Curso de PostgreSQL y Python

Índice de contenido

CURSO DE POSTGRE SQL Y PYTHON....7
PRESENTACIÓN....7
BASES DE DATOS....8
Definiciones....8
Ejercicio 1....8
Clasificaciones de las Bases de Datos....8
INTRODUCCIÓN A POSTGRESQL....10
Arquitectura....12
Instalación....13
Configuración para el curso....13
Conectarse a PostgreSQL (cliente de terminal)....14
Obteniendo ayuda....15
Crear y Borrar Bases de Datos....20
Conectarse a PostgreSQL (cliente de gráfico)....22
LENGUAJE SQL....25
Orígenes y Evolución....25
Características generales del SQL....26
Optimización....27
Lenguaje de definición de datos (DDL)....27
CREATE....27
ALTER....28
DROP....28
TRUNCATE....28
Lenguaje de manipulación de datos DML (Data Manipulation Language)....29
Definición....29
INSERT....29
UPDATE....32
DELETE....32
Ejercicios con el Lenguaje SQL....32
Ejercicio 1....32
Ejercicio 2....32
Ejercicio 3....33
Ejercicio 4....33
Ejercicio 5....33
Ejercicio 6....33
Ejercicio 7....33
Ejercicio 8....33
Ejercicio 9....33
Ejercicio 10....33
Ejercicio 11....34

Ejercicio 12....34
Administración de PostgreSQL....34
Gestión de Espacio Físico....34
Gestión de Respaldos....35
Respaldo de bajo nivel....35
Respaldar una Base de Datos....36
Respaldar todas las Bases de Datos....37
Tratando con bases de datos de gran tamaño....39
Gestión de Usuarios....41
Crear y borrar usuarios....42
Ver usuarios en el servidor....42
CREATE ROLE vs. CREATE USER....42
Atributos de los Roles....42
Restaurar seguridad de PostgreSQL....43
Privilegios....43
Gestión de Conexiones....46
PROGRAMANDO EN POSTGRESQL....52
Funciones con SQL....52
Sobrecarga de Funciones....54
Lenguajes Procedimentales....54
PL/pgSQL....55
Instalando PL/pgSQL....55
Estructura de programas con PL/pgSQL....56
Declaraciones....58
Alias para Parámetros de una Función....58
Expresiones....59
Declaraciones....59
Estructuras de Control....61
Bucles simples....62
Capturando errores....64
Cursores....65
Disparadores....67
CASO DE ESTUDIO #1....73
Instrucciones....73
Materiales de apoyo en línea....73
INTRODUCCIÓN A PYTHON....74
PRIMEROS PASOS CON PYTHON....78
Intérprete....78
Introspección....78
Funciones incluidas....79
Escribir un programa en un archivo....80

Listas....80
Tuplas....83
Diccionarios....84
Condicionales....85
Bucles....85
Instrucción “for”....86
Función “range”....86
Instrucción “break”....86
Instrucción “pass”....87
Manejo de Errores....87
Funciones....88
Clases....88
Trabajando con Archivos....89
Módulos....90
Paquetes....91
BASES DE DATOS CON PYTHON....92
Instalación....92
Uso del Psycopg2....92
Pasando Parámetros a consultas SQL....93
Errores Frecuentes....94
Adaptación de Tipos de Datos....94
Funciones de interés con cursores....95
CASO DE ESTUDIO #2....96
Caso 2.a....96
Caso 2.b....96
HERRAMIENTAS PARA TRABAJAR CON PYTHON....97
Entornos Virtuales....97
Control de Versiones....98
Documentación....100
Instalando Sphinx....100
QuickStart....100
conf.py....101
Recursos para reStructuredText....101
Editores e IDEs....102
PROGRAMACIÓN WEB....103
HTML....103
Etiquetas de HTML....103
Listado de Etiquetas HTML....104
Atributos HTML....107
Mi primera página Web....107
Mi segunda página Web....108

Herramienta....108
JavaScript....109
Ejemplos de JavaScript....109
¿Dónde colocar el código JavaScript?....110
Principios de JavaScript....111
Declaraciones....111
Bloques de código....112
Comentarios....112
Variables....112
Operadores....113
Operadores Lógicos....114
Condicionales....114
Mensajes desde JavaScript....115
Bucles....116
Funciones....118
Capturando Errores....119
Objetos....120
CSS....121
Sintaxis....121
Selector por id....122
Selector por clase....122
¿Dónde colocar las hojas de estilo?....122
PROGRAMACIÓN WEB CON PYTHON (CGI)....123
Ejercicio 1....125
INTRODUCCIÓN A DJANGO....126
Definición....126
Instalación....126
Iniciando un Proyecto....127
settings.py....127
urls.py....130
Probando nuestro Proyecto....130
Creando nuestra primera Aplicación....131
Modelo....131
Habilitando el Admin de Django....132
Ahora un poco de programación....132
Siguientes pasos....133
PROGRAMACIÓN GUI CON PYTHON....134
Tkinter....134
Hola Mundo (al estilo Tkinter)....134
Hola de Nuevo....134
Manejo de Eventos....135

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve
CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Creando un Menú....136
Barras de Herramientas....137
Simple Ventana de Diálogo....137
Ejercicio 1....138
INTERFACES GRÁFICAS CON PYTHON Y GTK....139
Definición....139
Ejemplo #1 - Usando el terminal....139
Ejemplo #2 - Escribiendo un script de PyGTK....139

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CURSO DE POSTGRE SQL Y PYTHON

PRESENTACIÓN

Las bases de datos son de gran importancia en un mundo inmerso dentro de las Tecnologías de Información y Comunicación. El conocer su funcionamiento y capacidades permite ampliar los horizontes respecto a las posibilidades que ofrecen las TIC. Así mismo, es necesario poder implementar los conceptos de las bases de datos en un ambiente programable de rápido desarrollo para permitir concretar dichos conocimientos en elementos tangibles por el usuario. Para ello se seleccionó Python, no sólo por su facilidad de uso, sino también por la gran cantidad de herramientas y soluciones que permiten construir soluciones efectivas con facilidad.

BASES DE DATOS

Definiciones

Una base de datos o banco de datos (en ocasiones abreviada con la sigla BD o con la abreviatura b.d.) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

-Wikipedia
A efectos prácticos se entiende por **Base de Datos** un registro o catálogo organizado de información que identifica una colección de elementos.

Ejercicio 1

¿Cómo organizaría usted una base de datos de sus CD de música y sus DVD de películas?

Clasificaciones de las Bases de Datos

Según su tipo:

1. Según la variabilidad de los datos almacenado
    1. Bases de datos estáticas
    2. Bases de datos dinámicas
2. Según el contenido
    1. Bases de datos bibliográficas
    2. Bases de datos de texto completo
    3. Directorios
    4. Bases de datos o “bibliotecas”

Según su Modelo:

1. Bases de datos jerárquicas
2. Base de datos de red
3. Bases de datos transaccionales

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

4. Bases de datos relacionales
5. Bases de datos multidimensionales
6. Bases de datos orientadas a objetos
7. Bases de datos documentales
8. Bases de datos deductivas
9. Gestión de bases de datos distribuida (SGBDD)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

INTRODUCCIÓN A POSTGRESQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Desarrollado por la Universidad de California en el Departamento de Ciencias de la Computación en Berkeley. Es basado en POSTGRES Versión 4.2 quien fue pionero en muchos conceptos que estuvieron disponibles en algunas bases de datos comerciales mucho después.

PostgreSQL es un descendiente de código abierto del código original de Berkeley. Soporta una gran parte del estándar SQL y ofrece muchas características modernas:

- consultas complejas
- llaves foráneas
- disparadores
- vistas
- integridad transaccional
- control de concurrencia en múltiples versiones

Así mismo, PostgreSQL puede ser ampliado en muchos sentidos, como por ejemplo agregando nuevos:

- tipos de datos
- funciones
- operadores
- agregados de funciones
- métodos índices
- lenguajes procedimentales

Así mismo por su licencia libre, PostgreSQL puede ser usado, modificado y distribuido por cualquier persona, libre de pago para cualquier propósito, privado, comercial o académico.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Otras de sus principales características son:

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente, los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Detalles de algunas características:

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Disparadores (triggers): Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora, todos los disparadores se definen por seis características:
    - El nombre del disparador o trigger
    - El momento en que el disparador debe arrancar
    - El evento del disparador deberá activarse sobre...
    - La tabla donde el disparador se activará
    - La frecuencia de la ejecución
    - La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Soporte para transacciones distribuidas.

Adicionalmente PostgreSQL permite la definición de funciones, tanto disparadoras como regulares mediante diversos lenguajes de programación como lo son:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de Oracle).
- Lenguaje C.
- C++.
- Java PL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.
- Lenguaje para aplicaciones estadísticas R por medio de PL/R

entre otros.

Arquitectura

PostgreSQL tiene una arquitectura cliente/servidor, esto significa que una sesión del programa consta de dos procesos (programas) que cooperan entre si:

- El servidor: es un proceso que gestiona los archivos de la base de datos, acepta conexiones, procesa las operaciones solicitadas por el cliente. El programa servidor se llama *postgres*.
- El cliente: es el programa que interactúa con el usuario u otro proceso que desee realizar operaciones dentro de la base de datos. Las herramientas clientes son muy diversas en su naturaleza y tipos, sin embargo podemos agruparlas en las siguientes categorías:
    - de texto (o interfaz de línea de comandos)
    - gráficas
    - de acceso web

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

- de administración de la bases de datos
- geográficas
- librerías especializadas

Es típico que en una aplicación con arquitectura cliente/servidor, el cliente y el servidor se encuentren en computadores distintos. En este caso ellos se comunican mediante una conexión de red TCP/IP.

Instalación

Para la instalación de PostgreSQL en Canaima GNU/Linux se requiere la instalación de los siguientes paquetes:

- postgresql
- postgresql-8.3
- postgresql-client
- postgresql-client-8.3
- postgresql-client-common
- postgresql-contrib
- postgresql-contrib-8.3
- postgresql-doc
- postgresql-doc-8.3
- pgadmin3
- pgadmin3-data

Para ello puede utilizar el gestor de paquetes **Synaptic** o en su defecto se recomienda el uso del comando **aptitude**:

```
$ su
$ aptitude install postgresql postgresql-8.3 postgresql-client postgresql-client-8.3 postgresql-client-common postgresql-contrib postgresql-contrib-8.3 postgresql-doc postgresql-doc-8.3 pgadmin3 pgadmin3-data
```

Configuración para el curso

Para efectos del curso optaremos por librar la base de datos de sus mecanismos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

de seguridad. Ello para concentrarnos en las funcionalidades del programa y luego, a su debido momento en las opciones de seguridad.
Para ello, como superusuario, proceda a escribir la siguiente línea en su terminal:

$ nano /etc/postgresql/8.3/main/pg_hba.conf

Edite el archivo de modo que quede como el siguiente archivo pg_hba.conf.

A continuación proceda a reiniciar el servicio de postgres para actualizar la configuración que acabamos de hacer.

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

Conectarse a PostgreSQL (cliente de terminal)

Para conectarse a postgreSQL utilice el comando **psql**, observe las opciones de esta comando escribiendo lo siguiente en su terminal:

$ psql --help

Este es psql 8.3.14, el terminal interactivo de PostgreSQL.

Modo de empleo:
  psql [OPCIONES]...[BASE-DE-DATOS [USUARIO]]

Opciones generales:
  -d BASE-DE-DATOS nombre de base de datos a conectarse (por omisión: "carlos")
  -c ORDEN   ejecutar sólo una orden (SQL o interna) y salir
  -f ARCHIVO   ejecutar órdenes desde archivo, luego salir
   -1 («uno»)   ejecutar archivo en una única transacción
  -l   listar bases de datos, luego salir

Opciones de entrada y salida:
  -a   mostrar las órdenes del script
  -e   mostrar órdenes enviadas al servidor
  -E   mostrar consultas generadas por órdenes internas
  -q   modo silencioso (sin mensajes, sólo resultado de consultas)
  -o ARCHIVO   enviar resultados de consultas a archivo (u |orden)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
  -n               deshabilitar edición de línea de órdenes (readline)
  -s               modo paso a paso (confirmar cada consulta)
  -S               modo de líneas (fin de línea termina la orden SQL)
  -L ARCHIVO       manda el log de la sesión a un archivo

Opciones de formato de salida:
  -A               modo de salida desalineado
  -H               modo de salida en tablas HTML
                  (-P format=html)
  -t               mostrar sólo filas (-P tuples_only)
  -T TEXTO         definir atributos de marcas de tabla HTML (ancho, borde)
                  (-P tableattr=)
  -x               activar modo expandido de salida de tablas (-P expanded)
  -P VAR[=ARG]     definir opción de impresión VAR en ARG (ver orden \pset)
  -F CADENA        definir separador de columnas (por omisión: «|»)
                  (-P fieldsep=)
  -R CADENA        definir separador de filas (por omisión: salto de línea)
                  (-P recordsep=)

Opciones de conexión:
  -h ANFITRIÓN     nombre del anfitrión o directorio de socket
                  (por omisión: «/var/run/postgresql»)
  -p PORT          puerto del servidor de bases de datos (por omisión: "5432")
  -U NOMBRE        nombre de usuario de la base de datos (por omisión: "postgres")
  -W               forzar petición de contraseña (debería ser automático)

Para obtener más ayuda, digite «\?» (para órdenes internas) o «\help»
(para órdenes SQL) dentro de psql, o consulte la sección de psql
en la documentación de PostgreSQL.

Reporte errores a <pgsql-bugs@postgresql.org>.
$
Proceda a conectarse con el comando:
$ psql -U postgres -d postgres -h localhost
```

Obteniendo ayuda

Principalmente obtenemos ayuda dentro de **psql** mediante los siguientes dos

comandos.

\?

```
postgres=#\?
General
  \c[onnect] [BASE-DE-DATOS|- USUARIO|- ANFITRIÓN|- PUERTO|-]
                conectar a una nueva base de datos (actual: «postgres»)
  \cd [DIR]       cambiar el directorio de trabajo
  \copyright      mostrar términos de uso y distribución de PostgreSQL
  \encoding [CODIFICACIÓN]
                mostrar o definir codificación del cliente
  \h [NOMBRE]     mostrar ayuda de sintaxis de órdenes SQL,
                * para todas las órdenes
  \prompt [TEXTO] NOMBRE
                preguntar al usuario el valor de la variable interna
  \password [USUARIO]
                cambiar la contraseña para un usuario en forma segura
  \q              salir de psql
  \set [NOMBRE [VALOR]]
                definir variables internas,
                listar todas si no se dan parámetros
  \timing         mostrar tiempo de ejecución de órdenes
                (actualmente desactivado)
  \unset NOMBRE   indefinir (eliminar) variable interna
  \! [ORDEN]      ejecutar orden en intérprete de órdenes (shell),
                o iniciar intérprete interactivo

Búfer de consulta
  \e [ARCHIVO]    editar el búfer de consulta (o archivo) con editor externo
  \g [ARCHIVO]    enviar búfer de consulta al servidor
                (y resultados a archivo u |orden)
  \p              mostrar el contenido del búfer de consulta
  \r              reiniciar (limpiar) el búfer de consulta
  \s [ARCHIVO]    mostrar historial de órdenes o guardarlo en archivo
  \w ARCHIVO      escribir búfer de consulta a archivo

Entrada/Salida
  \echo [CADENA]  escribir cadena a salida estándar
  \i ARCHIVO      ejecutar órdenes desde archivo
  \o [ARCHIVO]    enviar resultados de consultas a archivo u |orden
  \qecho [CADENA] escribir cadena a salida de consultas (ver \o)
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Informacional
  \d [NOMBRE]   describir tabla, índice, secuencia o vista
  \d{t|i|s|v|S} [PATRÓN] («+» para obtener más detalles)
                listar tablas/índices/secuencias/vistas/tablas de sistema
  \da [PATRÓN]   listar funciones de agregación
  \db [PATRÓN]   listar tablespaces («+» para más detalles)
  \dc [PATRÓN]   listar conversiones

  \dC   listar conversiones de tipo (casts)
  \dd [PATRÓN]   listar comentarios de objetos
  \dD [PATRÓN]   listar dominios
  \df [PATRÓN]   listar funciones («+» para más detalles)
  \dF [PATRÓN]   listar configuraciones de búsqueda en texto
                («+» para más detalles)
  \dFd [PATRÓN]   listar diccionarios de búsqueda en texto
                («+» para más detalles)
  \dFt [PATRÓN]   listar plantillas de búsqueda en texto
  \dFp [PATRÓN]   listar analizadores de búsqueda en texto
                («+» para más detalles)
  \dg [PATRÓN]   listar grupos
  \dn [PATRÓN]   listar esquemas («+» para más detalles)
  \do [NOMBRE]   listar operadores
  \dl   listar objetos grandes, lo mismo que \lo_list
  \dp [PATRÓN]   listar privilegios de acceso a tablas, vistas y secuencias
  \dT [PATRÓN]   listar tipos de dato («+» para más detalles)
  \du [PATRÓN]   listar usuarios
  \l   listar todas las bases de datos («+» para más detalles)
  \z [PATRÓN]   listar privilegios de acceso a tablas, vistas y secuencias
                (lo mismo que \dp)

Formato
  \a   cambiar entre modo de salida alineado y sin alinear
  \C [CADENA]   definir título de tabla, o indefinir si es vacío
  \f [CADENA]   mostrar o definir separador de campos para
                modo de salida sin alinear
  \H   cambiar modo de salida HTML (actualmente desactivado)
  \pset NOMBRE [VALOR]
                define opción de salida de tabla
                (NOMBRE := {format|border|expanded|fieldsep|footer|null|
                numericlocale|recordsep|tuples_only|title|tableattr|pager})
  \t   mostrar sólo filas (actualmente desactivado)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
  \T [CADENA]     definir atributos HTML de <table>, o indefinir si es vacío
  \x              cambiar modo expandido (actualmente desactivado)

Copy, Objetos Grandes
  \copy ...       ejecutar orden SQL COPY con flujo de datos al cliente
  \lo_export LOBOID ARCHIVO
  \lo_import ARCHIVO [COMENTARIO]
  \lo_unlink LOBOID
  \lo_list        operaciones con objetos grandes
```

y **\h**

