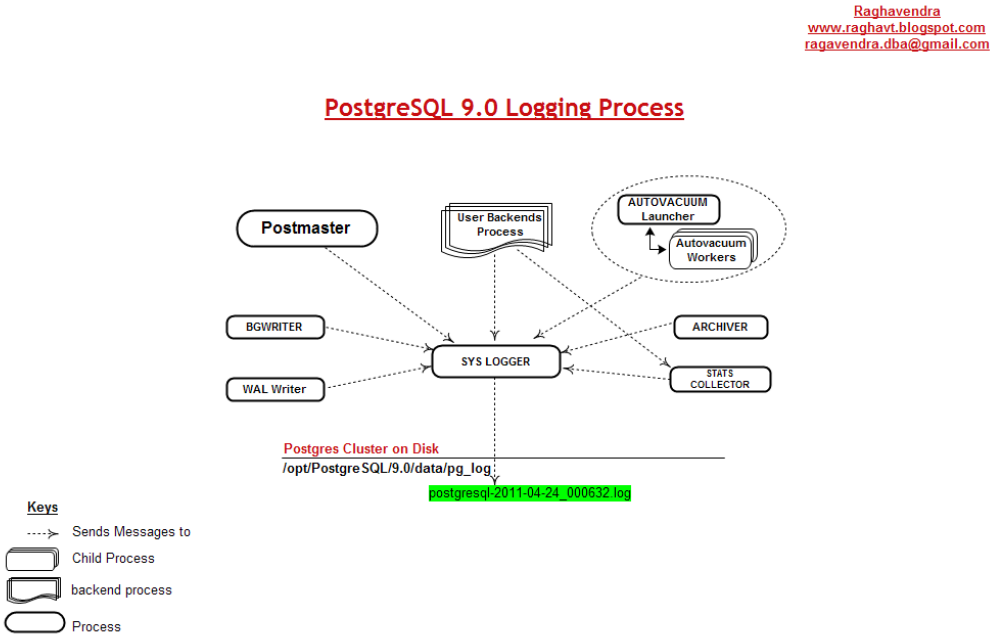

Figure 1.4: Demonio de Registro de Eventos

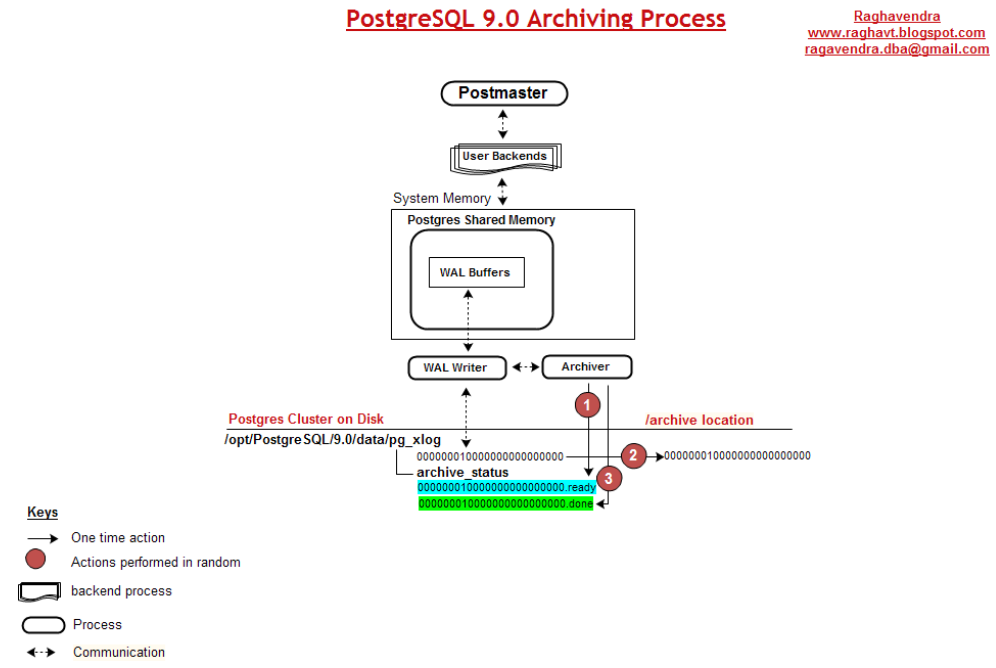

Figure 1.5: El proceso de archivado

pg_dump

genera un respaldo de una base de datos como archivo de texto u otros formatos

La cuenta de usuario de PostgreSQL

Creando un cluster

Iniciando el servidor

Deteniendo el servidor

Revisión de psql

1. Modos interactivo / no interactivo
2. Historia de comandos (con las teclas de flecha)
3. Completado de comandos e identificadores mediante TAB
4. Los comandos terminan con punto y coma, pueden abarcar varias líneas
5. Cuenta con metacomandos que modifican el comportamiento de psql o presentan información.
6. Es la herramienta que utilizaremos para cargar scrips desde archivos de texto (modo no interactivo)

Ejemplos de uso:

```
psql --version
psql --list
psql -U postgres -h localhost postgres
```

Notas adicionales sobre psql:

- el rol del search_path
- listado de bases de datos, y otros objetos
- listado de usuarios con du
- Cuando no indicamos el host tratará de conectarse a localhost por defecto
- revisar exploración de objetos en una base de datos de prueba
- información extendida con +

Bases de datos por defecto

- postgres. la base de datos inicial a dónde pueden conectarse los usuarios de no existir otra
- template1 . Sirve de plantilla por defecto para la creación de nuevas bases de datos.
- template0 . Sirve para reconstruir template1

CHAPTER

TWO

GENERACIÓN DE LOGS

- **Tenemos tres tipos de logs soportados:**
  - stderr (default)
  - csvlog
  - syslog
- El registro de bitácoras está gobernado por postgresql.conf
- Loggeo simultáneo
- Abilidad para controlar verbosidad