```
postgres=# \h
Ayuda disponible:
```

ABORT	COPY	DROP AGGREGATE	LISTEN
ALTER AGGREGATE	CREATE AGGREGATE	DROP CAST	LOAD
ALTER CONVERSION	CREATE CAST	DROP CONVERSION	LOCK
ALTER DATABASE	CREATE CONSTRAINT TRIGGER	DROP DATABASE	MOVE
ALTER DOMAIN	CREATE CONVERSION	DROP DOMAIN	NOTIFY
ALTER FUNCTION	CREATE DATABASE	DROP FUNCTION	PREPARE
ALTER GROUP	CREATE DOMAIN	DROP GROUP	PREPARE TRANSACTION
ALTER INDEX	CREATE FUNCTION	DROP INDEX	REASSIGN OWNED
ALTER LANGUAGE	CREATE GROUP	DROP LANGUAGE	REINDEX
ALTER OPERATOR CLASS	CREATE INDEX	DROP OPERATOR CLASS	RELEASE SAVEPOINT
ALTER OPERATOR	CREATE LANGUAGE	DROP OPERATOR	RESET
ALTER OPERATOR FAMILY	CREATE OPERATOR CLASS	DROP OPERATOR FAMILY	REVOKE
ALTER ROLE	CREATE OPERATOR	DROP OWNED	ROLLBACK
ALTER SCHEMA	CREATE OPERATOR FAMILY	DROP ROLE	ROLLBACK PREPARED
ALTER SEQUENCE	CREATE ROLE	DROP RULE	ROLLBACK TO SAVEPOINT
ALTER TABLE	CREATE RULE	DROP SCHEMA	SAVEPOINT
ALTER TABLESPACE	CREATE SCHEMA	DROP SEQUENCE	SELECT
ALTER TRIGGER	CREATE SEQUENCE	DROP TABLE	SELECT INTO
ALTER TEXT SEARCH CONFIGURATION	CREATE TABLE	DROP TABLESPACE	SET
ALTER TEXT SEARCH DICTIONARY	CREATE TABLE AS	DROP TRIGGER	SET CONSTRAINTS
ALTER TEXT SEARCH PARSER	CREATE TABLESPACE	DROP TEXT SEARCH CONFIGURATION	SET ROLE

ALTER TEXT SEARCH TEMPLATE	CREATE TRIGGER	DROP TEXT SEARCH DICTIONARY	SET SESSION AUTHORIZATION
ALTER TYPE	CREATE TEXT SEARCH CONFIGURATION	DROP TEXT SEARCH PARSER	SET TRANSACTION
ALTER USER	CREATE TEXT SEARCH DICTIONARY	DROP TEXT SEARCH TEMPLATE	SHOW
ALTER VIEW	CREATE TEXT SEARCH PARSER	DROP TYPE	START TRANSACTION
ANALYZE	CREATE TEXT SEARCH TEMPLATE	DROP USER	TRUNCATE
BEGIN	CREATE TYPE	DROP VIEW	UNLISTEN
CHECKPOINT	CREATE USER	END	UPDATE
CLOSE	CREATE VIEW	EXECUTE	VACUUM
CLUSTER	DEALLOCATE	EXPLAIN	VALUES
COMMENT	DECLARE	FETCH	
COMMIT	DELETE	GRANT	
COMMIT PREPARED	DISCARD	INSERT	

postgres=#
Este último comando podemos ilustrarlo con un ejemplo de cómo consultarlo.
postgres=# \h SELECT
Orden: SELECT
Descripción: recupera filas desde una tabla o vista
Sintaxis:

```
SELECT [ ALL | DISTINCT [ ON ( expresión [, ...] ) ] ]
    * | expresión [ AS nombre_salida ] [, ...]
    [ FROM item_from [, ...] ]
    [ WHERE condición ]
    [ GROUP BY expresión [, ...] ]
    [ HAVING condición [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
      [ ORDER BY expresión [ ASC | DESC | USING operador ] [ NULLS { FIRST |
LAST } ] [, ...] ]
    [ LIMIT { cantidad | ALL } ]
    [ OFFSET inicio ]
    [ FOR { UPDATE | SHARE } [ OF nombre_tabla [, ...] ] [ NOWAIT ] [...] ]
```

donde item_from puede ser uno de:

```
    [ ONLY ] nombre_tabla [ * ] [ [ AS ] alias [ ( alias_columna [, ...] ) ] ]
    ( select ) [ AS ] alias [ ( alias_columna [, ...] ) ]
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

nombre_función ([argumento [, ...]]) [AS] alias [(alias_columna [, ...] | definición_columna [, ...])]
nombre_función ([argumento [, ...]]) AS (definición_columna [, ...])
item_from [NATURAL] tipo_join item_from [ON condición_join | USING (columna_join [, ...])]
postgres=#
Salga de postgres y permanezca en su terminal.
postgres=#\q
$

Crear y Borrar Bases de Datos

Para crear una base de datos en el terminal utilice el siguiente comando:
$ createdb --help
createdb crea una base de datos PostgreSQL.

Empleo:
createdb [OPCIÓN]... [NOMBRE] [DESCRIPCIÓN]

Opciones:
-D, --tablespace=TBLSPC tablespace por omisión de la base de datos
-E, --encoding=CODIFICACIÓN
codificación para la base de datos
-O, --owner=DUEÑO usuario que será dueño de la base de datos
-T, --template=PATRÓN base de datos patrón a copiar
-e, --echo mostrar las órdenes enviadas al servidor
--help mostrar esta ayuda y salir
--version mostrar el número de versión y salir

Opciones de conexión:
-h, --host=ANFITRIÓN nombre del servidor o directorio del socket
-p, --port=PUERTO puerto del servidor
-U, --username=USUARIO nombre de usuario para la conexión
-W, --password forzar la petición de contraseña

Si no se especifica, se creará una base de datos con el mismo nombre que el usuario actual.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Reporte errores a <pgsql-bugs@postgresql.org>.
$
Es decir que usted escribirá algo similar a esto:

```
$ createdb -E UTF-8 -U postgres -h localhost prueba
```

Análogamente, para borrar una base de datos utilice el comando **dropdb**:

```
$ dropdb --help
dropdb elimina una base de datos de PostgreSQL.

Empleo:
  dropdb [OPCIÓN]... BASE-DE-DATOS

Opciones:
  -e, --echo                mostrar las órdenes a medida que se ejecutan
  -i, --interactive         preguntar antes de eliminar
  -h, --host=ANFITRIÓN      nombre del servidor o directorio del socket
  -p, --port=PUERTO         puerto del servidor
  -U, --username=USUARIO    nombre de usuario para la conexión
  -W, --password            forzar la petición de contraseña
  --help                    desplegar esta ayuda y salir
  --version                 desplegar información de versión y salir

Reporte errores a <pgsql-bugs@postgresql.org>.
$
```

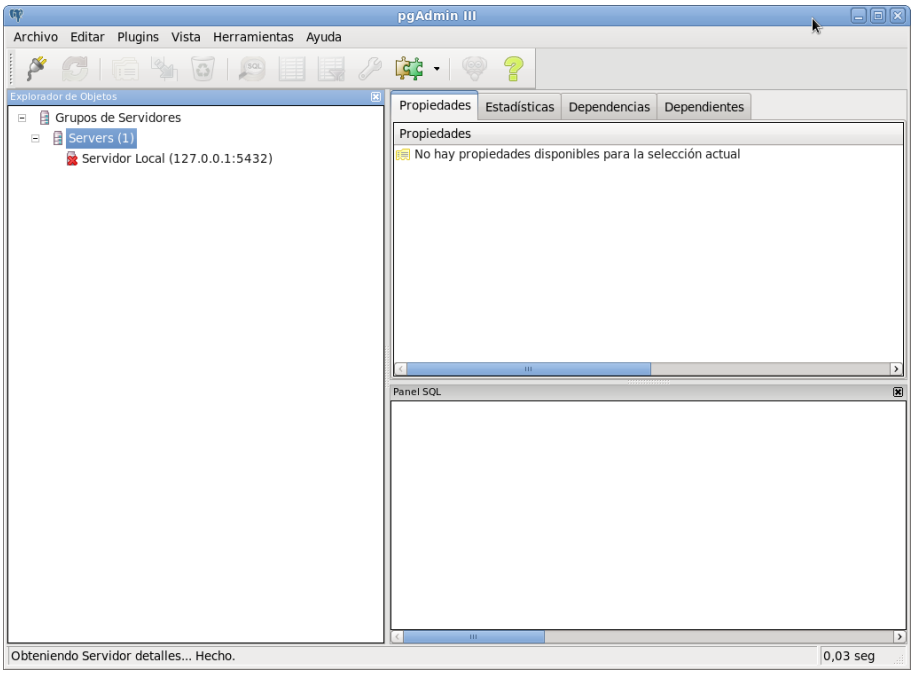
En otras palabras usted escribirá algo similar a esto:

```
$ dropdb -U postgres -h localhost prueba
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Conectarse a PostgreSQL (cliente de gráfico)

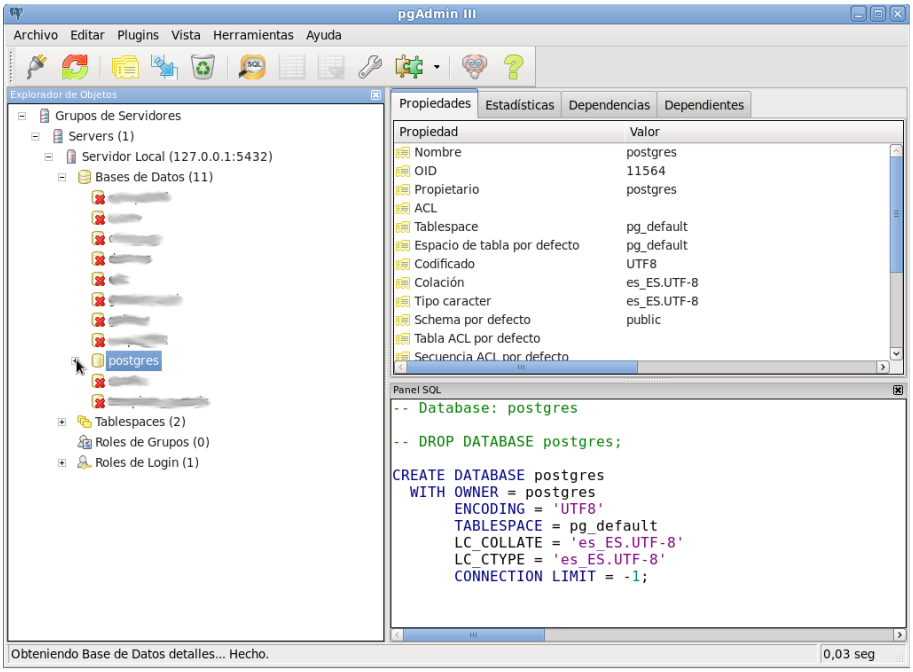
Existen varios clientes gráficos para PostgreSQL, sin embargo el más conocido y difundido es el **pgAdmin III**:



Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

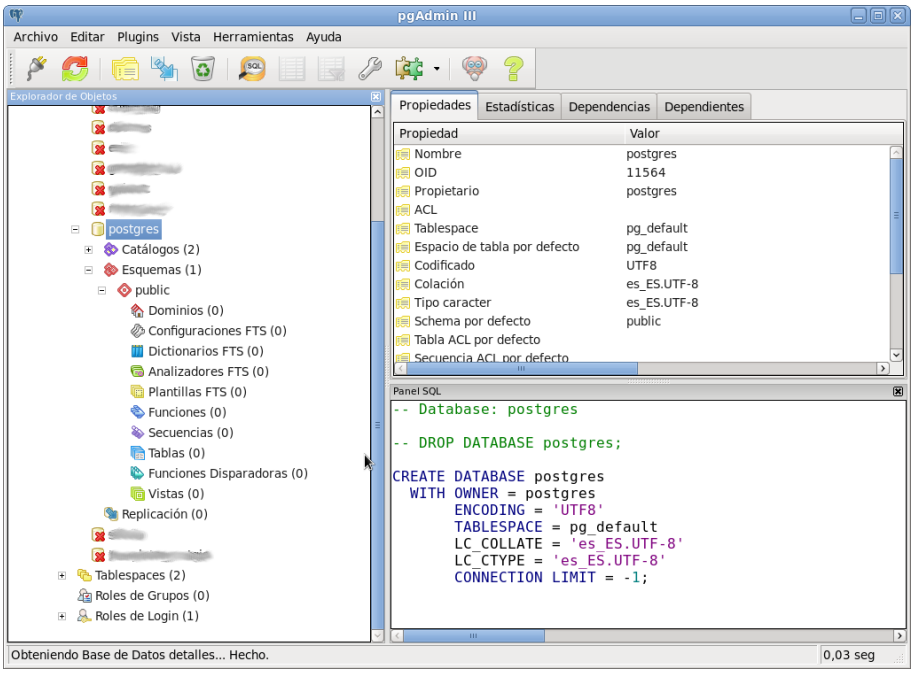
Cree una conexión al servidor.

posteriormente verá la lista de bases de datos en su computador



Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

dentro de cada base de datos podrá ver los elementos que la componen



LENGUAJE SQL

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

Orígenes y Evolución

Los orígenes del SQL están ligados a las de las bases de datos relacionales. En 1970 E. F. Codd propone el modelo relacional y asociado a éste un sublenguaje de acceso a los datosbasado en el cálculo de predicados. Basándose en estas ideas, los laboratorios de IBM definen el lenguaje SEQUEL (Structured English QUEry Language) que más tarde sería ampliamente implementado por el sistema de gestión de bases de datos (SGBD) experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos sistemas de gestión de bases de datos relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el “SQL-86” o “SQL1”. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo, este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado “SQL-92” o “SQL2”.

En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

El ANSI SQL sufrió varias revisiones y agregados a lo largo del tiempo:

Año	Nombre	Alias	Comentarios
1986	SQL-86	SQL-87	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	SQL-89		Revisión menor.
1992	SQL-92	SQL2	Revisión mayor.
1999	SQL:1999	SQL2000	Se agregaron expresiones regulares, consultas recursivas (para relaciones jerárquicas), triggers y algunas características orientadas a objetos.
2003	SQL:2003		Introduce algunas características de XML, cambios en las funciones, estandarización del objeto sequence y de las columnas autonuméricas.
2006	SQL:2006		ISO/IEC 9075-14:2006 Define las maneras en las cuales el SQL se puede utilizar conjuntamente con XML. Define maneras importar y guardar datos XML en una base de datos SQL, manipulándolos dentro de la base de datos y publicando el XML y los datos SQL convencionales en forma XML. Además, proporciona facilidades que permiten a las aplicaciones integrar dentro de su código SQL el uso de XQuery, lenguaje de consulta XML publicado por el W3C (World Wide Web Consortium) para acceso concurrente a datos ordinarios SQL y documentos XML.
2008	SQL:2008		Permite el uso de la cláusula ORDER BY fuera de las definiciones de los cursores. Incluye los disparadores del tipo INSTEAD OF. Añade la sentencia TRUNCATE.

Características generales del SQL

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones en éstos últimos.

Es un lenguaje declarativo de “alto nivel” o “de no procedimiento”, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación y la orientación a objetos. De esta forma una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros.

Optimización

Como ya se dijo arriba, y suele ser común en los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de su ejecución. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.

Lenguaje de definición de datos (DDL)

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE

Este comando crea un objeto dentro de la base de datos

Ejemplo (crear una función)

```
CREATE OR REPLACE FUNCTION 'NOMBRE FUNCION'('PARAMETROS')
  RETURNS 'TIPO RETORNO' AS
$BODY$
begin
'INSTRUCCIÓN SQL'
--por Ejemplo:
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
DELETE FROM con_empleado WHERE id_empleado = 'ANY' (ids);
end;
$BODY$
LANGUAGE 'plpgsql';
```

ALTER

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.

Ejemplo (agregar columna a una tabla)

```
ALTER TABLE 'TABLA_NOMBRE' (
  ADD NUEVO_CAMPO INT UNSIGNED meel
)
```

DROP

Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo

```
ALTER TABLE 'TABLA_NOMBRE' (
    DROP COLUMN 'CAMPO_NOMBRE1'
)
```

TRUNCATE

Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DROP, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE sólo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Manipulación de Datos), es en realidad una DDL, ya que internamente, el comando TRUNCATE borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo

```
TRUNCATE TABLE ''TABLA_NOMBRE1''
```

Lenguaje de manipulación de datos DML (Data Manipulation Language)

Definición

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional.

INSERT

Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.

Forma básica
INSERT INTO ''tabla'' (''columna1'', [''columna2,... '']) VALUES (''valor1'', [''valor2,...''])
Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo
INSERT INTO agenda_telefonica (nombre, numero) VALUES ('Roberto Jeldrez', 4886850);
Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:
INSERT INTO ''tabla'' VALUES (''valor1'', [''valor2,...''])
Ejemplo (asumiendo que ‘nombre’ y ‘número’ son las únicas columnas de la tabla ‘agenda_telefonica’):
INSERT INTO agenda_telefonica VALUES ('Roberto Jeldrez', 080473968);

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Formas avanzadas

Inserciones en múltiples filas

Una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO ''tabla'' (''columna1'', [''columna2,... ''])
        VALUES (''valor1a'', [''valor1b,...'']), (''value2a'', [''value2b,...'']),...
```

Esta característica es soportada por DB2, PostgreSQL (desde la versión 8.2), MySQL, y H2.

Ejemplo (asumiendo que 'nombre' y 'número' son las únicas columnas en la tabla 'agenda_telefonica'):

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850'), ('Alejandro Sosa', '4556550');
```

Que podía haber sido realizado por las sentencias

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850');
INSERT INTO agenda_telefonica VALUES ('Alejandro Sosa', '4556550');
```

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente rendimiento que la sentencia de inserción múltiple.

Para insertar varias filas en MS SQL puede utilizar esa construcción:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212'
UNION ALL
SELECT 'Peter Doe', '555-2323';
```

Tenga en cuenta que no se trata de una sentencia SQL válida de acuerdo con el estándar SQL (SQL: 2003), debido a la cláusula subselect incompleta.

Para hacer lo mismo en Oracle se usa DUAL TABLE, siempre que se trate de solo una simple fila:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM DUAL
UNION ALL
SELECT 'Peter Doe','555-2323' FROM DUAL
```

Una implementación conforme al estándar de esta lógica se muestra el siguiente ejemplo, o como se muestra arriba (no aplica en Oracle):

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM LATERAL ( VALUES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VALUES (1) ) AS t(c)
```

Copia de filas de otras tablas

Un INSERT también puede utilizarse para recuperar datos de otros, modificarla si es necesario e insertarla directamente en la tabla. Todo esto se hace en una sola sentencia SQL que no implica ningún procesamiento intermedio en la aplicación cliente. Un SUBSELECT se utiliza en lugar de la cláusula VALUES. El SUBSELECT puede contener JOIN, llamadas a funciones, y puede incluso consultar en la misma TABLA los datos que se inserta. Lógicamente, el SELECT se evalúa antes que la operación INSERT esté iniciada. Un ejemplo se da a continuación.
INSERT INTO phone_book2

```
SELECT *
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe')
```

Una variación es necesaria cuando algunos de los datos de la tabla fuente se está insertando en la nueva tabla, pero no todo el registro. (O cuando los esquemas de las tablas no son iguales.)

```
INSERT INTO phone_book2 ( [name], [phoneNumber] )

SELECT [name], [phoneNumber]
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe')
```

El SELECT produce una tabla (temporal), y el esquema de la tabla temporal debe coincidir con el esquema de la tabla donde los datos son insertados.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

UPDATE

Una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.

Forma básica

UPDATE 'tabla'
  SET ''columna1'' = ''valor1'' ,''columna2'' = ''valor2'',...
WHERE ''columnaN'' = ''valorN''

Ejemplo

UPDATE My_table SET field1 = 'updated value' WHERE field2 = 'N';

DELETE

Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla,

Forma básica

DELETE FROM ''tabla'' WHERE ''columna1'' = ''valor1''

Ejemplo

DELETE FROM My_table WHERE field2 = 'N';

Ejercicios con el Lenguaje SQL

Observe el código de los archivos basics.sql y advanced.sql, una vez observado siga las instrucciones que se indican a continuación.

Ejercicio 1

Siguiendo la estructura del comando **CREATE TABLE** cree las tablas que diseñó para su base de datos de CDs o DVDs.

Ejercicio 2

Utilizando la instrucción **INSERT** agregue al menos 3 registros para su colección.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Ejercicio 3

Utilice la instrucción **SELECT** para listar todos los registros de una tabla.

Ejercicio 4

Utilice la instrucción **SELECT** para listar solamente las películas prestadas.

Ejercicio 5

Utilice la instrucción **SELECT** para listar cuánto tiempo tiene prestada una película.

Ejercicio 6

Investigue sobre el funcionamiento de la opción **DISTINCT** dentro de la instrucción **SELECT**.

Ejercicio 7

Con la instrucción **SELECT** haga la unión de dos tablas.

Ejercicio 8

Utilizando la opción **JOIN** haga de nuevo el ejercicio anterior. Investigue el funcionamiento de esta opción en la documentación de PostgreSQL.

Ejercicio 9

Elimine un registro con la instrucción **DELETE**.

Ejercicio 10

Ejecute la instrucción **TRUNCATE** a una tabla.

Ejercicio 11

Borre una tabla con la instrucción **DROP TABLE**.

Ejercicio 12

Diseñe nuevamente su base de datos de CDs y DVDs con tablas relacionadas entre si utilizando llaves foráneas.

Administración de PostgreSQL

PostgreSQL tiene distintos ámbitos dentro de los cuales se deben administrar recursos. Entre ellos se encuentran la gestión referente a:

- Espacio en Disco
- Respaldos
- Usuarios
- Conexiones

Gestión de Espacio Físico

Una de las formas de controlar el uso del disco duro por parte de PosgreSQL es creando TABLESPACES. Es así como direccionaremos los archivos de nuestras bases de datos a espacios precisos dentro de nuestro sistema de archivos. Para crear un TABLESPACE debemos ser superusuario dentro de nuestra base de datos y usar el siguiente comando:

postgres=# \help create tablespace

Orden: CREATE TABLESPACE

Descripción: define un nuevo tablespace

Sintaxis:
CREATE TABLESPACE nombre_de_tablespace [OWNER nombre_de_usuario] LOCATION 'directorio'
postgres=#

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Por ejemplo:
postgres=# CREATE TABLESPACE arreglo1 LOCATION /data/disco1

Una vez tengamos esto listo el observar el uso del disco será simplemente utilizar el comando df o du según sea su preferencia.

Otro modo de calcular el peso de una tabla es, luego de el proceso de VACUUM, consultar el número de páginas que ocupa la misma. Típicamente una página consta de 8 kilobytes, de esta manera calcularemos el tamaño de determinada tabla:
postgres=# SELECT relfilenode, relpages FROM pg_class WHERE relname = '[tabla a consultar]';

El proceso de hacer VACUUM es importante debido a que la variable *relpages* se actualiza solamente con algunas instrucciones. Para más información sobre el comando VACUUM y el uso de disco vea la sección **23.1. Routine Vacuuming** del manual de PostgreSQL.

Gestión de Respaldos

Respaldo de bajo nivel

La idea de este tipo de respaldos es guardar todos los archivos de la carpeta de la base de datos. Uno de los métodos es usando el comando **tar**.
$ tar -cf respaldo.tar /var/lib/postgresql/8.3/main/

Sin embargo, hay dos restricciones que hay que tomar en cuenta:

1. El servidor **DEBE** estar detenido para que el respaldo sea usable. Medidas como deshabilitar conexiones simplemente no funcionan.
2. Conociendo la estructura de los archivos puede verse tentado a respaldar solamente unas tablas seleccionadas. Esto no funcionará debido a que la información contenida en esos archivos es solo la mitad de los datos. La otra mitad está en los archivos de “commit logs”. Una tabla es solo usable con estos archivos.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Respaldar una Base de Datos

Para hacer el respaldo de una base de datos y generar de salida un archivo de texto plano con instrucciones que reconstruya el estado de la misma, para ello utilizamos el comando **pg_dump**.

Ejemplo:

$ pg_dump [base de datos] > [archivo de salida]

Si desea recuperar los datos de su respaldo deberá crear una base de datos y posteriormente ejecutar el archivo de respaldo. El proceso será similar a lo que se indica a continuación:

$ createdb -T template0 [base de datos]
$ psql [base de datos] < [archivo de entrada]

Para más información del funcionamiento de este comando recordemos usar la opción, *–help* del comando.

```
$ pg_dump --help
pg_dump dumps a database as a text file or to other formats.

Usage:
  pg_dump [OPTION]... [DBNAME]

General options:
  -f, --file=FILENAME        output file name
  -F, --format=c|t|p         output file format (custom, tar, plain text)
  -v, --verbose              verbose mode
  -Z, --compress=0-9         compression level for compressed formats
  --lock-wait-timeout=TIMEOUT fail after waiting TIMEOUT for a table lock
  --help                     show this help, then exit
  --version                  output version information, then exit

Options controlling the output content:
  -a, --data-only            dump only the data, not the schema
  -b, --blobs                include large objects in dump
  -c, --clean                clean (drop) database objects before recreating
  -C, --create               include commands to create database in dump
  -E, --encoding=ENCODING    dump the data in encoding ENCODING
```

```
  -n, --schema=SCHEMA         dump the named schema(s) only
  -N, --exclude-schema=SCHEMA do NOT dump the named schema(s)
  -o, --oids                  include OIDs in dump
  -O, --no-owner              skip restoration of object ownership in
                              plain-text format
  -s, --schema-only           dump only the schema, no data
  -S, --superuser=NAME        superuser user name to use in plain-text format
  -t, --table=TABLE           dump the named table(s) only
  -T, --exclude-table=TABLE   do NOT dump the named table(s)
  -x, --no-privileges         do not dump privileges (grant/revoke)
  --binary-upgrade            for use by upgrade utilities only
  --inserts                   dump data as INSERT commands, rather than COPY
  --column-inserts            dump data as INSERT commands with column names
  --disable-dollar-quoting    disable dollar quoting, use SQL standard quoting
  --disable-triggers          disable triggers during data-only restore
  --no-tablespaces            do not dump tablespace assignments
  --role=ROLENAME             do SET ROLE before dump
  --use-set-session-authorization
                              use SET SESSION AUTHORIZATION commands instead of
                              ALTER OWNER commands to set ownership

Connection options:
  -h, --host=HOSTNAME      database server host or socket directory
  -p, --port=PORT          database server port number
  -U, --username=NAME      connect as specified database user
  -w, --no-password        never prompt for password
  -W, --password           force password prompt (should happen automatically)

If no database name is supplied, then the PGDATABASE environment
variable value is used.

Report bugs to <pgsql-bugs@postgresql.org>.
$
```

Para trasladar una base de datos de un servidor a otro podemos utilizar este comando de la forma en que se indica:

```
$ pg_dump -h [servidor 1] [base de datos] | psql -h [servidor 2] [base de datos]
```

Respaldar todas las Bases de Datos

Similar al comando anterior el comando **pg_dumpall** realiza un respaldo de los

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

datos de todas las bases de datos del servidor. Cuando utilizamos este comando, a diferencia del comando **pg_dump**, respaldamos también los datos como los roles de usuarios y espacios de tablas.
El proceso para respaldar con **pg_dumpall** será como se indica:
$ pg_dumpall > [archivo de salida]
El resultado de este respaldo puede ser recuperado con el comando:
$ psql -f [archivo de entrada] postgres
Para más detalles sobre el comando veamos la ayuda:
$ pg_dumpall --help
pg_dumpall extracts a PostgreSQL database cluster into an SQL script file.

Usage:
  pg_dumpall [OPTION]...

General options:
  -f, --file=FILENAME   output file name
  --lock-wait-timeout=TIMEOUT fail after waiting TIMEOUT for a table lock
  --help   show this help, then exit
  --version   output version information, then exit

Options controlling the output content:
  -a, --data-only   dump only the data, not the schema
  -c, --clean   clean (drop) databases before recreating
  -g, --globals-only   dump only global objects, no databases
  -o, --oids   include OIDs in dump
  -O, --no-owner   skip restoration of object ownership
  -r, --roles-only   dump only roles, no databases or tablespaces
  -s, --schema-only   dump only the schema, no data
  -S, --superuser=NAME   superuser user name to use in the dump
  -t, --tablespaces-only   dump only tablespaces, no databases or roles
  -x, --no-privileges   do not dump privileges (grant/revoke)
  --binary-upgrade   for use by upgrade utilities only
  --inserts   dump data as INSERT commands, rather than COPY
  --column-inserts   dump data as INSERT commands with column names
  --disable-dollar-quoting   disable dollar quoting, use SQL standard quoting
  --disable-triggers   disable triggers during data-only restore
  --no-tablespaces   do not dump tablespace assignments
  --role=ROLENAME   do SET ROLE before dump
  --use-set-session-authorization
                             use SET SESSION AUTHORIZATION commands instead of
                             ALTER OWNER commands to set ownership

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
Connection options:
  -h, --host=HOSTNAME      database server host or socket directory
  -l, --database=DBNAME    alternative default database
  -p, --port=PORT          database server port number
  -U, --username=NAME      connect as specified database user
  -w, --no-password        never prompt for password
  -W, --password           force password prompt (should happen automatically)

If -f/--file is not used, then the SQL script will be written to the standard
output.

Report bugs to <pgsql-bugs@postgresql.org>.
$
```

Tratando con bases de datos de gran tamaño

El exportar los datos de una base de datos a formato de texto plano generalmente crea
archivos de gran tamaño, ahora bien, en el caso de las bases de datos de gran tamaño esto se vuelve engorroso. Es por ello que recurrimos a técnicas de compresión para gestionar estos archivos, como se ilustra a continuación:

```
$ pg_dump [base de datos] | gzip > [archivo comprimido].gz
```

y se puede recuperar la información mediante:

```
$ gunzip -c [archivo comprimido].gz | psql [base de datos]
```

o a través del siguiente comando:

```
$ cat [archivo comprimido].gz | gunzip | psql [base de datos]
```

Otra técnica también utilizada es el cortar los archivos de gran tamaño en piezas más pequeñas. Para ello utilizamos el comando **split**:

```
$ pg_dump [base de datos] | split -b [tamaño] - [archivo]
```

y recuperamos la información mediante:

```
$ cat [archivo]* | psql [base de datos]
```

Adicionalmente podemos hacer que el comando pg_dump nos genere una salida personalizada. Dado que PostgreSQL trae incluida la librería de compresión zlib los respaldos son comprimidos con la misma en un archivo de salida. Esto genera una

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

salida de tamaño similar al obtenido con gzip.

$ pg_dump -Fc [base de datos] > [archivo]

Dado que el formato de salida no es un archivo apto para **psql** debemos utilizar el comando **pg_restore** para restaurar estos datos.

$ pg_restore -d [base de datos] [archivo]

Para más detalle veremos la ayuda para el comando **pg_restore**:

$ pg_restore

```
Restablece una base de datos de PostgreSQL usando un archivo
creado por pg_dump.
Uso:
  pg_restore [OPCIÓN]... [ARCHIVO]

Opciones generales:
  -d, --dbname=NOMBRE      nombre de la base de datos a la que conectarse
  -f, --file=ARCHIVO       nombre del archivo de salida
  -F, --format=c|t         formato del volcado (debería ser automático)
  -l, --list               imprime una tabla resumida de contenidos
                           del archivador
  -v, --verbose            modo verboso
  --help                   muestra esta ayuda y termina
  --version                muestra información de la versión y termina

Opciones que controlan la recuperación:
  -a, --data-only          restablece sólo los datos, no el esquema
  -c, --clean              tira (drop) la base de datos antes de recrearla
  -C, --create             crea la base de datos de destino
  -e, --exit-on-error      abandonar al encontrar un error
                           por omisión, se continúa la restauración
  -I, --index=NOMBRE       restablece el índice nombrado
  -j, --jobs=NUM           máximo de procesos paralelos para restaurar
  -L, --use-list=ARCHIVO   usa la tabla de contenido especificada para ordenar
                           la salida de este archivo
  -n, --schema=NAME        restablece sólo los objetos en este esquema
  -O, --no-owner           no restablece los dueños de los objetos
  -P, --function=NOMBRE(args)
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
                        restablece la función nombrada
  -s, --schema-only       restablece el esquema únicamente, no los datos

  -S, --superuser=NOMBRE  especifica el nombre del superusuario que se usa
                        para deshabilitar los disparadores (triggers)
  -t, --table=NOMBRE      restablece la tabla nombrada
  -T, --trigger=NOMBRE    restablece el disparador (trigger) nombrado
  -x, --no-privileges     no restablece los privilegios (grant/revoke)
  --disable-triggers      deshabilita los disparadores (triggers) durante el
                        restablecimiento sólo de datos
  --no-data-for-failed-tables
                        no restablece datos de tablas que no pudieron
                        ser creadas
  --no-tablespaces        no vuelca asignaciones de tablespace
  --role=ROLENAME         hace SET ROLE antes de restaurar
  --use-set-session-authorization
                        usa órdenes SET SESSION AUTHORIZATION en lugar de
                        ALTER OWNER para restablecer dueños
  -1, --single-transaction restablece en una única transacción

Opciones de la conexión:
  -h, --host=ANFITRIÓN    anfitrión de la base de datos o
                        directorio del enchufe (socket)
  -p, --port=PUERTO       número del puerto de la base de datos
  -U, --username=USUARIO  nombre de usuario con el cual conectarse
  -w, --no-password       nunca pedir una contraseña
  -W, --password          fuerza un prompt para la contraseña
                        (debería ser automático)

Si no se especifica un archivo de entrada, se usa la entrada estándar.

Reporta errores a <pgsql-bugs@postgresql.org>.
$
```

Gestión de Usuarios

Dentro de PostgreSQL los usuarios son manejados por perfiles (o Roles) que definen las capacidades y permisos otorgados para que una persona acceda a los distintos espacios de la base de datos.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Crear y borrar usuarios

Cree un usuario con el comando
$ createuser [nombre]

y borre el usuario con el comando
$ dropuser [nombre]

Ejecute el mismo ejercicio anterior mediante las instrucciones SQL **CREATE ROLE** y **DROP ROLE**

Ver usuarios en el servidor

Examine los usuarios disponibles en su base de datos mediante la consulta
SELECT rolname FROM pg_roles;

CREATE ROLE vs. CREATE USER

Examine las diferencias entre las siguientes instrucciones:
CREATE ROLE [nombre]
y
CREATE USER [nombre]

Atributos de los Roles

Examine los atributos de Roles

- LOGIN
- SUPERUSER
- CREATEDB
- CREATEROLE
- PASSWORD

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Restaurar seguridad de PostgreSQL

Defina una clave para el usuario **postgres** de su base de datos. Posteriormente modifique su archivo **pg_hba.conf** para que PostgreSQL valide contraseñas con el algoritmo md5. Intente acceder nuevamente a la base de datos desde pgAdmin3.

Privilegios

Examine los comandos **GRANT** y **REVOKE**, comente con el instructor en que consisten los privilegios.

```
postgres=#\h GRANT
Command:     GRANT
Description: define access privileges

Syntax:
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES |
TRIGGER }
    [,...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column [, ...] )

    [,...] | ALL [ PRIVILEGES ] ( column [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [,...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE sequence_name [, ...]
         | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
    ON DATABASE database_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON FOREIGN DATA WRAPPER fdw_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON FOREIGN SERVER server_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
     ON { FUNCTION function_name ( [ [ argmode ] [ arg_name ] arg_type [, ...] ] )
[, ...]
         | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE lang_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | UPDATE } [,...] | ALL [ PRIVILEGES ] }
    ON LARGE OBJECT loid [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespace_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT role_name [, ...] TO role_name [, ...] [ WITH ADMIN OPTION ]
postgres=#\h REVOKE
Command:     REVOKE
Description: remove access privileges
Syntax:
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    [,...] | ALL [ PRIVILEGES ] }

    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { SELECT | INSERT | UPDATE | REFERENCES } ( column [, ...] )
   [,...] | ALL [ PRIVILEGES ] ( column [, ...] ) }
   ON [ TABLE ] table_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { USAGE | SELECT | UPDATE }
   [,...] | ALL [ PRIVILEGES ] }
   ON { SEQUENCE sequence_name [, ...]
       | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { CREATE | CONNECT | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
   ON DATABASE database_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { USAGE | ALL [ PRIVILEGES ] }
   ON FOREIGN DATA WRAPPER fdw_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { USAGE | ALL [ PRIVILEGES ] }
   ON FOREIGN SERVER server_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argmode ] [ arg_name ] arg_type [, ...] ] )
[, ...]
       | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { USAGE | ALL [ PRIVILEGES ] }
   ON LANGUAGE lang_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]

   { { SELECT | UPDATE } [,...] | ALL [ PRIVILEGES ] }
   ON LARGE OBJECT loid [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
   ON SCHEMA schema_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { CREATE | ALL [ PRIVILEGES ] }
   ON TABLESPACE tablespace_name [, ...]
   FROM { [ GROUP ] role_name | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ ADMIN OPTION FOR ]
   role_name [, ...] FROM role_name [, ...]
   [ CASCADE | RESTRICT ]
```

Gestión de Conexiones

Para acceder a PostgreSQL, y debido a que es una arquitectura cliente servidor, se requiere un mecanismo de conexión. Para conexiones locales en sistemas *NIX este mecanismo se conoce como sockets. De otra forma las conexiones al gestor de bases de datos se hace mediante la red en sus distintos protocolos (IPv4 e

IPv6).