- Rotación automática de logs basada en criterios como antigüedad y tamaño
- Los logs manejados por el logger (stderr, csvlog) están almacenados en pg_log
- Los logs manejados por syslog /etc/rsyslog.conf son direccionado de acuerdo a sus reglas específicas

2.1 Parámetros

log_destination

stderr,csvlog,syslog

logging_collector

on / off

El logging_collector se encarga de las tareas de rotación e identificación (nombres) de logs.

log_directory

directorio destino de los logs

CHAPTER

THREE

CONTROLES DE ACCESO A OBJETOS DE LA BASE DE DATOS

3.1 Usuarios, grupos y roles

1. Los usuarios son globales en todo el cluster (los nombres deben ser únicos)
2. El super-usuario postgres
3. **Los privilegios son administrados mediante:**
    - GRANT-REVOKE
    - ALTER
    - CREATE | DROP ROLE | USER
4. Herramientas de línea de comandos : createuser | dropuser
5. Quién crea objetos es su dueño y puede otorgar permisos
6. Cambiar el dueño (ALTER)
7. Rol especial 'PUBLIC', todos los usuarios son miembros
8. La variable PGUSER
9. Permisos por defecto 'ALTER DEFAULT PRIVILEGES'

3.1.1 Comandos de administración de usuarios

- CREATE USER
- ALTER USER
- DROP USER

Creación de usuarios

- Sintaxis básica

```
CREATE USER username
   [ WITH
   | [ ENCRYPTED | UNENCRYPTED] PASSWORD 'password'
   | CREATEDB | NOCREATEDB
```

```
    | CREATEUSER | NOCREATEUSER
    | IN GROUP groupname [, ...]
    | VALID UNITL 'abstime' ]
```

- Ejemplo:

  Crear un usuario llamado 'Manuel' que pueda crear otros usuarios y nuevas bases de datos, cuya cuenta expirará el 25 de Mayo de 2013.

```
CREATE USER manuel PASSWORD 'pgsql'
   CREATEDB CREATEUSER
VALID UNTIL '2013-05-25';
```

Modificar usuarios

- Sintaxis básica

```
ALTER USER username
      [ WITH
      | [ ENCRYPTED | UNENCRYPTED] PASSWORD 'password'
      | CREATEDB | NOCREATEDB
      | CREATEUSER | NOCREATEUSER
      | IN GROUP groupname [, ...]
      | VALID UNITL 'abstime' ]
```

Ejemplo:

Modificar al usuario 'manuel' para que ya no pueda crear nuevas bases de datos.

```
ALTER USER manuel NOCREATEDB;
SELECT * FROM pg_user WHERE usename = 'manuel';
```

3.1.2 Tareas:

Crear un superusuario

```
createuser -e -S -U postgres <nombreDeUsuario>
```

3.2 Grant. Otorgar derechos a usuarios.

Warning: Para borrar o modificar la definición de un objeto de la base de datos, tienes que ser o el dueño del objeto o un super-usuario.

3.2.1 Bases de datos

```
CREATE DATABASE acl;
REVOKE CONNECT ON DATABASE acl FROM PUBLIC;
```

- Tratar de conectarse como usuario 'manuel'
- Otorgarle el permiso de conexión

```
GRANT CONNECT ON DATABASE acl TO manuel;
```

3.2.2 Schemas

```
GRANT {{CREATE|USAGE}}
ON SCHEMA schema_name
TO ROLE;
```

Ejemplo:

- Como usuario postgres, conectarnos a la base de datos acl

```
DROP SCHEMA public CASCADE;
CREATE SCHEMA privado;
CREATE TABLE privado.t1(id serial primary key,val varchar);
INSERT INTO privado.t1 (val) VALUES ('value1');
```

- Tratar de visualizar la tabla privado.t1 como usuario manuel

```
SELECT * FROM privado.t1;
```

- Dale los permisos adecuados

```
GRANT SELECT TABLE  privado.t1 TO manuel;
```

- Trata de seleccionar nuevamente la tabla privado.t1 con el usuario manuel.

Actividad

Si ya se le dió permiso en la tabla, ¿qué es lo que falla?

- Arregla el problema anterior y otórgale permisos de creación de objetos en la base de datos acl al usuario 'manuel'.

3.3 El archivo pg_hba.conf

Características generales.

- Un registro por línea
- Líneas en blanco y el texto que siga a un signo # son ignorados
- Los campos de cada registro puede estar separados por blancos o tabuladores

Composición de cada registro

- Tipo de conexión (local,host,hostssl,hostnossl)
- Nombre de la base de datos
- Nombre de usuario
- El rango de IP del cliente
- Método de autentificación

Orden de las reglas de conexión

- Por cada conexión se exploran las reglas de arriba a abajo

- **La primera que coincide con el tipo de conexión (tipo de conexión, dirección IP del cliente, base de datos solicitada, nomb**

  - Si la autentificación en base a la regla anterior falla, no se siguen explorando otras reglas y se niega la conexión.

- Si ninguna regla aplica, la conexión es rechazada

Formatos de registros

Tipo			Dir.IP	Máscara	Método auth	Opciones auth
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]

El significado de los campos es el siguiente:

- local

  Este registro coincide con intentos de conexión usando sockets del dominio de Unix. Sin algún registro de este tipo, las conexiones utilizando sockets del dominio de Unix no están permitidas.

- host

  Este registro coincide con intentos de conexión hechos utilizando TCP/IP. Registros "host" coinciden con intentos de conexión utilizando o no SSL.

- hostssl

  Este registro coincide con intentos de conexión hechos utilizando TCP/IP, but solo cuando la conexión es hecha con encripción SSL.

- hostnossl

  Este tipo de registro guarda el comportamiento opuesto a hostssl, sólo coincide con intentos de conexión hechos sobre TCP/IP que no usan SSL.

- database

  Una lista de bases de datos separada por comas. Puede ser también una referencia a un archivo anteponiendo una arroba al nombre. all es una referencia a todas las bases de datos. Otros modos: sameuser, samegroup.

- user

  Una lista de usuarios o grupos (especificados con un signo más (+) luego del nombre del grupo) de la base de datos separada por comas. Puede ser también una referencia a un archivo anteponiendo una arroba al nombre. all es una referencia a todos los usuarios.