Para habilitar conexiones hacemos uso de la sección correspondiente en el archivo postgres.conf:

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

#listen_addresses = 'localhost'            # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost', '*' = all
                                # (change requires restart)
port = 5432                             # (change requires restart)
max_connections = 100                   # (change requires restart)
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).  You might
# also need to raise shared_buffers to support more connections.
#superuser_reserved_connections = 3  # (change requires restart)
unix_socket_directory = '/var/run/postgresql'               # (change requires restart)
#unix_socket_group = ''                 # (change requires restart)
#unix_socket_permissions = 0777             # begin with 0 to use octal notation

                                # (change requires restart)
#bonjour_name = ''                  # defaults to the computer name
                                # (change requires restart)

# - Security and Authentication -

#authentication_timeout = 1min              # 1s-600s
ssl = true                      # (change requires restart)
#ssl_ciphers = 'ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH'   # allowed SSL ciphers
                                # (change requires restart)
#ssl_renegotiation_limit = 512MB     # amount of data between renegotiations
#password_encryption = on
#db_user_namespace = off

# Kerberos and GSSAPI
#krb_server_keyfile = ''            # (change requires restart)
#krb_srvname = 'postgres'           # (change requires restart, Kerberos only)
```

```
#krb_server_hostname = ''           # empty string matches any keytab entry
                                    # (change requires restart, Kerberos only)
#krb_caseins_users = off            # (change requires restart)
#krb_realm = ''                     # (change requires restart)

# - TCP Keepalives -
# see "man 7 tcp" for details

#tcp_keepalives_idle = 0            # TCP_KEEPIDLE, in seconds;
                                    # 0 selects the system default
#tcp_keepalives_interval = 0        # TCP_KEEPINTVL, in seconds;
                                    # 0 selects the system default
#tcp_keepalives_count = 0           # TCP_KEEPCNT;
                                    # 0 selects the system default
```

Sin embargo la gestión de las conexiones a distintos equipos de la red se realiza es el archivo pg_hba.conf. El uso de este archivo se detalla en la sección **21.1. The pg_hba.conf file** del manual. veamos algunas consideraciones.

El archivo **pg_hba.conf** está estructurado de en 5 columnas que son a saber:

- TYPE
- DATABASE
- USER
- CIDR-ADDRESS
- METHOD

La columna *TYPE* trata del tipo de conexión a servidores, sus opciones son:

- *local*, para conexiones vía sockets en sistemas de UNIX
- *host*, conexiones vía TCP/IP a servidores
- *hostssl*, como el anterior pero sólo habilitando servidores con capacidad de incriptación.
- *hostnossl*, al contrario del anterior sólo habilita los servidores sin incriptación.

La columna *DATABASE* se refiere a las bases de datos que están habilitadas en la definición de permisos. Así mismo *USER* habilita las conexiones para determinados usuarios o grupos de usuarios.

CIDR-ADDRESS se refiere a las direcciones IP de los hosts que pueden conectarse al servidor. Se expresa de forma de IP/máscara, donde típicamente construimos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

direcciones así:

- Máscara 255.0.0.0 la representamos mediante 8
- Máscara 255.255.0.0 la representamos mediante 16
- Máscara 255.255.255.0 la representamos mediante 24
- Máscara 255.255.255.255 la representamos mediante 32

Finalmente *METHOD* se refiere a la forma en que vamos a permitir la autenticación de las conexiones a nuestro servidor. Los métodos que PostgreSQL tiene para este fin son:

- trust
- reject
- md5
- crypt
- password
- gss
- sspi
- krb5
- ident
- ldap
- pam

Adicional a esto puede agregarse una columna más para las opciones del método de autenticación.

Ejemplo:

```
# Allow any user on the local system to connect to any database under
# any database user name using Unix-domain sockets (the default for local
# connections).
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local   all         all                               trust

# The same using local loopback TCP/IP connections.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    all         all         127.0.0.1/32          trust

# The same as the last line but using a separate netmask column
#
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
# TYPE  DATABASE    USER        IP-ADDRESS    IP-MASK           METHOD
host    all         all         127.0.0.1     255.255.255.255   trust

# Allow any user from any host with IP address 192.168.93.x to connect
# to database "postgres" as the same user name that ident reports for
# the connection (typically the Unix user name).
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    postgres    all         192.168.93.0/24       ident sameuser

# Allow a user from host 192.168.12.10 to connect to database
# "postgres" if the user's password is correctly supplied.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    postgres    all         192.168.12.10/32      md5

# In the absence of preceding "host" lines, these two lines will
# reject all connection from 192.168.54.1 (since that entry will be
# matched first), but allow Kerberos 5 connections from anywhere else
# on the Internet.  The zero mask means that no bits of the host IP
# address are considered so it matches any host.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    all         all         192.168.54.1/32       reject
host    all         all         0.0.0.0/0             krb5

# Allow users from 192.168.x.x hosts to connect to any database, if
# they pass the ident check.  If, for example, ident says the user is
# "bryanh" and he requests to connect as PostgreSQL user "guest1", the
# connection is allowed if there is an entry in pg_ident.conf for map
# "omicron" that says "bryanh" is allowed to connect as "guest1".
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    all         all         192.168.0.0/16        ident omicron

# If these are the only three lines for local connections, they will
# allow local users to connect only to their own databases (databases
# with the same name as their database user name) except for administrators
# and members of role "support", who can connect to all databases.  The file
# $PGDATA/admins contains a list of names of administrators.  Passwords
# are required in all cases.
```

```
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local   sameuser    all                               md5
local   all         @admins                           md5
local   all         +support                          md5

# The last two lines above can be combined into a single line:
local   all         @admins,+support                  md5

# The database column can also use lists and file names:
local   db1,db2,@demodbs  all                         md5
```

PROGRAMANDO EN POSTGRESQL

Funciones con SQL

Las funciones SQL ejecutan una lista arbitraria de comandos SQL y devuelven el resultado de la última consulta en una lista. El resultado será la primera fila de la última consulta, siendo que si la misma no devuelve registros la función devolverá NULL y si se desea devolver una lista de resultados hay que indicar la opción SETOF tipo_de_datos.

Puede utilizar cualquier comando, como lo son SELECT, INSERT, UPDATE y DELETE. Sin embargo no se pueden utilizar comandos referentes a estados de transacciones como son BEGIN, COMMIT, ROLLBACK o SAVEPOINT.

Observemos un ejemplo

```
CREATE FUNCTION limpia_empleados() RETURNS void AS '
    DELETE FROM empleados WHERE salario < 0;
' LANGUAGE SQL;

SELECT limpia_empleados();
```

El ejemplo más sencillo es una función que no tome argumentos y devuelva un tipo de datos básico:

```
CREATE FUNCTION uno() RETURNS integer AS $$
    SELECT 1 AS result;
$$ LANGUAGE SQL;

-- Sintaxis alternativa utilizando cadena de caracteres:

CREATE FUNCTION uno() RETURNS integer AS '
    SELECT 1 AS result;
' LANGUAGE SQL;

SELECT uno();
```

Note que podemos colocar un alias a la columna de resultados como se muestra a continuación:

```
CREATE FUNCTION sumalos(integer, integer) RETURNS integer AS $$
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
   SELECT $1 + $2;
$$ LANGUAGE SQL;
```

SELECT sumalos(1, 2) AS respuesta;
Verifique en el capítulo **9. Funciones y Operadores** de la documentación las funciones matemáticas, estadísticas, de texto, fechas, etcétera con las que cuenta para utilizar este tipo de funciones.

Podría interesarle actualizar un dato en una tabla, como en el siguiente ejemplo. Observe el funcionamiento de esta función.

```
CREATE FUNCTION retiro (integer, numeric) RETURNS integer AS $$
   UPDATE banco
   SET balance = balance - $2
   WHERE cuenta = $1;
   SELECT balance FROM banco WHERE cuenta = $1;
$$ LANGUAGE SQL;
```

Así mismo a las funciones podemos definirles parámetros de salida como se muestra a continuación:

```
CREATE FUNCTION sumalos(IN x int, IN y int, OUT sum int)
   AS 'SELECT $1 + $2'
LANGUAGE SQL;
```

Sin embargo la verdadera potencia de esta opción es proveer de una forma conveniente varias columnas de una vez. Vea el siguiente ejemplo:

```
CREATE FUNCTION suma_y_producto (x int, y int, OUT sum int, OUT product int)
   AS 'SELECT $1 + $2, $1 * $2'
LANGUAGE SQL;
```

```
SELECT * FROM suma_y_producto(11,42);
  sum | product
 -----+---------
   53 |     462
```

Cuando se desea que una función devuelva un grupo de datos debe definirse la misma con la instrucción SETOF definiendo un tipo de datos. Veamos un ejemplo:

```
CREATE FUNCTION getfoo(int) RETURNS SETOF foo AS $$
   SELECT * FROM foo WHERE fooid = $1;
```

```
$$ LANGUAGE SQL;

SELECT * FROM getfoo(1) AS t1;
      fooid | foosubid | fooname
     -------+----------+---------
          1 |        1 | Joe
          1 |        2 | Ed
 (2 rows)
```

Sobrecarga de Funciones

Más de una función puede ser definida con el mismo nombre SQL siempre que los argumentos que lleven sean distintos. Cuando se ejecuta la consulta el servidor determinará cual función llamará dependiendo de los tipos de datos y el número de argumentos provistos.

CREATE FUNCTION test(int, real) RETURNS ...
CREATE FUNCTION test(int, real, real) RETURNS ...
Sin embargo debe tenerse cuidado de no hacer definiciones que sean ambiguas como las que se muestran a continuación:
CREATE FUNCTION test(int, real) RETURNS ...
CREATE FUNCTION test(smallint, double precision) RETURNS ...

Lenguajes Procedimentales

PostgreSQL permite escribir funciones definidas por el usuario escritos en otros lenguajes además de SQL y C. Estos lenguajes generalmente se conocen como **Lenguajes Procedimentales**. Para una función escrita en un lenguaje procedimental el servidor de base de datos delega en un manejador (handler) la interpretación de la lógica del programa escrito. El *handler* es una función en Lenguaje C que permite bajo demanda hacer la conversión, análisis sintáctico, ejecución, etcétera del programa escrito por el usuario.

PostgreSQL pone a disposición de forma estándar los siguientes lenguajes procedimentales:

- PL/pgSQL
- PL/Tcl

- PL/Perl
- PL/Python

Y adicionalmente posee otros lenguajes como lo son:

- PL/Java
- PL/PHP
- PL/Py
- PL/R
- PL/Ruby
- PL/Scheme
- PL/sh

PL/pgSQL

PL/pgSQL es un lenguaje cargable que tiene por objetivo;

- Crear funciones y disparadores
- Agregar estructuras de control al lenguaje SQL
- Ejecutar cálculos complejos
- Heredar todos los tipos de datos definidos por usuarios, funciones y operadores
- Definido como confiable por el servidor
- Fácil de usar

Entre las ventajas de usar PL/pgSQL están que a diferencia del SQL donde cada comando debe ser ejecutado individualmente, con este lenguaje procedimental es posible condensar bloques que se ejecutan dentro del servidor. Esto significa que las se generan grandes ahorros en comunicaciones cliente/servidor de modo que,

- Comunicaciones extras entre servidor y cliente son eliminadas
- Cálculos intermedios que no son del interés del cliente no son transferidos sino usados dentro del servidor
- Muchas sesiones de consultas de datos pueden ser evitadas

Instalando PL/pgSQL

Una vez creada una base de datos el proceso de instalar el lenguaje

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

procedimental de PostgreSQL (PL/pgSQL) puede realizarse desde la un terminal mediante el comando

$ createlang [lenguaje] [base de datos]

donde en el lenguaje se colocará **plpgsql** y el nombre de su base de datos. O dentro de la base de datos escribimos

CREATE LANGUAGE plpgsql;

Existen otros lenguajes procedimentales como son PL/Python, PL/Tcl y PL/Perl. Dependiendo del caso podrá usar uno u otro según sus habilidades y conveniencia.

Estructura de programas con PL/pgSQL

PL/pgSQL es un lenguaje de bloques por lo que un programa en este se estructura de la siguiente manera:

```
[ << ETIQUETA >> ]
[ DECLARE
      declaraciones ]
BEGIN
      instrucciones
END;
```

Para escribir comentarios dentro de un programa de PL/pgSQL hay 2 formas:

- para una sola línea, mediante un doble guión (–)
- para un bloque, mediante las señas de apertura y cierre (/* y */)

Veamos un ejemplo de una función en PL/pgSQL

```
CREATE FUNCTION somefunc() RETURNS integer AS $$
<< outerblock >>

DECLARE
   quantity integer := 30;
BEGIN
   RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 30
   quantity := 50;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
   --
   -- Create a subblock
   --
   DECLARE
      quantity integer := 80;
   BEGIN
      RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 80
      RAISE NOTICE 'Outer quantity here is %', outerblock.quantity;  -- Prints 50
   END;
```

PL/pgSQL es un lenguaje de bloques por lo que un programa en este se estructura de la siguiente manera:

```
[ << ETIQUETA >> ]
[ DECLARE
      declaraciones ]
BEGIN
      instrucciones
END;
```

Para escribir comentarios dentro de un programa de PL/pgSQL hay 2 formas:

- para una sola línea, mediante un doble guión (–)
- para un bloque, mediante las señas de apertura y cierre (/* y */)

Veamos un ejemplo de una función en PL/pgSQL

```
CREATE FUNCTION somefunc() RETURNS integer AS $$
<< outerblock >>

DECLARE
   quantity integer := 30;
BEGIN
   RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 30
   quantity := 50;
   --
   -- Create a subblock
   --
   DECLARE
      quantity integer := 80;
   BEGIN
      RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 80
      RAISE NOTICE 'Outer quantity here is %', outerblock.quant

   RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 50
```

```
   RETURN quantity;
END;
$$ LANGUAGE plpgsql;
```

Nota: No confundir las instrucciones BEGIN/END con los equivalentes de transacciones, en PL/pgSQL se refieren a agrupar otras instrucciones.

Declaraciones

Todas las variables usadas en un bloque de instrucciones deben ser declaradas en la sección de declaraciones, con la única excepción de la instrucción **FOR**. Las variables pueden ser cualquier tipo de datos SQL como *integer*, *varchar*, *char*, etcétera.

Algunos ejemplos serían:

```
user_id integer;
cantidad numeric(5);
url varchar;
mi_registro tablename%ROWTYPE;
mi_campo tablename.columnname%TYPE;
un_registro RECORD;
cantidad2 integer DEFAULT 32;
url varchar := 'http://mysite.com';
user_id CONSTANT integer := 10;
```

La sintaxis general de las declaraciones es:

```
nombre [ CONSTANT ] tipo_de_dato [ NOT NULL ] [ { DEFAULT | := } expresión ];
```

Alias para Parámetros de una Función

Existen 2 formas de colocar alias a los parámetros de una función, la primera es colocando en la declaración de la función el nombre del alias,

```
CREATE FUNCTION iva(subtotal real) RETURNS real AS $$
BEGIN
   RETURN subtotal * 0.12;
END;
$$ LANGUAGE plpgsql;
```

O utilizando en la zona de declaraciones la forma que se indica,
nombre ALIAS FOR $n;
vea el ejemplo:

```
CREATE FUNCTION iva(real) RETURNS real AS $$
DECLARE
   subtotal ALIAS FOR $1;
BEGIN
   RETURN subtotal * 0.12;
END;
$$ LANGUAGE plpgsql;
```

otra forma de escribir esta función es:

```
CREATE FUNCTION iva(subtotal real, OUT impuesto real) AS $$
BEGIN
   impuesto := subtotal * 0.12;
END;
$$ LANGUAGE plpgsql;
```

Expresiones

Cuando PL/pgSQL recibe una expresión como:
IF expresion THEN
la evalúa internamente como:
SELECT expresion
donde la expresión podrán ser del tipo $a < b$, $a = b$, etcétera.

Declaraciones

- Asignación:

```
impuesto := subtotal * 0.12;
mi_registro.user_id := 20;
```

- Ejecutando Comandos sin Resultados:

```
DECLARE
   key TEXT;
   delta INTEGER;
BEGIN
   ...
   UPDATE mytab SET val = val + delta WHERE id = key;
```

- Ejecutando Comandos con una sola fila como resultado:

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

SELECT select_expressions INTO [STRICT] target FROM ...;
INSERT ... RETURNING expressions INTO [STRICT] target;
UPDATE ... RETURNING expressions INTO [STRICT] target;
DELETE ... RETURNING expressions INTO [STRICT] target;

Si no se especifica la opción **STRICT** devolverá la primera fila de los registros seleccionados.

Ejemplo:

```
BEGIN
  SELECT * INTO STRICT myrec FROM emp WHERE nombre = mi_nombre;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
       RAISE EXCEPTION 'employee % not found', mi_nombre;
    WHEN TOO_MANY_ROWS THEN
      RAISE EXCEPTION 'employee % not unique', mi_nombre;
END;
```

- Ejecutar comandos dinámicamente

```
EXECUTE comando-texto [ INTO [STRICT] destino ];
```

Ejemplo:

```
EXECUTE 'UPDATE tbl SET '
    quote_ident(colname)
    ' = $$'
    newvalue
    '$$ WHERE key = '
    quote_literal(keyvalue);
```

- Obteniendo resultados de estado

```
GET DIAGNOSTICS variable = item [ , ... ];
```

Ejemplo:

```
GET DIAGNOSTICS var = ROW_COUNT;
```

- Hacer nada

```
NULL;
```

Ejemplo:

```
BEGIN
    y := x / 0;
 EXCEPTION
    WHEN division_by_zero THEN
       NULL;  -- ignore the error
END;
```

o

```
BEGIN
    y := x / 0;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
  EXCEPTION
    WHEN division_by_zero THEN  -- ignore the error
END;
```

Estructuras de Control

Regresando valores de una función
RETURN expresión;
Otros tipos de retorno de valores son *RETURN NEXT* o *RETURN QUERY consulta*, veamos un ejemplo.

```
CREATE TABLE municipios (
   id INTEGER,
   subid INTEGER,
   nombre TEXT
);

INSERT INTO municipios VALUES (1, 1, 'Distrito Capital');
INSERT INTO municipios VALUES (5, 3, 'Brion');
INSERT INTO municipios VALUES (5, 7, 'Los Salias');

CREATE OR REPLACE FUNCTION tomaMunicipios() RETURNS SETOF municipios AS
$BODY$
DECLARE
  r municipios%rowtype;
BEGIN
  FOR r IN SELECT * FROM municipios
  WHERE id > 0
  LOOP
    -- algún proceso puede ser realizado
    RETURN NEXT r; -- devuelve el valor actual del SELECT
  END LOOP;
  RETURN;
END

$BODY$
LANGUAGE 'plpgsql' ;

SELECT * FROM tomamunicipios();
```

Condicionales:

- IF ... THEN

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
IF boolean-expression THEN
  statements
END IF;
```

- IF ... THEN ... ELSE

```
IF boolean-expression THEN
  statements
ELSE
  statements
END IF;
```

- IF ... THEN ... ELSE IF

Ejemplo:

```
IF demo_row.sex = 'm' THEN
  pretty_sex := 'man';
ELSE
  IF demo_row.sex = 'f' THEN
    pretty_sex := 'woman';
  END IF;
END IF;
```

- IF ... THEN ... ELSIF ... THEN ... ELSE

Ejemplo:

```
IF number = 0 THEN
  result := 'cero';
ELSIF number > 0 THEN
  result := 'positivo';
ELSIF number < 0 THEN
  result := 'negativ0';
ELSE
  -- hmm, solo hay otra posibilidad y es que el numero sea null
  result := 'NULL';
END IF;
```

- IF ... THEN ... ELSEIF ... THEN ... ELSE

ELSEIF es un alias para ELSIF

Bucles simples

Definición:

```
[ <<label>> ]
LOOP
   statements
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
END LOOP [ label ];
```

Ejemplos:

```
-- ejemplo 1
LOOP
  -- algunos cálculos
  IF count > 0 THEN
    EXIT;  -- exit loop
  END IF;
END LOOP;

-- ejemplo 2
LOOP
   -- some algunos cálculos
  EXIT WHEN count > 0;  -- same result as previous example
END LOOP;

-- ejemplo 3
BEGIN
  -- some computations
  IF stocks > 100000 THEN
    EXIT;  -- causes exit from the BEGIN block
  END IF;
END;
```

Bucles con la instrucción **FOR**

```
FOR i IN 1..10 LOOP
  -- i tomara los valores 1,2,3,4,5,6,7,8,9,10
END LOOP;

FOR i IN REVERSE 10..1 LOOP
   -- i tomara los valores 10,9,8,7,6,5,4,3,2,1
END LOOP;

FOR i IN REVERSE 10..1 BY 2 LOOP
  -- i tomara los valores 10,8,6,4,2
END LOOP;
```

Haciendo bucles a través de consultas.

Sintaxis:

```
[ <<label>> ]
FOR target IN query LOOP
  statements
END LOOP [ label ];
```

Ejemplo:

```
CREATE FUNCTION cs_refresh_mviews() RETURNS integer AS $$
DECLARE
  mviews RECORD;
BEGIN
  PERFORM cs_log('Refreshing materialized views...');

  FOR mviews IN SELECT * FROM cs_materialized_views ORDER BY sort_key LOOP

    -- Now "mviews" has one record from cs_materialized_views

                    PERFORM cs_log('Refreshing materialized view ' ||
quote_ident(mviews.mv_name) || ' ...');
    EXECUTE 'TRUNCATE TABLE ' || quote_ident(mviews.mv_name);
          EXECUTE 'INSERT INTO ' || quote_ident(mviews.mv_name) || ' ' ||
mviews.mv_query;
  END LOOP;

  PERFORM cs_log('Done refreshing materialized views.');
  RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

Capturando errores

Sintaxis:

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  WHEN condition [ OR condition ... ] THEN
    handler_statements
  [ WHEN condition [ OR condition ... ] THEN
      handler_statements
   ... ]
END;
```

Ejemplo:

```
CREATE TABLE db (a INT PRIMARY KEY, b TEXT);
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
CREATE FUNCTION merge_db(key INT, data TEXT) RETURNS VOID AS
$$
BEGIN
  LOOP
    -- first try to update the key

    UPDATE db SET b = data WHERE a = key;
    IF found THEN
       RETURN;
    END IF;
    -- not there, so try to insert the key
    -- if someone else inserts the same key concurrently,
    -- we could get a unique-key failure
    BEGIN
       INSERT INTO db(a,b) VALUES (key, data);
       RETURN;
    EXCEPTION WHEN unique_violation THEN
       -- do nothing, and loop to try the UPDATE again
    END;
  END LOOP;
END;
$$
LANGUAGE plpgsql;

SELECT merge_db(1, 'david');
SELECT merge_db(1, 'dennis');
```

Cursores

- Declaración:

```
name [ [ NO ] SCROLL ] CURSOR [ ( arguments ) ] FOR query;
```

Ejemplos:

```
DECLARE
 curs1 refcursor;
 curs2 CURSOR FOR SELECT * FROM tenk1;
 curs3 CURSOR (key integer) IS SELECT * FROM tenk1 WHERE unique1 = key;
```

- Abrir cursores:

```
OPEN unbound_cursor [ [ NO ] SCROLL ] FOR query;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
OPEN unbound_cursor [ [ NO ] SCROLL ] FOR EXECUTE query_string;

OPEN bound_cursor [ ( argument_values ) ];
```

Ejemplos:

```
OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;

OPEN curs1 FOR EXECUTE 'SELECT * FROM ' || quote_ident($1);

OPEN curs2;
OPEN curs3(42);
```

- Usar cursores:
  - FETCH
  - MOVE
  - UPDATE/DELETE WHERE CURRENT OF
  - CLOSE
- Devolviendo cursores:

Ejemplo:

```
CREATE FUNCTION reffunc2() RETURNS refcursor AS '
DECLARE
  ref refcursor;
BEGIN
  OPEN ref FOR SELECT col FROM test;
  RETURN ref;
END;
' LANGUAGE plpgsql;

BEGIN;
SELECT reffunc2();

     reffunc2
--------------------
<unnamed cursor 1>
(1 row)

FETCH ALL IN "<unnamed cursor 1>";
COMMIT;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Disparadores

Palabras claves dentro de los disparadores.

- NEW
- OLD
- TG_NAME
- TG_WHEN
- TG_LEVEL
- TG_OP
- TG_RELID
- TG_RELNAME
- TG_TABLE_NAME
- TG_TABLE_SCHEMA
- TG_NARGS
- TG_ARGV[]

Ejemplo 1:

```
CREATE TABLE emp (
  empname text,
  salary integer,
  last_date timestamp,
  last_user text
);

CREATE FUNCTION emp_stamp() RETURNS trigger AS $emp_stamp$
  BEGIN
     -- Check that empname and salary are given
     IF NEW.empname IS NULL THEN
         RAISE EXCEPTION 'empname cannot be null';
     END IF;
     IF NEW.salary IS NULL THEN
         RAISE EXCEPTION '% cannot have null salary', NEW.empname;
     END IF;

     -- Who works for us when she must pay for it?
     IF NEW.salary < 0 THEN
         RAISE EXCEPTION '% cannot have a negative salary', NEW.empname;
     END IF;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    -- Remember who changed the payroll when
    NEW.last_date := current_timestamp;
    NEW.last_user := current_user;
    RETURN NEW;
  END;
$emp_stamp$ LANGUAGE plpgsql;

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON emp
  FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
```

Ejemplo 2:

```
CREATE TABLE emp (
  empname           text NOT NULL,
  salary            integer
);

CREATE TABLE emp_audit(
  operation         char(1)   NOT NULL,
  stamp             timestamp NOT NULL,
  userid            text      NOT NULL,
  empname           text      NOT NULL,
  salary integer
);

CREATE OR REPLACE FUNCTION process_emp_audit() RETURNS TRIGGER AS
$emp_audit$
  BEGIN
    --
    -- Create a row in emp_audit to reflect the operation performed on emp,
    -- make use of the special variable TG_OP to work out the operation.
    --
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO emp_audit SELECT 'D', now(), user, OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO emp_audit SELECT 'U', now(), user, NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO emp_audit SELECT 'I', now(), user, NEW.*;
        RETURN NEW;
    END IF;
```

```
    RETURN NULL; -- result is ignored since this is an AFTER trigger
  END;
$emp_audit$ LANGUAGE plpgsql;

CREATE TRIGGER emp_audit
AFTER INSERT OR UPDATE OR DELETE ON emp
   FOR EACH ROW EXECUTE PROCEDURE process_emp_audit();
```

Ejemplo 3:

```
--
-- Main tables - time dimension and sales fact.
--
CREATE TABLE time_dimension (
    time_key                integer NOT NULL,
    day_of_week             integer NOT NULL,
    day_of_month            integer NOT NULL,
    month                   integer NOT NULL,
    quarter                 integer NOT NULL,
    year                    integer NOT NULL
);
CREATE UNIQUE INDEX time_dimension_key ON time_dimension(time_key);

CREATE TABLE sales_fact (
    time_key                integer NOT NULL,
    product_key             integer NOT NULL,
    store_key               integer NOT NULL,
    amount_sold             numeric(12,2) NOT NULL,
    units_sold              integer NOT NULL,
    amount_cost             numeric(12,2) NOT NULL
);
CREATE INDEX sales_fact_time ON sales_fact(time_key);

--
-- Summary table - sales by time.
--
CREATE TABLE sales_summary_bytime (
    time_key                integer NOT NULL,
    amount_sold             numeric(15,2) NOT NULL,

    units_sold              numeric(12) NOT NULL,
    amount_cost             numeric(15,2) NOT NULL
);
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
CREATE UNIQUE INDEX sales_summary_bytime_key ON sales_summary_bytime(time_key);

--
-- Function and trigger to amend summarized column(s) on UPDATE, INSERT, DELETE.
--
CREATE OR REPLACE FUNCTION maint_sales_summary_bytime() RETURNS TRIGGER AS $maint_sales_summary_bytime$
   DECLARE
      delta_time_key          integer;
      delta_amount_sold       numeric(15,2);
      delta_units_sold        numeric(12);
      delta_amount_cost       numeric(15,2);
   BEGIN

      -- Work out the increment/decrement amount(s).
      IF (TG_OP = 'DELETE') THEN

         delta_time_key = OLD.time_key;
         delta_amount_sold = -1 * OLD.amount_sold;
         delta_units_sold = -1 * OLD.units_sold;
         delta_amount_cost = -1 * OLD.amount_cost;

      ELSIF (TG_OP = 'UPDATE') THEN

         -- forbid updates that change the time_key -
         -- (probably not too onerous, as DELETE + INSERT is how most
         -- changes will be made).
         IF ( OLD.time_key != NEW.time_key) THEN
                  RAISE EXCEPTION 'Update of time_key : % -> % not allowed', OLD.time_key, NEW.time_key;
         END IF;

         delta_time_key = OLD.time_key;
         delta_amount_sold = NEW.amount_sold - OLD.amount_sold;
         delta_units_sold = NEW.units_sold - OLD.units_sold;
         delta_amount_cost = NEW.amount_cost - OLD.amount_cost;

      ELSIF (TG_OP = 'INSERT') THEN
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    delta_time_key = NEW.time_key;
    delta_amount_sold = NEW.amount_sold;
    delta_units_sold = NEW.units_sold;
    delta_amount_cost = NEW.amount_cost;

END IF;


-- Insert or update the summary row with the new values.
<<insert_update>>
LOOP

    UPDATE sales_summary_bytime
       SET amount_sold = amount_sold + delta_amount_sold,
          units_sold = units_sold + delta_units_sold,
          amount_cost = amount_cost + delta_amount_cost
       WHERE time_key = delta_time_key;

    EXIT insert_update WHEN found;

    BEGIN
       INSERT INTO sales_summary_bytime (
                time_key,
                amount_sold,
                units_sold,
                amount_cost)
          VALUES (
                delta_time_key,
                delta_amount_sold,
                delta_units_sold,
                delta_amount_cost
               );

       EXIT insert_update;

    EXCEPTION
       WHEN UNIQUE_VIOLATION THEN
          -- do nothing
    END;
END LOOP insert_update;
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
        RETURN NULL;

    END;
$maint_sales_summary_bytime$ LANGUAGE plpgsql;

CREATE TRIGGER maint_sales_summary_bytime
AFTER INSERT OR UPDATE OR DELETE ON sales_fact
    FOR EACH ROW EXECUTE PROCEDURE maint_sales_summary_bytime();

INSERT INTO sales_fact VALUES(1,1,1,10,3,15);
INSERT INTO sales_fact VALUES(1,2,1,20,5,35);
INSERT INTO sales_fact VALUES(2,2,1,40,15,135);
INSERT INTO sales_fact VALUES(2,3,1,10,1,13);
SELECT * FROM sales_summary_bytime;
DELETE FROM sales_fact WHERE product_key = 1;
SELECT * FROM sales_summary_bytime;
UPDATE sales_fact SET units_sold = units_sold * 2;
SELECT * FROM sales_summary_bytime;
```

CASO DE ESTUDIO #1

Instrucciones

Para el caso de estudio siga los distintos pasos:

1. Tome un tema de interés
2. Establezca un área dentro del tema seleccionado
3. Proceda a delinear los datos que se manejan dentro del tema
4. Identifique las relaciones entre los datos
5. Construya las tablas de su caso de estudio
6. Plasme las tablas dentro de su diseño mediante un diagrama (Puede usar para ello las herramientas pgDesigner, dia o Umbrello)
7. Genere o escriba la rutina SQL para implementar su diseño.
8. Ejecute su rutina SQL en su base de datos remota
9. Identifique tareas que desea realizar dentro de su base de datos
10. Automatice dichas tareas mediante funciones y disparadores realizados en PL/pgSQL.

El resultado de su caso de estudio respáldelo y envíelo al instructor por correo electrónico.

Materiales de apoyo en línea

- Web de PostgreSQL
- Normalización de bases de datos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

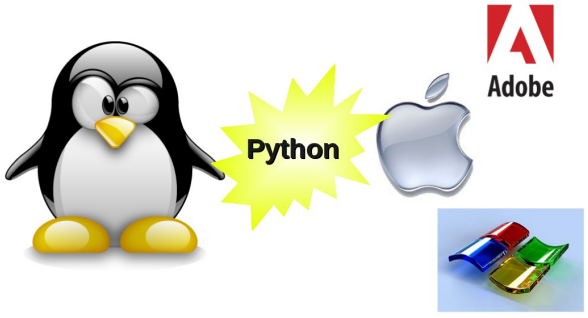
INTRODUCCIÓN A PYTHON

Python en la Empresa

¿Por qué python es la mejor alternativa?
Experiencias con el uso Python dentro de una Empresa

- Razones de Guido van Rossum
  - Reducción de tiempo de desarrollo
    - Código 2 a 10 veces más corto que C, C++ o Java
  - Programa de Mantenimiento Mejorado
    - Código extremadamente leíble
  - Menos Entrenamiento
    - El lenguaje es muy sencillo de aprender

Software Libre vs. Software Privativo



Bases de Datos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

¿Qué ofrece python?

a una empresa

- Plataforma uniforme para desarrollo
- Menos requerimientos de sistemas
- Ahorro en herramientas de desarrollo
- Estabilidad en sus sistemas

¿Qué ofrece python?

a un programador

- Código legible
- Más programa en menos líneas
- Excelente documentación
- Libertad de elegir entre diferentes opciones
- Buenas prácticas

¿Qué ofrece python?

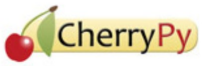
a un emprendedor

- Mayor efectividad
- Portabilidad de sus desarrollos
- Soluciones existentes en casi todos los ámbitos de negocio
- Posibilidad de soluciones rápidas y exitosas

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Web



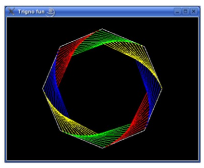


¿Qué plataformas puedo usar con python?




GUI's






Scripting

- Soluciones rápidas y efectivas
- Transparentes al usuario y sus procesos
- De mantenimiento sencillo
- Prácticamente presentes en todos los proyectos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Soluciones e Integración con los SIG

- Python es un estándar en SIG
- Provee desarrollos rápidos y efectivos
- Para análisis
- Para integración de datos
- Para incorporación a procesos

Soluciones a la medida

- Proveer al cliente exactamente lo que necesita
- El usuario debe sentirse cómodo con la solución presentada
- Deben hacerse entregas previas para que el cliente se las pruebe

A modo de Conclusión

- El código fuente es el 10% del negocio de software, el esfuerzo de codificar debe ser equivalente a este porcentaje
- El software libre es una muy buena alternativa
- El código fuente se lee más veces de las que se escribe
- Utilice las buenas prácticas de programación y metodologías de desarrollo probadas
- Busque quien lo ha hecho primero que usted y aproveche su experiencia y comentarios
- Utilice python :-)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

PRIMEROS PASOS CON PYTHON

Intérprete

Python es un lenguaje interpretado, fuerte y dinámicamente tipeado. Esto quiere decir que los programas escritos en Python son convertidos a lenguaje de máquina al momento de ejecutarlos. Así mismo, podemos interactuar con el intérprete interactivo de modo que podamos tener una conversación con el mismo. Lo referente al tipeado lo estudiaremos más adelante.

Para llamar al intérprete interactivo procedemos a llamarlo desde el terminal de su sistema operativo:

```
$ python
Python 2.7 (r27:82500, Sep 16 2010, 18:03:06)
[GCC 4.5.1 20100907 (Red Hat 4.5.1-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

El indicador **>>>** en el terminal indica que usted a entrado a una sesión interactiva con Python. Para probar el intérprete escriba lo que sigue a continuación:

```
>>> import this
```

Esto le presentará el huevo de pascua más famoso de Python, conocido como el Zen de Python.

Ahora procedamos a hacer algo más interesante ;-)

Introspección

Python es un lenguaje de programación orientado a objetos, la introspección nos permite explorar los aspectos de los elementos que estemos usando dentro de nuestro programa. Conversemos un poco con Python y descubramos de qué se

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

trata.
>>> 5*3

Examine el resultado, luego pruebe escribir
>>> a=5
>>> b=3
>>> a*b

ahora escriba, *type(a)*. *type()* es una función que le hará saber cuál tipo de dato es el correspondiente a una variable. Escriba ahora
>>> c="cnti "
>>> a*c
>>> type(c)
>>> dir(c)
>>> print c.__doc__

Funciones incluidas

Pruebe dentro de su intérprete interactivo las siguientes instrucciones.

- abs(valor)
- bool(valor)
- callable(valor)
- chr(numero)
- cmp(x,y)
- complex(r,j)
- file() / open()
- float(x)
- hash(objeto)
- hex()
- id(objeto)
- input("mensaje")
- int(valor, base)

- len()
- print
- list()
- long(valor, base)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

- map(función, lista)
- max()
- min()
- oct()
- pow()

en cada caso escriba primero **print [función].__doc__** y vea la descripción de la misma.

Escribir un programa en un archivo

Copie en un archivo de texto el siguiente código y luego llame al archivo *mi_programa.py*.

```
#!/bin/env python
# -*- coding: utf-8 -*-

print "Este es mi primer programa"
```

y luego en su terminal escriba:

```
$ python mi_programa.py
```

Estructuras con Python

Listas

Las listas como se verá, son uno de los caballos de batalla de Python. Las listas son colecciones de elementos que almacenamos en una estructura que posee una estructura similar a los arreglos o vectores en otros lenguajes. Dentro de una lista podemos almacenar distintos tipos de datos. Veamos algunos ejemplos de usos:

```
>>> l = ["hola", "mundo", "cnti", "canaima", "python"]
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python']
>>> l[0]
'hola'
>>> l[4]
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

'python'

Podemos recorrer la lista en sentido inverso mediante índices negativos:

```
>>> l[-1]

'python'
>>> l[-3]
'cnti'
```

También es posible cortar una lista:

```
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python']
>>> l[1:3]
['mundo', 'cnti']
>>> l[1:-1]
['mundo','cnti','canaima']
>>> l[0:3]
['hola', 'mundo', 'cnti']
>>> l[:3]
['hola', 'mundo', 'cnti']
>>> l[3:]
['canaima', 'python']
>>> l[:]
['hola', 'mundo', 'cnti', 'canaima', 'python']
```

Para agregar más elementos a una lista procedemos como sigue:

```
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python']
>>> l.append("nuevo")
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python', 'nuevo']
>>> l.insert(2, "nuevo")
>>> l
['hola', 'mundo', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo']
>>> l.extend(["dos", "elementos"])
>>> l
['hola', 'mundo', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo', 'dos', 'elementos']
```

Para buscar elementos en una lista usamos las siguientes formas:

```
>>> l
['hola', 'mundo', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo', 'dos', 'elementos']
```

```
>>> l.index("canaima")
4
>>> l.index("nuevo")
2
>>> l.index("chao")

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 'chao' is not in list
>>> "chao" in l
False
```

Para eliminar elementos de una lista contamos con las siguientes opciones:

```
>>> l
['hola', 'mundo', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo', 'dos', 'elementos']
>>> l.pop()
'elementos'
>>> l
['hola', 'mundo', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo', 'dos']
>>> l.remove("mundo")
>>> l
['hola', 'nuevo', 'cnti', 'canaima', 'python', 'nuevo', 'dos']
>>> l.remove("nuevo")
>>> l
['hola', 'cnti', 'canaima', 'python', 'nuevo', 'dos']
>>> l.remove("chao")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
>>> del l[0]
>>> l
['cnti', 'canaima', 'python', 'nuevo', 'dos']
```

Otras operaciones con listas son:

```
>>> l = ['hola', 'mundo', 'cnti']
>>> l = l + ['canaima', 'python']
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python']
>>> l += ["dos"]
>>> l
['hola', 'mundo', 'cnti', 'canaima', 'python', 'dos']
>>> l = [1, 2] * 3
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
>>> l
[1, 2, 1, 2, 1, 2]
```

Adicionalmente es posible anidar listas:

```
>>> mat = [
...       [1, 2, 3],
...       [4, 5, 6],
...       [7, 8, 9],
...      ]
```

Por último también veamos las listas por comprensión que se definen mediante operaciones desde otras listas. Veamos un ejemplo.