- address

  Especifica la dirección IP de los equipos cliente. Puede hacer referencia a direcciones en formato CIDR, con máscara en una columna independiente, nombres de hosts. all hace referencia a todas las direcciones

- method

  Esta columna especifica cómo los usuarios que cumplan con las condiciones anteriores serán autentificados.

3.4 Métodos de autentificación

- Revisaremos los más comunes

Método	Descripción
trust	Se permiten conexiones sin password, útil en ambientes experimentales
reject	El usuario es rechazado de manera explícita, sirve para filtrar conexiones de un grupo.
md5	El usuario debe proveer un password encriptado en MD5. Es una buena elección para la mayoría de las situaciones
password	El usuario debe proveer un password en texto plano. Esto no es recomendable por obvias razones.

CHAPTER

FOUR

TIPOS DE DATOS

En el más bajo nivel, PostgreSQL soporta los siguienes tipos de datos

- Booleano
- Caracter
- Númerico
- Temporal (basado en tiempo)
- Binary Large Object (BLOB)

4.1 Booleano

- Formas de especificar valores booleanos

Interpretado como falso	Interpretado como verdadero
'1'	'0'
'yes'	'no'
'y'	'n'
'true'	'false'
't'	'f'

- Ejemplo

```
CREATE DATABASE test;
CREATE TABLE booltype(
id serial primary key,
valused varchar(10),
boolres bool
);
INSERT INTO booltype (valused,boolres) VALUES ('TRUE',true),('1','1'),('t','t'),
   ('no','no'),('f','f'),('Null',NULL),('FALSE',FALSE);
SELECT * FROM booltype;
```

4.2 Caracter

Tenemos tres variantes:

- un sólo caracter
- cadenas de texto de longitud fija

- cadenas de texto de longitud variable

Definición	Significado
char	Un sólo caracter
char(n)	Un conjunto de caracteres de exactamente n caracteres de longitud, rellenados con espacios.
varchar(n)	Un conjunto de caracteres de hasta n caracteres de longitud sin relleno.
varchar	Un conjunto de caracteres sin relleno de longitud ilimitada
text	Un conjunto de caracteres sin relleno de longitud ilimitada

-Ejemplos

```
DROP TABLE chartype;
CREATE TABLE chartype(
   id serial primary key,
   singlechar char,
   fixedchar char(13),
   variablechar   varchar(128)
);

INSERT INTO chartype (singlechar,fixedchar,variablechar)
   VALUES ('F','012345-68910','Bernardo y Leviatan'),
   ('S','1-4XDRGE9@KKÉ','Arbitro'),('F','345-88hhkk','Dimitir'),
   ('F','0-349-10877-8','Whit'), (NULL,'','D.M.S');

INSERT INTO chartype (singlechar,fixedchar,variablechar)
    VALUES('L','Una cadena que es muy larga','L');

SELECT * FROM chartype;

SELECT singlechar,length(singlechar),fixedchar,length(fixedchar),
   variablechar,length(variablechar) FROM chartype;
```

4.3 Numéricos:

- Enteros

Subtipo	Nombre Estandar	Descripción	Rango
small integer	smallint	Entero de 2 bytes con signo	-32768 a 32767
integer	int	Entero de 4 bytes con signo	-2147483648 a 2147483647
serial		Igual que int, asociado a una secuencia	
big integer	bigint	Entero de 8 bytes con signo	-2^63 a 2^63 aprox.
bit serial	bigserial	Igual que bigint,asociado a una secuencia	

- Punto flotante

Subtipo	Nombre Estandar	Descripción
float4	real	Número de punto flotante de 4 bytes con 6 dígitos de precisión
float8,float	double precision	Número de punto flotante de precisión doble - 8 bytes con 15 dígitos de precisión
numeric,decimal	numeric(p,s)	Un número real con p dígitos, s de los cuales van después del punto decimal

- Ejemplos:

```
CREATE TABLE numtype(
asmallint smallint,
anint     int,
afloat    real , --float4
adouble   float, --float8, double precision
anumeric  numeric(5,2)
```

```
);

INSERT INTO numtype VALUES(2, 2,2.0, 2.0, 2.0),
 (-100, -100,123.456789,123.456789, 123.456789),
 (-32768, -123456789, 1.23456789,1.23456789, 1.23456789);

INSERT INTO numtype VALUES(-32768, -123456789,
 123456789.123456789,123456789.123456789, 123456789.123456789);

INSERT INTO numtype VALUES(-32768, -123456789,
   123456789.123456789,123456789.123456789, 123.123456789);

SELECT * FROM numtype;
```

4.4 Tipos de datos temporales

Definición	Significado
date	Almacena información de fechas
time	Almacena información de la hora hasta milisegundos
timestamp	Almacena fecha y hora
interval	Almacena información sobre dif.entre timestamps
timestamptz	Almacena timestamp + timezone
range	Almacena rangos de tiempo como una unidad

4.4.1 timestamptz

¿ Cómo almacena PostgreSQL la información de zona horaria ?

PostgreSQL guarda todo timestamptz haciendo primero la conversión a UTC.

¿ En que zona horaria me muestra PostgreSQL los timestamptz ?

Hace la conversión de UTC a la zona horaria actual del cliente

- Ejemplos:

¿ Cuál es mi zona horaria actual ?

```
show timezone;
```

¿ Qué hora es en Mérida si en el D.F. son las:

```
SELECT '2013-05-19 20:15:18 -5'::timestamptz AT TIME ZONE 'America/Merida';
```

4.4.2 interval

Ejemplos:

```
SELECT '1 year 1 month'::interval;
SELECT '57 minutes'::interval;
SELECT '5 months 2 weeks 3 days 6 hours'::interval;
```

Usualmente es el resultado de sustraer dos timestamp, por ejemplo:

```
SELECT current_timestamp - '2013-05-01';
```

También podemos sumarlas a un timestamp:

```
SELECT '2013-05-01'::date +
   '1 year 1 month 1 week 10 hours 1 minute 16 seconds'::interval;
```

4.4.3 range

- Ejemplo

```
CREATE EXTENSION btree_gist;

CREATE TABLE salon_horario
   (id serial primary key, salon_id integer,clase_id integer, horario tsrange);

ALTER TABLE salon_horario ADD EXCLUDE
   USING gist(salon_id WITH = , horario WITH &&);

INSERT INTO salon_horario VALUES (1,1,2,'[2013-01-10 14:00:00, 2013-01-10 14:30:00)');

INSERT INTO salon_horario VALUES (2,1,3,'[2013-01-10 14:29:00, 2013-01-10 15:00:00)');
```

Referencia

http://www.postgresql.org/docs/current/static/rangetypes.html

4.5 Tipos de datos especiales

4.5.1 Arreglos

Para declarar una columna en una tabla como arreglo, simplemente añade [] después del tipo, no es necesario declarar el número de elementos.

Ejemplo:

- Supón que tenemos una tabla de empleados y queremos habilitar un indicador para mostrar los días de la semana que trabajan.

```
CREATE TABLE empworkday(
id serial primary key,
name varchar,
workdays int[]
);

INSERT INTO empworkday (name,workdays) VALUES ('Miguel','{1,1,1,1,1,0,0}');
INSERT INTO empworkday (name,workdays) VALUES ('Paris','{1,1,1,1,1,1,0}');

SELECT name FROM empworkday WHERE workdays[6] = 1;
```

4.5.2 hstore

Sirve para almacenar conjuntos de pares llave/valor dentro de un solo campo. Puede ser útil en distintos escenarios, como filas con varios atributos que son examinados rara vez, o para datos semi-estructurados. Llaves y valores son

simplemente cadenas de texto.