```
>>> l1 = [1, 2, 3, 4]
>>> l = [[x, x**2] for x in l1]
>>> l
[[1, 1], [2, 4], [3, 9], [4, 16]]
```

Tuplas

Las tuplas son listas inmutables, esto quiere decir que son colecciones de solo lectura. En este sentido una vez declarada una tupla pueden accederse a sus datos, mas no modificarse.

```
>>> t = ("hola", "mundo", "cnti", "canaima", "python")
>>> t
('hola', 'mundo', 'cnti', 'canaima', 'python')
>>> t[0]
'hola'
>>> t[-1]
'python'
>>> t[1:3]
('mundo', 'cnti')
>>> t.index("hola")
0
>>> "mundo" in t
True
```

Acciones destinadas a modificar las listas no son válidas para las tuplas, veamos

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

unos ejemplos:

```
>>> t
('hola', 'mundo', 'cnti', 'canaima', 'python')
>>> t.append('nuevo')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> t.remove("canaima")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'remove'
```

Diccionarios

Conocidos en otros lenguajes como arreglos asociativos, o memorias asociativas los diccionarios son agrupaciones de datos muy útiles dentro de Python. A diferencia de las listas y las tuplas donde el índice de un elemento es un número en el caso de los diccionarios se usan llaves o claves.

La mejor forma de imaginar un diccionario es como un grupo no ordenado de pares *llave:valor*

```
>>> tel = {'Carlos': '0326-555.22.02', 'Clara': '0895-888.22.02', 'Jose': '0343-543.54.43', }
>>> tel['Pedro'] = '0727-777.27.27'
>>> tel
{'Jose': '0343-543.54.43', 'Pedro': '0727-777.27.27', 'Clara': '0895-888.22.02', 'Carlos': '0326-555.22.02'}
>>> del tel['Jose']

>>> tel['Gustavo'] = '0955-222.32.32'
>>> tel
{'Gustavo': '0955-222.32.32', 'Pedro': '0727-777.27.27', 'Clara': '0895-888.22.02', 'Carlos': '0326-555.22.02'}
>>> tel.keys()
['Gustavo', 'Pedro', 'Clara', 'Carlos']
>>> 'Pedro' in tel
True
```

```
>>> tel.items()
[('Gustavo', '0955-222.32.32'), ('Pedro', '0727-777.27.27'), ('Clara', '0895-888.22.02'), ('Carlos', '0326-555.22.02')]
>>> tel.pop('Pedro')
'0727-777.27.27'
>>> dict([('cpu','x86' ), ('ram', 2), ('disco', 500)])
{'ram': 2, 'cpu': 'x86', 'disco': 500}
```

Condicionales

La instrucción if permite controlar el flujo de un programa en un par o más opciones. En Python no existe la instrucción *switch/case*, en cambio se usa la secuencia de *if .. elif .. .. else*

```
>>> x = int(raw_input("Ingrese un numero entero: "))
Ingrese un numero entero: 30
>>> if x < 0:
...     x = 0
...     print "Negativo cambiado a cero"
... elif x == 0:
...     print "cero"
... elif x == 1:
...     print "unico"
... else:
...     print "otro caso"
...
otro caso
```

Bucles

Los bucles son formas de iterar sobre una serie de datos o durante una condición que permite atender a procesos que así lo requieran.

Instrucción "for"

La instrucción *for* difiere en Python al funcionamiento tradicional en otros lenguajes, tales como C o Pascal. En vez de iterar sobre una serie de valores numéricos definidos por una condición de parada y un paso, Python utiliza la definición de una colección.

```
>>> a = ['gato', 'ventana', 'aguacate']
>>> for x in a:


...     print x, len(x)
...
gato 4
ventana 7
aguacate 8
```

Función "range"

Cuando se requiere una secuencia de números al estilo de como se definen en la instrucción *for* en otros lenguajes, hacemos uso de la función incluida *range*.

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(-10, -100, -30)
[-10, -40, -70]
```

Instrucción "break"

break rompe un bucle realizado con una instrucción *for* o *while*, veamos un ejemplo de su funcionamiento.

```
>>> for n in range(2, 10):
...     for x in range(2, n):
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
...         if n % x == 0:
...             print n, 'es igual a', x, ' * ', n/x
...             break
...     else:
...         # loop fell through without finding a factor
...         print n, 'es un numero primo'
```

Instrucción "pass"

la instrucción *pass* es para decir a Python explícitamente que no haga nada.

```
>>> while True:
...     pass  # Esperar hasta interrumpir con el teclado (Ctrl+C)
...
>>> class MyEmptyClass:
...     pass
...
>>> def initlog():
...     pass   # Remember to implement this!
```

Manejo de Errores

El manejo de errores se realiza mediante la serie de instrucciones *try* .. *except* .. *finally*. Veamos un ejemplo

```
>>> try:
...     x = 1/0
... except Exception as e:
        print 'el error es: ', e
... finally:
        print 'esta porción siempre se ejecuta'

 ...
 el error es:  integer division or modulo by zero
 esta porción siempre se ejecuta
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Funciones

Podemos definir una función mediante la instrucción *def*, veamos un ejemplo

```
>>> def fib(n):
...     """Imprime una serie de Fibonacci hasta el valor n."""
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...
>>> fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

podemos asociar funciones a una variable:

```
>>> fib
<function fib at 10042ed0>
>>> f = fib
>>> f(100)
0 1 1 2 3 5 8 13 21 34 55 89
```

Modifique la función para que genere una lista en vez de imprimir el resultado.

Este es un pequeño programa que pide confirmación:

```
def ask_ok(prompt, retries=4, complaint='Yes or no, please!'):
    while True:
        ok = raw_input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True

        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise IOError('refusenik user')
        print complaint
```

Clases

Para definir una clase en Python usamos la siguiente sintaxis

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
class MiClase(OtraClase):
    atributo1 = ""
    atributo2 = ""
    def __init__(self):
        pass # inicializa la clase
    def metodo1(self, arg1, arg2):
        pass
    def metodo2(self):
        pass
```

Hagamos un ejemplo:

```
>>> class Persona():
...     def __init__(self,nombre):
...         self.nombre = nombre
...         self.estatura = 0
...         self.edad = 0
...     def crece(self, cm):
...         self.estatura += cm
...     def cumpleanos(self):
...         self.edad +=1
...
>>> p = Persona('Pedro')
```

Trabajando con Archivos

Para leer un archivo Python se vale de la instrucción *open* que devuelve un objeto tipo *file* al que podemos recorrer.

```
>>> f = open("mi_archivo.txt")
>>> try:
...     for line in f:
...         print line
... finally:
...     f.close()
...
```

En caso que el archivo esté comprimido podemos accederlo importando el módulo *gzip*.

```
>>> import gzip
>>> f = gzip.open('mi_archivo.txt.gz')
>>> contenido = f.read()
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
>>> f.close()
```

Módulos

Un módulo es un archivo que contiene definiciones y declaraciones en Python. Para llamar a un módulo usamos la instrucción *import* llamando al archivo con el sufijo *.py*.
Tomemos por ejemplo el siguiente archivo

```
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

En el intérprete de python llamamos a nuestro módulo (archivo)

```
>>> import fibo
```

Ahora hacemos uso de nuestras funciones disponibles dentro del módulo

```
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

Podemos hacer un alias de nuestra función

```
>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Otra forma de llamar a los módulos es mediante la forma *from ... import*

```
>>> from fibo import fib, fib2
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Paquetes

Un paquete es una agrupación de módulos y puede ilustrarse como la siguiente estructura de carpetas y archivos

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
            __init__.py
            wavread.py
            wavwrite.py
            aiffread.py
            aiffwrite.py
            auread.py
            auwrite.py
            ...
      effects/                  Subpackage for sound effects
            __init__.py
            echo.py
            surround.py
            reverse.py
            ...
      filters/                  Subpackage for filters
            __init__.py
            equalizer.py
            vocoder.py
            karaoke.py
            ...
```

y se pueden hacer llamados a todo el paquete o simplemente a partes de el.

```
>>> import sound
```

o

```
>>> import sound.effects.echo
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

BASES DE DATOS CON PYTHON

El esquema que utiliza Python para la conexión a bases de datos se conoce como DB API 2.0. En el mismo está diseñado para que puedan hacerse hilos (o sesiones) seguros en los cuales se hagan uso intenso de instrucciones SQL tales como INSERT o UPDATE. En particular vamos a examinar el uso del módulo Psycopg2.

Comencemos por instalar Psycopg2.

Instalación

Para instalar Psycopg2 aplicamos el paquete python-psycopg2 en nuestro gestor de aplicaciones.

$ aptitude install python-psycopg2

Uso del Psycopg2

Una vez instalado Psycopg2 siga los siguientes pasos

```
>>> import psycopg2

# Conectarse a una base de datos existente
>>> conn = psycopg2.connect("dbname=test user=postgres")

# Abrir un cursor para realizar las operaciones con la base de datos
>>> cur = conn.cursor()

# Ejecutar un comando: este crea una tabla nueva
>>> cur.execute("CREATE TABLE test (id serial PRIMARY KEY, num integer, data varchar);")

# al pasar los datos Pycopg2 realiza las conversiones adecuadas de los datos
# (no más SQL injections!)
>>> cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)",
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
...     (100, "abc'def"))

# Consultar una base de datos para obtener unos datos como objetos de Python
>>> cur.execute("SELECT * FROM test;")
>>> cur.fetchone()
(1, 100, "abc'def")

# Hacer que los cambios en la base de datos sean permanentes
>>> conn.commit()

# Cerrar la comunicación con la base de datos
>>> cur.close()
>>> conn.close()
```

Pasando Parámetros a consultas SQL

Psycopg convierte las variables de Python en los valores adecuados para PostgreSQL

Ejemplo: el llamado de la función

```
>>> cur.execute(
...     """INSERT INTO some_table (an_int, a_date, a_string)
...         VALUES (%s, %s, %s);""",
...     (10, datetime.date(2005, 11, 18), "O'Reilly"))
```

Se convierte en el comando SQL

```
INSERT INTO some_table (an_int, a_date, a_string)
 VALUES (10, '2005-11-18', 'O''Reilly');
```

Argumentos nombrados también son válidos colocando %(*nombre*)s en los espacios. Así al pasar los argumentos desde un diccionario es posible obviar el orden original de los datos.

```
>>> cur.execute(
...     """INSERT INTO some_table (an_int, a_date, another_date, a_string)
...         VALUES (%(int)s, %(date)s, %(date)s, %(str)s);""",
...     {'int': 10, 'str': "O'Reilly", 'date': datetime.date(2005, 11, 18)})
```

Veamos algunos ejemplos a tomar en cuenta

```
>>> cur.execute("INSERT INTO numbers VALUES (%d)", (42,)) # MAL
>>> cur.execute("INSERT INTO numbers VALUES (%s)", (42,)) # correcto
>>> cur.execute("INSERT INTO foo VALUES (%s)", "bar")    # MAL
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
>>> cur.execute("INSERT INTO foo VALUES (%s)", ("bar"))  # MAL
>>> cur.execute("INSERT INTO foo VALUES (%s)", ("bar",)) # correcto
```

En el primer caso se intentó usar la plantilla %d en vez de %s, en los otros casos el segundo argumento de *execute* **DEBE** ser una tupla.

Errores Frecuentes

La Inyección SQL es uno de los problemas más frecuentes cuando trabajamos con aplicaciones que requieren el uso de base de datos. Es por ello que vamos a ver en un ejemplo:

```
>>> SQL = "INSERT INTO authors (name) VALUES ('%s');" # OJO.. ERROR
>>> data = ("O'Reilly", )
>>> cur.execute(SQL % data) # ESTO FALLA!!!
ProgrammingError: syntax error at or near "Reilly"
LINE 1: INSERT INTO authors (name) VALUES ('O'Reilly')
                                           ^
```

¿Por qué falla? pues debido a la conversión de datos de la consulta, es por ello que debemos apoyarnos en las validaciones y conversiones que Psycopg2 hace por nosotros.

```
>>> SQL = "INSERT INTO authors (name) VALUES (%s);" # sin comillas
>>> data = ("O'Reilly", )
>>> cur.execute(SQL, data) # sin operador %
```

Advertencia

Nunca, **nunca**, **NUNCA** use la concatenación de cadenas de Python (+) o la interpolación de parámetros (%) para pasar variables a un string SQL.
NUNCA!!!

Adaptación de Tipos de Datos

usaremos la función mogrify para ver que sucede detrás de las cámaras con nuestros datos.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
>>> cur.mogrify("SELECT %s, %s, %s;", (None, True, False))
>>> 'SELECT NULL, true, false;'
>>> cur.mogrify("SELECT %s, %s, %s, %s;", (10, 10L, 10.0, Decimal("10.00")))
>>> 'SELECT 10, 10, 10.0, 10.00;'
>>> import datetime
>>> dt = datetime.datetime.now()
>>> dt
datetime.datetime(2011, 5, 7, 10, 45, 43, 304394)
>>> cur.mogrify("SELECT %s, %s, %s;", (dt, dt.date(), dt.time()))
"SELECT '2010-02-08T01:40:27.425337', '2010-02-08', '01:40:27.425337';"
>>> cur.mogrify("SELECT %s;", (dt - datetime.datetime(2010,1,1),))
"SELECT '38 days 6027.425337 seconds';"
>>> cur.mogrify("SELECT %s;", ([10, 20, 30], ))
'SELECT ARRAY[10, 20, 30];'
>>> cur.mogrify("SELECT %s IN %s;", (10, (10, 20, 30)))
'SELECT 10 IN (10, 20, 30);'
```

Funciones de interés con cursores

- execute
- mogrify
- executemany
- fetchone
- fetchmany
- tetchall
- query
- statusmessage

CASO DE ESTUDIO #2

Seleccione uno de los 2 casos de estudio.

Caso 2.a

Juego del Ahorcado
Hacer un script para jugar al ahorcado.

- las palabras del juego deben estar o en un archivo de texto plano o en una tabla dentro de una base de datos.
- la persona tendrá 10 intentos para adivinar la palabra

Caso 2.b

Script para su base de datos
Hacer un script que permita realizar las siguientes operaciones a una tabla de su base de datos:

- Agregar un registro
- Listar los registros
- Borrar un registro

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

HERRAMIENTAS PARA TRABAJAR CON PYTHON

Para el desarrollo de aplicaciones es conveniente contar con una serie de herramientas que permitan sacar el mayor provecho de la plataforma. Es por ello que acá abordaremos el tema de forma simple. Abordaremos pues el tema a partir de tres componentes; Entornos Virtuales, Control de Versiones, Documentación y Editor/IDE de desarrollo.

Entornos Virtuales

Existen multitud de herramientas y técnicas para crear entornos virtuales dentro de Python, sin embargo la más popular es **virtualenv**.

Primero procedamos a instalar virtualenv en nuestro computador:

$ su -c "aptitude install python-virtualenv"

Luego en nuestro terminal escribimos

$ virtualenv --help

y vemos las opciones del programa, en particular la llamada a *–no-site-packages*.

Ahora si, vamos a crear un entorno virtual, escriba en su terminal

```
$ virtualenv --no-site-packages env
$ source env/bin/activate
(env)$
```

Esto quiere decir que no solamente hemos creado un entorno virtual sino que también lo hemos activado. Ahora dentro de el entorno virtual procedamos a instalar algunos programas

```
(env)$ pip install mercurial
(env)$ pip install django==1.2.5
(env)$ pip install psycopg2
(env)$ pip install sphinx
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Los paquetes que usamos en nuestro entorno virtual se encuentran en el índice de paquetes de Python en http://pypi.python.org/pypi (también conocida como la Cheese Shop, por el famoso sketch del los Monty Python). Cómo se ve en el caso de Django podemos especificar la versión de algún paquete que deseamos instalar, esto es especialmente útil si nuestro sistema de paquetería tiene versiones distintas a las que requerimos para nuestros desarrollos.

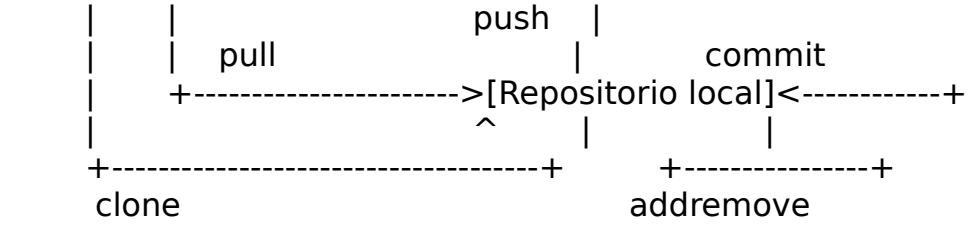
Luego de usar el entorno virtual podemos salir con la instrucción:
(env)$ deactivate
$

Control de Versiones

Igualmente en el mundo de los sistemas de control de versiones hay una infinidad de opciones, desde los conocidos cvs, subversión y bazar pasando por los más novedosos, git y mercurial.

En el mundo Python mercurial es una de las herramientas que viene a ayudar con esta tarea. Es importante tener en cuenta que si bien el uso de un sistema de control de versiones no es obligatorio, si es una muy buena práctica, al punto que es inconcebible una buena gestión de proyectos informáticos sin un buen sistema de control de versiones.

Veamos cómo vemos el ciclo de desarrollo con un sistema de control de versiones con Mercurial, así tengamos en cuenta este gráfico,



El mismo representa el ciclo de vida regular entre un repositorio remoto y un repositorio de trabajo con mercurial. El primer paso para hacer este ciclo es proceder a crear el **repositorio remoto**, colocándonos en la carpeta que

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

seleccionamos para que sea nuestro repositorio escribimos:
$ hg init .

Lo cual crea una carpeta oculta (en unix) llamada .hg que será en adelante el centro de control de versiones de mercurial. Hay gestores de repositorios que hace este paso por nosotros y nos proveen de la dirección donde se encuentra dicho servidor.

Una vez creado el repositorio remoto, procedemos a hacer una copia de trabajo del mismo. Esta copia de trabajo se obtiene mediante el *clonado* del repositorio original, proceso que realizamos con el siguiente comando:

$ hg clone /ruta/al/repositorio/remoto/ /carpeta/local/

Nota
En caso de no colocar el nombre de la carpeta local, mercurial asume que ud. desea usar el mismo nombre que el repositorio original.

Nota
La ruta al repositorio remoto puede ser una dirección **http** una carpeta dentro de su sistema de archivos o incluso una carpeta dentro de una conexión **ssh**

Trabajando sobre la copia de trabajo del repositorio se realizan los cambios sobre los archivos con las incorporaciones, modificaciones y remociones pertinentes. Una vez que finalizamos una sesión de trabajo nos aseguramos de incorporar y borrar de nuestro repositorio todos los archivos a los que hacemos control de versiones. Para ello usamos el comando:

$ hg addremove

Y finalmente agregamos los cambios realizados al sistema de control de versiones en forma de un *conjunto de cambios* (changeset). Para ello utilizamos la instrucción:

$ hg commit

Al ejecutar esta opción de mercurial se nos abre un editor de texto para que se ingrese la información sobre el cambio. Sin embargo hay un método más expedito que es agregando la opción adicional -m:

$ hg commit -m '[SU MENSAJE SOBRE LA CONSIGNACIÓN DE CAMBIOS]'

Una vez que terminamos de hacer todos nuestros *conjuntos de cambios* procede, si así lo desea, enviar estos al repositorio central de nuevo. Ello se hace con la siguiente instrucción:

$ hg push /ruta/al/repositorio/remoto/

Finalmente para halar los cambios de un repositorio sobre el que esté trabajando se usa:
$ hg pull
$ hg update

Documentación

La documentación de un proyecto es una de las partes fundamentales del trabajo de un desarrollador. Es por ello que es muy importante que tengamos a la mano una herramienta que nos facilite el trabajo.

En el mundo de las aplicaciones de Python la herramienta de documentación por excelencia es Sphinx. Sphinx utiliza un formato conocido como reStructuredText para escribir los contenidos.

Instalando Sphinx

Puede hacerlo mediante cualquier método que usted seleccione:

- Por su paquetería de GNU/Linux (*$ aptitude install python-sphinx*)
- Con setuptools ($ easy_install sphinx)
- Con el instalador de paquetes pip ($ pip install sphinx)

QuickStart

Proceda a crear un directorio y acceda al mismo:
$ mkdir documentacion

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
$ cd documentacion
Siga con el siguiente paso para iniciar su proyecto de documentación:
$ sphinx-quickstart
> Root path for the documentation [.]: <ENTER>
> Separate source and build directories (y/N) [n]: y
> Name prefix for templates and static dir [_]: <ENTER>
> Project name: ejemplo
> Author name(s): Su Nombre
> Project version: 0.0.1
> Project release [0.0.1]: <ENTER>
> Source file suffix [.rst]: <ENTER>
> Name of your master document (without suffix) [index]: <ENTER>
...
```

conf.py

Contiene los detalles de configuración de nuestro proyecto. Algunas de las tareas que usted querrá hacer con este archivo son:

- Cambiar la versión de su documentación
- Ajustar los nombres de los autores
- Colocar un logotipo a su proyecto
- Usar un tema para la generación de su documentación entre otras tareas.

Este archivo es bastante sencillo y está muy bien documentado, se recomienda recorrerlo con su editor de textos favorito y examinar las distintas opciones.

Recursos para reStructuredText

¿Cómo aprender este lenguaje de escritura de documentos? los siguientes links nos llevan a la documentación oficial,

- http://docutils.sourceforge.net/rst.html
- http://docutils.sourceforge.net/docs/user/rst/quickref.html
- http://docutils.sourceforge.net/docs/user/rst/cheatsheet.txt

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

aun así sphinx tiene una guía que puede ser de mucha ayuda, ver.
Esta guía del curso ha sido diseñada y escrita en Sphinx, descargue los fuentes acá y a partir del texto cree una guía de algún tema incluido en el curso.

Editores e IDEs

He aquí algunas alternativas:

- VIM (Vim como un IDE para Python)
- Emacs (Video de Emacs con Rope, otro artículo)
- DreamPie
- gEdit
- PyDev
- SPE
- Komodo Edit
- Anjuta
- Geany

Para una lista más extensa visite http://wiki.python.org/moin/PythonEditors
Al final la selección de un IDE o de un Editor es una elección muy personal, sólo como comentario y consejo, independientemente de la herramienta que use, primero comprenda que quiere hacer, el lenguaje y luego busque la herramienta. No condicione su proyecto a las opciones que ofrece su IDE o Editor.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

PROGRAMACIÓN WEB

Antes de ver las particularidades de la programación web con Python, repasemos algunos conceptos de la programación web con pequeñas pastillas:

- La programación web responde a una arquitectura cliente/servidor
- El servidor y el cliente se comunican a través de la red con un protocolo y un lenguaje
- El protocolo estándar de comunicación es HTTP
- El lenguaje estándar que el navegador interpreta es HTML
- El estándar actual de HTML es su versión 4.01
- La programación web del lado del cliente corresponde a 3 patas de una silla
    - HTML (Lenguaje Marcado)
    - JavaScript (Lenguaje de Programación)
- CSS (Hojas de Estilo)

HTML

- **HTML** significa *HyperText Markup Language*
- HTML No es un lenguaje de programación, es un *lenguaje de marcado*
- Un lenguaje de marcado usa una serie de **etiquetas**
- HTML utiliza etiquetas de marcado para describir páginas web.
- El estándar actual es HTML 4.01
- Los archivos que contienen documentos HTML generalmente tienen la extensión .html o .htm

Etiquetas de HTML

- Las etiquetas HTML están rodeadas por **<** y **>**.
- Comúnmente vienen en pares, por ejemplo ** viene acompañado de la etiqueta **.
- La primera etiqueta se le conoce como *etiqueta de apertura o de inicio*.
- La segunda etiqueta se le conoce como *etiqueta de cierre o final*.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Listado de Etiquetas HTML

El siguiente es el listado de etiquetas HTML correspondiente al estándar actual.

Etiqueta	Descripción
<!–...–>	Comentario
<!DOCTYPE>	Tipo de documento
<a>	Ancla
<abbr>	Abreviatura
<acronym>	Acrónimo
<address>	Información de contacto del autor/dueño de la página
<applet>	**Desaprobado**.
<area />	Área dentro de un image-map
<b>	Texto en negrillas
<base />	Dirección por defecto para todos los links en una página
<basefont />	**Desaprobado**.
<bdo>	Dirección del texto
<big>	Textos grandes
<blockquote>	Cita larga
<body>	Cuerpo del documento
 	Salto de línea
<button>	Botón
<caption>	Título de una tabla
	Desaprobado.
<cite>	Cita
<code>	Códigos fuentes
<col />	Atributos para una columna o más en una tabla
<colgroup>	Grupo de columnas en una tabla para formato
<dd>	Descripción de un termino en una lista de definiciones

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Etiqueta	Descripción
<del>	Texto borrado
<dfn>	Definición de un término
<dir>	**Desaprobado**.
	Sección dentro del documento
<dl>	Lista de definiciones
<dt>	Término (item) en una lista de definiciones
	Texto Enfatizado
<fieldset>	Borde alrededor de los elementos de un formulario
<font>	**Desaprobado**.
<form>	Formulario HTML
<frame />	Marco en un conjunto de marcos (frameset)
<frameset>	Conjunto de marcos
<h1> to <h6>	Encabezados HTML
<head>	Información sobre el documento
<hr />	Línea Horizontal
<html>	Documento HTML
<i>	Texto en Itálicas
<iframe>	Marco incrustado
<img />	Imagen
<input />	Control de ingreso de datos
<ins>	Texto Insertado
<isindex>	**Desaprobado**.
<kbd>	Texto del teclado
<label>	Etiqueta para un control *input*
<legend>	Título para un *fieldset*
	Item de una lista
<link />	Relación con un documento o fuente externa
<map>	Mapa de una imagen
<menu>	**Desaprobado**.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Etiqueta	Descripción
<meta />	Metadata sobre el documento HTML
<noframes>	Contenido alternativo para usuarios que no soportan marcos
<noscript>	Contenido alternativo para usuarios que no soportan scripts
<object>	Objeto Incrustado
	Lista ordenada
<optgroup>	Grupo de opciones relacionadas en una lista
<option>	Opción en una lista de selección
<p>	Párrafo
<param />	Parámetros para un objeto
<pre>	Texto preformateado
<q>	Cita corta
<s>	**Desaprobado**.
<samp>	Código Fuente de Ejemplo
<script>	Script
<select>	Lista de selección
<small>	Texto Pequeño
	Sección en el documento
<strike>	**Desaprobado**.
	Texto Fuerte
<style>	Estilos para el documento
<sub>	Texto de subíndices
<sup>	Texto de superíndices
<table>	Tabla
<tbody>	Grupo de contenidos en una tabla
<td>	Celda de una Tabla
<textarea>	Control de ingreso de texto de múltiples líneas
<tfoot>	Grupo de contenidos en el pié de una tabla

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Etiqueta	Descripción
<th>	Celda encabezado de una tabla
<thead>	Grupo de contenidos en el encabezado de una tabla
<title>	Título de un documento
<tr>	Fila en una tabla
<tt>	Texto de teletipo
<u>	**Desaprobado**.
	Lista sin orden
<var>	Parte variable en el texto
<xmp>	**Desaprobado**.

Atributos HTML

- Las etiquetas HTML poseen atributos
- Los atributos proveen información adicional del elemento
- Los atributos modifican tanto su comportamiento como su presentación
- Los atributos siempre se declaran en la etiqueta de inicio
- Los atributos vienen en pares nombre/valor y se escriben **nombre=”valor”**
- Los valores siempre van entre comillas
- Los atributos que posee todo elemento son:
    - **id**: especifica un id único para el elemento
    - **class**: especifica el nombre de una clase para un elemento
    - **style**: especifica un estilo para un elemento
    - **title**: especifica información extra para el elemento

Mi primera página Web

Abra su editor de texto seleccionado y escriba el siguiente código *HTML* dentro de un archivo al cual llamará **hola_mundo.html**.

```
<html>
   <head>
      <title>Hola Mundo</title>
   </head>
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
  <body>
    <h1>Hola Mundo!</h1>
    <hr />
    <p>Esto es un párrafo</p>
  </body>
</html>
```

Cambie los valores dentro de las etiquetas **<title>**, **<h1>** y **<p>**. Guarde el archivo y examine los cambios.

Mi segunda página Web

En el editor de su selección escriba el siguiente código fuente:

```
<html>
  <head>
    <title>Hola de Nuevo Mundo</title>
  </head>
  <body>
    <h1>Hola de Nuevo Mundo</h1>
  </body>
</html>
```

Ahora proceda a agregar los siguientes códigos a continuación de la etiqueta <h1> y observe los cambios.

```
<h2>Hola de Nuevo Mundo</h2>

<h3>Hola de Nuevo Mundo</h3>
<h4>Hola de Nuevo Mundo</h4>
```

Guarde y refresque la página

```
<p>Este es un p&aacute;rrafo</p>
<p>Este es otro p&aacute;rrafo</p>
```

Guarde y refresque la página

```
<a href="http://www.google.com/">esto es un link</a>
```

Guarde y refresque la página

```
<img src="ballena.png" width="200px" height="200px" />
```

Herramienta

En este punto, entre en Firefox y en la sección de Agregados busque e instale **Firebug**.

Nota: Valide que su versión de Firefox sea compatible con la versión descargada de Firebug.

JavaScript

- JavaScript es un lenguaje de programación script.
- JavaScript fue diseñado para interactuar con páginas web
- JavaScript es un lenguaje de programación ligero
- Usualmente JavaScript está incrustado en el código HTML
- JavaScript es un lenguaje interpretado
- Cualquiera puede usar JavaScript libre de pago
- JavaScript **NO** es Java

Con JavaScript usted puede:

- Programar dentro de HTML
- Hacer dinámica su página Web
- Reaccionar ante eventos en la página
- Leer y Escribir elementos en la página
- Validar Datos
- Detectar información del Navegador
- Crear cookies

El nombre real de Java Script es ECMAScript, y el estándar oficial del lenguaje es ECMA-262.

Ejemplos de JavaScript

```
<html>
   <body>
      <script type="text/javascript">
         document.write("HOLA MUNDO!");
      </script>
   </body>
</html>
<html>
   <body>
      <script type="text/javascript">
         document.write("<h1>Hola de Nuevo!</h1>");
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    </script>
  </body>
</html>
```

¿Dónde colocar el código JavaScript?

En el encabezado

```
<html>
  <head>
    <script type="text/javascript">
      function message()
      {
      alert("Esta señal de alerta fue llamada al cargar la página");
      }
    </script>
  </head>
  <body onload="message()">
  </body>
</html>
```

En el cuerpo

```
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Este mensaje es escrito por JavaScript");
    </script>
  </body>
</html>
```

En el encabezado y en el cuerpo

```
<html>
  <head>
    <script type="text/javascript">
      function message()
      {
      alert("Esta señal de alerta fue llamada al cargar la página");
      }
    </script>
  </head>
  <body onload="message()">
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
      <script type="text/javascript">
         document.write("Este mensaje es escrito por JavaScript");
      </script>
   </body>
</html>
```

Usando un archivo externo

Archivo: demo.js

```
function message() {
   alert("Esta señal de alerta fue llamada  desde un archivo al cargar la página");
}
```

Archivo: ubica_js.html

```
<html>
   <head>
      <script type="text/javascript" src="demo.js"></script>
   </head>
   <body onload="message()">
   </body>
</html>
```

Principios de JavaScript

Estos son elementos del lenguaje que se utilizarán regularmente.

Declaraciones

- JavaScript es sensible a las mayúsculas y minúsculas
- Una declaración es una instrucción que se da en JavaScript
- El código es una secuencia de declaraciones
- Cada declaración termina con el símbolo de **;**

Ejemplo:

```
<script type="text/javascript">
   document.write("<h1>Esto es un encabezado</h1>");
   document.write("<p>Esto es un párrafo</p>");
   document.write("<p>esto es otro párrafo</p>");
</script>
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Bloques de código

- Los bloques de código en JavaScript se agrupan en llaves (**{**, **}**).

Ejemplo:

```
<script type="text/javascript">
   {
   document.write("<h1>Esto es un encabezado</h1>");
   document.write("<p>Esto es un párrafo</p>");
   document.write("<p>esto es otro párrafo</p>");
   }
</script>
```

Comentarios

Los comentarios dentro de JavaScript corresponden a los símbolos **//** si es en un sola línea o **/*** y ***/** en caso de ser de múltiples líneas.

Ejemplo:

```
<script type="text/javascript">
   // este es un comentario corto
   document.write("<h1>Esto es un encabezado</h1>");
   /*
      este
      comentario
      es
      largo
   */
   document.write("<p>Esto es un párrafo</p>");
   document.write("<p>esto es otro párrafo</p>");
</script>
```

Variables

- Las variables en JavaScript deben comenzar siempre por un carácter o el símbolo _.
- Las variables son sensibles a las mayúsculas y a las minúsculas
- Se utiliza la instrucción **var** para declarar una variable
- Al declarar una variable puede asignarle inmediatamente un valor

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
var x;
var nombre;
var x=5;
var nombre="Carlos";
```

Si asigna valores a variables no declaradas tendrá el mismo efecto que las líneas anteriores

```
x=5;nombre="Carlos";
```

Operadores

JavaScript posee los siguientes operadores

Operador	Función
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
++	Incremento
–	Decremento

Operadores de Comparación

Operador	Función
==	Igual
===	Exactamente Igual
!=	Distinto
<	Menor que
>	Mayor que
<=	Menor Igual
>=	Mayor Igual

Ejemplo:

```
if (edad<18) document.write("Menor de Edad");
```

Operadores Lógicos

Operador	Función
&&	Y
||	O
!	No

Condicionales

- if

Ejemplo:

```
<script type="text/javascript">

var d=new Date();
var time=d.getHours();

if (time<12)
  {
  document.write("<b>Buenos D&iacute;as</b>");
  }
</script>
```

- if/else

Ejemplo:

```
<script type="text/javascript">

var d=new Date();
var time=d.getHours();

if (time<12)
  {
  document.write("<b>Buenos D&iacute;as</b>");
  }
else if (time>12 && time<16)
  {
  document.write("<b>Buenas Tardes</b>");
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

```
 }
else
 {
 document.write("<b>Hola Mundo!</b>");
 }
</script>
```

También existe la instrucción **switch/case**.

Mensajes desde JavaScript

- alert

Ejemplo:

```
<html>
   <head>
   <script type="text/javascript">
      function show_alert()
      {
      alert("Hola, este es un mensaje de alerta");
      }
   </script>
   </head>
   <body>
      <input type="button" onclick="show_alert()" value="Mensaje de Alerta" />
   </body>
</html>
```

- confirm

Ejemplo:

```
<html>
<head>
   <script type="text/javascript">
   function show_confirm()
   {
   var r=confirm("Presione un Botón");
   if (r==true)
     {
     alert("Ud seleccionó OK!");
     }
   else
     {
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    alert("Ud seleccionó Cancelar!");
    }
  }
  </script>
</head>
  <body>
    <input type="button" onclick="show_confirm()" value="Confirmar" />
  </body>
</html>
```

- prompt

Ejemplo:

```
<html>
  <head>
    <script type="text/javascript">

    function show_prompt()
    {
    var name=prompt("Su nombre por favor","Harry Potter");
    if (name!=null && name!="")
      {
      document.write("Hola " + name + "! C&oacute;mo est&aacute;s?");
      }
    }
    </script>
  </head>
  <body>
    <input type="button" onclick="show_prompt()" value="Mensaje" />
  </body>
</html>
```

Bucles

- for

Ejemplo:

```
<html>
  <body>
    <script type="text/javascript">
    var i=0;
    for (i=0;i<=5;i++)
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
      {
      document.write("El nro es: " + i);
      document.write("<br />");
      }
      </script>
   </body>
</html>
```

- while

```
<html>
   <body>
   <script type="text/javascript">
   var i=0;
   while (i<=5)
     {
     document.write("El nro. es: " + i);
     document.write("<br />");
     i++;
     }
   </script>
</body>
</html>
```

o también puede utilizarse como:

```
<html>
   <body>
      <script type="text/javascript">
      var i=0;
      do
        {
        document.write("El nro. es " + i);
```

- for

Ejemplo:

```
<html>
   <body>
      <script type="text/javascript">
      var i=0;
      for (i=0;i<=5;i++)
      {
      document.write("El nro es: " + i);
      document.write("<br />");
      }
      </script>
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
   </body>
</html>
```

- while

```
<html>
   <body>
   <script type="text/javascript">
   var i=0;
   while (i<=5)
    {
    document.write("El nro. es: " + i);
    document.write("<br />");
    i++;
    }
   </script>
</body>
</html>
```

o también puede utilizarse como:

```
<html>
   <body>
      <script type="text/javascript">
      var i=0;
        document.write("<br />");
        i++;
        }
      while (i<=5);
      </script>
   </body>
</html>
```

Funciones

Las funciones sirven para agrupar un conjunto de instrucciones que hacen una tarea en particular.
Ejemplo:

```
<html>
   <head>
      <script type="text/javascript">
      function displaymessage()
      {
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
      alert("Hola Mundo!");
      }
      </script>
   </head>
   <body>
      <form>
         <input type="button" value="Clic Aquí!" onclick="displaymessage()" />
      </form>
   </body>
</html>
```

Capturando Errores

Esta operación se realiza con las instrucciones **try/catch**.
Ejemplo:

```
<html>
   <head>
      <script type="text/javascript">
      var txt="";
      function message()
      {

      try
        {
        adddlert("Bienvenido!");
        }
      catch(err)
        {
        txt="Hubo un error en esta página.\n\n";
        txt+="Descripción del Error: " + err.description + "\n\n";
        txt+="Haga clic en OK para continuar.\n\n";
        alert(txt);
        }
      }
      </script>
   </head>
   <body>
      <input type="button" value="View message" onclick="message()" />
   </body>
</html>
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

CNTI CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN

Objetos

- String

Ejemplo:

```
var txt="Hello world!";
document.write(txt.length);
```

- Date

Ejemplo:

```
var myDate=new Date();
myDate.setFullYear(2011,0,28);
var today = new Date();
if (myDate>today)
  {
  alert("Hoy es antes del 28 de enero de 2011");
  }
else
  {
  alert("Hoy es después del 28 de enero de 2011");
  }
```