Mismo ejemplo que el usado para arreglos:

```
CREATE TABLE empworkday_hstore(
id serial primary key,
name varchar,
workdays hstore
);

INSERT INTO empworkday_hstore (name,workdays)
SELECT 'Miguel',hstore(array['Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','Domingo']
,array['1','1','1','1','1','0','0']);

INSERT INTO empworkday_hstore (name,workdays)
SELECT 'Paris',hstore(array['Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','Domingo']
,array['1','1','1','1','1','1','0']);

SELECT (each(workdays)).* FROM empworkday_hstore WHERE name = 'Miguel';

UPDATE empworkday_hstore SET workdays = workdays || hstore('Sabado','1')
 WHERE name = 'Miguel';

SELECT (each(workdays)).* FROM empworkday_hstore WHERE name = 'Miguel';

SELECT akeys(workdays),avals(workdays) FROM empworkday_hstore WHERE name = 'Paris';
SELECT distinct skeys(workdays) FROM empworkday_hstore;
```

Referencia

http://www.postgresql.org/docs/current/static/hstore.html

4.6 Manipulación de datos

4.6.1 Conversión entre tipos de datos

la función cast

cast(column-name AS type-definition-to-convert-to)

o su equivalente:

column-name::type-definition-to-convert-to

- Ejemplos:

```
SELECT cast('2013-05-20'::date AS varchar(10));
SELECT '2013-05-20'::date::varchar(10);

WITH nums AS
   (SELECT  unnest(ARRAY[21.95,9.95,15.75,19.75,25.32,11.49,11.50]) AS nums)
SELECT nums::int FROM nums;
```

4.6.2 Macros o constantes especiales

Podemos usar algunas constantes para obtener información sobre nuestro entorno de trabajo:

```
SELECT current_date,current_time,current_timestamp,
   current_user,current_database(),now();
```

CHAPTER

FIVE

TABLAS

Creación de tablas. Sintaxis Básica

```
CREATE [TEMPORARY] TABLE  table-name (
   { column_name data_type [column-constraint]}
   [ CONSTRAINT table-constraint ]
 ) [INHERITS (existing-table-name)]
```

Consejo Es recomendable manejar nuestros scripts de la estructura de la base de datos por separado, preferentemente mediante control de versiones.

Tip: Hay varias herramientas de modelado gráfico para PostgreSQL, entre ellas SQL Power Architect y pgModeler

5.1 Principales constraints

Definición	Significado
NOT NULL	La columna no puede almacenar valores nulos
UNIQUE	Los valores en la columna no se pueden repetir [1]
PRIMARY KEY	Una combinación de UNIQUE y not NULL, solo una por tabla
DEFAULT	Permite asignar un valor por defecto
CHECK(condición)	Permite validar el dato en inserciónes y actualizaciones
REFERENCES	Forza a que el valor sea alguno de los almacenados en otra tabla (integridad referencial)

Ejemplo:

```
CREATE TABLE dispositivo(
id serial primary key,
nombre varchar not null,
numserie varchar not null unique
);

CREATE TABLE posicion(
id serial primary key,
dispositivo_id int not null,
fecha_hora timestamptz default now(),
mensaje varchar not null,
kilometraje real not null,
constraint fk_dispositivo_posicion FOREIGN KEY (dispositivo_id) REFERENCES dispositivo(id),
constraint kilometraje_positivo check(kilometraje > 0)
```

[1] Los valores nulos son considerados distintos unos de otros

```
);

INSERT INTO dispositivo (nombre,numserie) VALUES ('disp1','1234');
INSERT INTO dispositivo (nombre,numserie) VALUES ('disp2','1234');
INSERT INTO dispositivo (nombre,numserie) VALUES (NULL,'1235');

INSERT INTO posicion ( dispositivo_id,mensaje,kilometraje) VALUES (1,'mensaje1',100.25);
INSERT INTO posicion ( dispositivo_id,mensaje,kilometraje) VALUES (1,'mensaje2',-100.25);
INSERT INTO posicion ( dispositivo_id,mensaje,kilometraje) VALUES (1,NULL,100.25);
```

5.1.1 Limitaciones de constraints

- Solo pueden hacer referencia a columnas de la misma tabla
- No podemos especificar queries, pero podemos referir cualquier funcion

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*