- Array

Ejemplos:

```
var myCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
var myCars=new Array("Saab","Volvo","BMW");
var myCars=["Saab","Volvo","BMW"];
```

- Boolean

Las siguientes expresiones siempre se declaran falsas

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

en caso contrario son verdaderas

Ejemplo:

```
var myBoolean=new Boolean(1);
var myBoolean=new Boolean(true);
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Carlos");
```

CSS

- **CSS**:Cascading Style Sheets
- Define los estilos como se muestran los elementos HTML
- Fueron añadidos en HTML 4.0 para solventar un problema (HTML no preveía estilos sino contenido, con la incorporación de etiquetas de formatos comenzó una pesadilla de diseño en la internet)
- Las Hojas de Estilo se guardan generalmente en archivos .css
- Las Hojas de Estilo externas ahorran gran cantidad de trabajo

Sintaxis

Selector {llave:valor; llave:valor;}
Ejemplo:

```
h1 {color:blue; font-size:12px;}
```

Ejemplo:

```
<html>
   <head>
      <title>Hola CSS!</title>
      <style>
      p {
         color:red;
         text-align:center;
         }
      </style>
   </head>
   <body>
      <p>HOLA CSS!! veamos cómo funciona</p>
   </body>
</html>
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Selector por id

Ejemplo:
#algunid
{
text-align:center;
color:red;
}

Selector por clase

Ejemplo:
.centrado {text-align:center;}
p.centrado {text-align:center;}

¿Dónde colocar las hojas de estilo?

En un archivo externo:
<head>
<link rel="stylesheet" type="text/css" href="mis_estilos.css" />
</head>
Archivo **mis_estilos.css**:
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
dentro del código html,
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
incrustado en los elementos html,
Esto es un párrafo.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

PROGRAMACIÓN WEB CON PYTHON (CGI)

La programación web con Python puede ser muy variada, pero en general se resume en dos grandes categorías, aplicaciones cgi y las aplicaciones creadas a partir de un entorno de programación o Framework.

En el primer caso un programa de Python tendría una forma como la que sigue:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import cgi

print "Content-Type: text/html\n\n"
print """<html>
<head>
   <title>CGI Python</title>
   <style>
   body {
      font-family: Dejavu Sans, Liberation Sans;
   }
   h1 {
      text-align: center;
   }
   p {
      width: 500px;
      margin: auto;
      color: #555;
      margin-top: 15px;
      text-align: justify;
   }
   </style>
</head>
<body>
   <h1>El Quijote</h1>
   <hr />
   <p>En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha
   mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga
   antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que
   carnero, salpicón las más noches, duelos y quebrantos los sábados,
   lantejas los viernes, algún palomino de añadidura los domingos,
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

consumían las tres partes de su hacienda. El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera. Frisaba la edad de nuestro hidalgo con los cincuenta años; era de complexión recia, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza. Quieren decir que tenía el sobrenombre de Quijada, o Quesada, que en esto hay alguna diferencia en los autores que deste caso escriben; aunque, por conjeturas verosímiles, se deja entender que se llamaba Quejana. Pero esto importa poco a nuestro cuento; basta que en la narración dél no se salga un punto de la verdad.

Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba ocioso, que eran los más del año, se daba a leer libros de caballerías, con tanta afición y gusto, que olvidó casi de todo punto el ejercicio de la caza, y aun la administración de su hacienda. Y llegó a tanto su curiosidad y desatino en esto, que vendió muchas hanegas de tierra de sembradura para comprar libros de caballerías en que leer, y así, llevó a su casa todos cuantos pudo haber dellos; y de todos, ningunos le parecían tan bien como los que compuso el famoso Feliciano de Silva, porque la claridad de su prosa y aquellas entricadas razones suyas le parecían de perlas, y más cuando llegaba a leer aquellos requiebros y cartas de desafíos, donde en muchas partes hallaba escrito: La razón de la sinrazón que a mi razón se hace, de tal manera mi razón enflaquece, que con razón me quejo de la vuestra fermosura. Y también cuando leía: ...los altos cielos que de vuestra divinidad divinamente con las estrellas os fortifican, y os hacen merecedora del merecimiento que merece la vuestra grandeza.
</body>
</html>"""

hidalgo con los cincuenta años; era de complexión recia, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza. Quieren decir que tenía el sobrenombre de Quijada, o Quesada, que en esto hay alguna diferencia en los autores que deste caso escriben; aunque, por conjeturas verosímiles, se deja entender que se llamaba Quejana. Pero esto importa poco a nuestro cuento; basta que en la narración dél no se salga un punto de la verdad.

```
    <p>Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba
    ocioso, que eran los más del año, se daba a leer libros de caballerías,
    con tanta afición y gusto, que olvidó casi de todo punto el ejercicio
    de la caza, y aun la administración de su hacienda. Y llegó a tanto su
    curiosidad y desatino en esto, que vendió muchas hanegas de tierra de
    sembradura para comprar libros de caballerías en que leer, y así, llevó
    a su casa todos cuantos pudo haber dellos; y de todos, ningunos le
    parecían tan bien como los que compuso el famoso Feliciano de Silva,
    porque la claridad de su prosa y aquellas entricadas razones suyas le
    parecían de perlas, y más cuando llegaba a leer aquellos requiebros y
    cartas de desafíos, donde en muchas partes hallaba escrito: La razón de
    la sinrazón que a mi razón se hace, de tal manera mi razón enflaquece,
    que con razón me quejo de la vuestra fermosura. Y también cuando leía:
    ...los altos cielos que de vuestra divinidad divinamente con las
    estrellas os fortifican, y os hacen merecedora del merecimiento que
    merece la vuestra grandeza.</p>
</body>
</html>"""
```

Ejercicio 1

Cree un script que vacíe los datos de una tabla en una web.

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

INTRODUCCIÓN A DJANGO

Definición

Según Wikipedia:

Django es un framework de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt En Junio del 2008 fue anunciado que la recién formada Django Software Foundation se hará cargo de Django en el futuro. La versión estable (a marzo de 2011) es la 1.3.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio de DRY (del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.
ver más.

Instalación

Para instalar Django tiene las siguientes opciones:

- Por su paquetería de GNU/Linux (*$ aptitude install python-django*)
- Con setuptools ($ easy_install Django)
- Con el instalador de paquetes pip ($ pip install Django)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Iniciando un Proyecto

Para crear un nuevo proyecto con django debemos escribir en la línea de comandos la siguiente instrucción:
$ django-admin startproject mi_app
Esto creará una estructura de directorios como sigue:
- mi_prj
  - __init__.py
  - manage.py
  - settings.py (configuraciones del proyecto)
  - urls.py - (control de las urls)

Examinemos un momento los archivos settings.py y urls.py

settings.py

```
# Django settings for cnti project.

DEBUG = True
TEMPLATE_DEBUG = DEBUG

ADMINS = (
    # ('Your Name', 'your_email@domain.com'),
)

MANAGERS = ADMINS

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.', # Add 'postgresql_psycopg2', 'postgresql', 'mysql', 'sqlite3' or 'oracle'.
        'NAME': '',                      # Or path to database file if using sqlite3.
        'USER': '',                      # Not used with sqlite3.
        'PASSWORD': '',                  # Not used with sqlite3.
        'HOST': '',                      # Set to empty string for localhost. Not used with sqlite3.
        'PORT': '',                      # Set to empty string for default. Not used with sqlite3.
    }
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
}

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# On Unix systems, a value of None will cause Django to use the same
# timezone as the operating system.
# If running in a Windows environment this must be set to the same as your
# system time zone.
TIME_ZONE = 'America/Chicago'

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en-us'

SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale
USE_L10N = True

# Absolute filesystem path to the directory that will hold user-uploaded files.
# Example: "/home/media/media.lawrence.com/"
MEDIA_ROOT = ''

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash if there is a path component (optional in other cases).
# Examples: "http://media.lawrence.com", "http://example.com/media/"
MEDIA_URL = ''

# URL prefix for admin media -- CSS, JavaScript and images. Make sure to use a
# trailing slash.
# Examples: "http://foo.com/media/", "/media/".
ADMIN_MEDIA_PREFIX = '/media/'

# Make this unique, and don't share it with anybody.
SECRET_KEY = 'it$s42548km$l7hx+^2)+57(o55@)1%s172$s6-0v4@yqzel@l'
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
# List of callables that know how to import templates from various sources.
TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
#     'django.template.loaders.eggs.Loader',
)

MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
)

ROOT_URLCONF = 'cnti.urls'

TEMPLATE_DIRS = (
                # Put strings here, like "/home/html/django_templates" or
"C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
)
```

Parametrice su archivo en las variables *TIME_ZONE* y *LANGUAGE_CODE*. Además configure las variables de la base de datos.

urls.py

A diferencia de otros acercamientos al desarrollo de aplicaciones las urls dentro de Django no representan ni porciones del modelo, ni archivos en directorios, las urls representan lo que el programador quiere que representen. Dicho de otro modo, los urls de su decisión son suya

```
from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    # Example:
    # (r'^mi_prj/', include('mi_prj.foo.urls')),

    # Uncomment the admin/doc line below to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # (r'^admin/', include(admin.site.urls)),
)
```

Probando nuestro Proyecto

Para ver que nuestro Proyecto funcione entremos a nuestro directorio y en la línea de comandos escribamos:

```
$ cd mi_prj
$ ./manage.py runserver
```

Luego visite la página http://localhost:8000 y verifique

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Creando nuestra primera Aplicación

Para crear nuestra primera aplicación procedemos a escribir en nuestra línea de comandos lo que sigue:

$ python manage.py startapp blog

y veremos que hemos agregado a nuestro directorio una carpeta llamada blog con cuatro nuevos archivos.

```
__init__.py
blog
    __init__.py
    models.py
    tests.py
    views.py
manage.py
settings.py
settings.pyc
urls.py
```

comencemos entonces por agregar nuestra aplicación al Proyecto, antes de trabajar con el modelo, incluyéndola en la lista de INSTALLED_APPS.

Modelo

Incluiremos en esta primera versión solamente un modelo para manejar los Post del Blog. En el archivo blog/models.py incluimos lo que sigue:

```
class Post(models.Model):

    titulo = models.CharField("Título")
    contenido = models.TextField()
    fecha = models.DateTimeField(auto_now=True)
```

ahora para que nuestro modelo se refleje en nuestra base de datos escribimos:

```
$ python manage.py validate    # para validar nuestro modelo
$ python manage.py syncdb
```

posteriormente veamos como trabajamos con un nuevo modelo desde el

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

intérprete escribiendo:

$ python manage.py shell

o directamente con la base de datos mediante el comando:

$ python manage.py dbshell

Habilitando el Admin de Django

El **admin** de Django es un entorno administrativo dispuesto para que facilitar el desarrollo de los proyectos de aplicaciones web. Para ello descomente las líneas sobre el admin en los archivos **settings.py** y **urls.py**. Luego sincronicemos nuestra base de datos para cargar las tablas concernientes al admin.

Posteriormente agregue en la aplicación blog un archivo llamado **admin.py** en donde escribiremos lo que sigue:

```
from django.contrib import admin
from models import *
admin.site.register(Post)
```

Finalmente levantemos el servidor de desarrollo.

Ahora un poco de programación

Ahora que tenemos nuestro modelo andando y el admin soportando el manejo de datos, crearemos una vista para desplegar nuestro rudimentario blog. Primero probemos un poco el funcionamiento de las vistas modificando el archivo views.py en nuestro proyecto.

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hola Mundo!")
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

ahora en el archivo urls.py mapeamos la vista agregando

```
...
(r'^blog/$', 'blog.views.index'),
...
```

Probemos nuestra vista visitando http://localhost:8000/blog/
Luego modifiquemos nuestra vista a:

```
from django.shortcuts import render_to_response
from models import *

def index(request):
    posts = Post.get_objects.all().order_by('-fecha')[:5]
    return render_to_response('blog.html',{'posts':post})
```

para que esto tenga sentido debemos crear el archivo blog.html dentro de la carpeta de plantillas (que deberá definir dentro del archivo settings.py).
El archivo blog.html deberá ser algo como:

```
<html>
    <head>
        <title>Blog</title>
    </head>
    <body>
    {% for p in posts %}
        <h1>{{ p.titulo }}</h1>
        <p>{{ p.contenido }}</p>
    {% endfor %}
    </body>
</html>
```

Siguientes pasos

1. Complete la página de su blog con las opciones más comunes (Usuarios, Tags, Categorías)
2. Cree su propia aplicación para el registro de algún proceso de su interés (Participantes para un Evento, Tickets de Soporte Técnico, etc)
3. Tenga a mano la documentación de Django (incluida en la carpeta extras)

Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

PROGRAMACIÓN GUI CON PYTHON

Opciones para GUI:

- Tkinter
- PyQT
- PyGTK
- wxpython

Tkinter

El módulo **tkinter** (Tk Interface) es la interfaz por defecto para la GUI Tk. Este módulo esta disponible para la mayoría de las distribuciones Linux, Mac y Windows. A partir de su versión 8.0 Tk toma la apariencia exacta del ambiente en que corre.

Hola Mundo (al estilo Tkinter)

```
>>> from Tkinter import *
>>> root = Tk()
>>> w = Label(root, text="Hola, Mundo!")
>>> w.pack()
>>> root.mainloop()
```

Hola de Nuevo

```
from Tkinter import *

class App:

    def __init__(self, master):
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
        frame = Frame(master)
        frame.pack()

        self.button = Button(frame, text="Salir", fg="red", command=frame.quit)
        self.button.pack(side=LEFT)

        self.hi_there = Button(frame, text="Hola", command=self.say_hi)
        self.hi_there.pack(side=LEFT)

    def say_hi(self):
        print "Epa! Hola a Todos!!"

root = Tk()

app = App(root)

root.mainloop()
```

Manejo de Eventos

```
from Tkinter import *

root = Tk()

def callback(event):
    print "hizo clic en: ", event.x, event.y

frame = Frame(root, width=100, height=100)
frame.bind("<Button-1>", callback)
frame.pack()

root.mainloop()
from Tkinter import *
import tkMessageBox

def callback():
    if tkMessageBox.askokcancel("Salir", "Realmente desea salir?"):
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    root.destroy()

root = Tk()
root.protocol("WM_DELETE_WINDOW", callback)

root.mainloop()
```

Creando un Menú

```
from Tkinter import *

def callback():
   print "Llamado a la función callback!"

root = Tk()

# crear un menu
menu = Menu(root)
root.config(menu=menu)

filemenu = Menu(menu)
menu.add_cascade(label="Archivo", menu=filemenu)
filemenu.add_command(label="Nuevo", command=callback)
filemenu.add_command(label="Abrir...", command=callback)
filemenu.add_separator()
filemenu.add_command(label="Salir", command=callback)

helpmenu = Menu(menu)
menu.add_cascade(label="Ayuda", menu=helpmenu)
helpmenu.add_command(label="Acerca de...", command=callback)

mainloop()
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

Barras de Herramientas

```
from Tkinter import *

root = Tk()

def callback():
    print "called the callback!"

# create a toolbar
toolbar = Frame(root)

b = Button(toolbar, text="nuevo", width=6, command=callback)
b.pack(side=LEFT, padx=2, pady=2)

b = Button(toolbar, text="abrir", width=6, command=callback)

b.pack(side=LEFT, padx=2, pady=2)

toolbar.pack(side=TOP, fill=X)

mainloop()
```

Simple Ventana de Diálogo

```
from Tkinter import *

class MyDialog:

    def __init__(self, parent):

        top = self.top = Toplevel(parent)

        Label(top, text="Valor").pack()

        self.e = Entry(top)
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
        self.e.pack(padx=5)

        b = Button(top, text="OK", command=self.ok)
        b.pack(pady=5)

    def ok(self):

        print "El valor es: ", self.e.get()

        self.top.destroy()


root = Tk()
Button(root, text="Hola!").pack()
root.update()

d = MyDialog(root)

root.wait_window(d.top)
```

Ejercicio 1

Haciendo uso de la Documentación de Tkinter crear un programa que abra una base de datos SQLITE, seleccione una tabla y recorra sus registros.

INTERFACES GRÁFICAS CON PYTHON Y GTK

Definición

Ejemplo #1 - Usando el terminal

```
>>> import pygtk
>>> pygtk.require('2.0')
>>> import gtk
>>> window = gtk.Window(gtk.WINDOW_TOPLEVEL)
>>> window.set_title('Hola pyGTK!')
>>> window.show()
```

Ejemplo #2 - Escribiendo un script de PyGTK

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk

class HelloWorld:

    # Esta es una función de llamados. Los argumentos de datos son ignorados
    # en este ejemplo. Mas llamados a continuación
    def hello(self, widget, data=None):
        print "Hello World"

    def delete_event(self, widget, event, data=None):
        # Si devuelve FALSE en la manejador de señales de "delete_event",
        # GTK emitirá la señal de "destroy". Devolviendo TRUE significa que
        # usted no desea que su ventana sea destruida.
        # Esto es especialmente útil para generar un mensaje emergente de
        # 'Esta seguro que quiere salir?'
        print "delete event occurred"
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
    # Cambie FALSE por TRUE y la ventana principal no será destruida por
    # el evento "delete_event".
    return False

  def destroy(self, widget, data=None):
    print "destroy signal occurred"
    gtk.main_quit()

  def __init__(self):
    # crea una nueva ventana
    self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)

    # When the window is given the "delete_event" signal (this is given
    # by the window manager, usually by the "close" option, or on the
    # titlebar), we ask it to call the delete_event () function
    # as defined above. The data passed to the callback
    # function is NULL and is ignored in the callback function.
    self.window.connect("delete_event", self.delete_event)

    # Here we connect the "destroy" event to a signal handler.
    # This event occurs when we call gtk_widget_destroy() on the window,
    # or if we return FALSE in the "delete_event" callback.
    self.window.connect("destroy", self.destroy)

    # Sets the border width of the window.
    self.window.set_border_width(10)

    # Creates a new button with the label "Hello World".
    self.button = gtk.Button("Hello World")

    # When the button receives the "clicked" signal, it will call the
    # function hello() passing it None as its argument.  The hello()
    # function is defined above.
    self.button.connect("clicked", self.hello, None)

    # This will cause the window to be destroyed by calling
    # gtk_widget_destroy(window) when "clicked".  Again, the destroy
    # signal could come from here, or the window manager.
    self.button.connect_object("clicked", gtk.Widget.destroy, self.window)

    # This packs the button into the window (a GTK container).
```


Av. Andrés Bello, Torre BFC, Piso 16, Municipio Libertador, Caracas – Venezuela
Master: (+58 212) 597.45.90 – www.cnti.gob.ve

```
        self.window.add(self.button)

        # The final step is to display this newly created widget.
        self.button.show()

        # and the window
        self.window.show()

    def main(self):
        # All PyGTK applications must have a gtk.main(). Control ends here
        # and waits for an event to occur (like a key press or mouse event).
        gtk.main()

# If the program is run directly or passed as an argument to the python
# interpreter then create a HelloWorld instance and show it
if __name__ == "__main__":
    hello = HelloWorld()
    hello.main()
```