Manejo de Datos

Ya que se tiene la confianza en que los archivos de entrada están limpios y en un formato conveniente para su uso, veremos una breve introducción a la carga y manejo de datos en una base de datos convencional.

ETL en PostgreSQL

La base a utilizar es PostgreSQL. Una base de datos relacional, Open Source y bastante poderosa. Se puede bajar e instalar siguiendo las instrucciones en `http://www.postgresql.org/download/`[6].
Una vez instalado[7], se puede acceder con el usuario por default:

```
sudo -u postgres psql postgres
```

Una vez adentro, se puede cambiar la contraseña por default del usuario postgres:

```
/*Cambio de Password*/
\password postgres
```

Posteriormente se creará una base y se cargarán los datos[8]:

```
/*Creacion de la base de datos*/
\password postgres
```

A continuación, veremos cómo crear una base de datos y una tabla y se subirán los datos de Hits Informativo:

```
/*Creacion de la base de datos con encoding UTF8*/
create database cdmx encoding='utf-8';
\connect cdmx;
```

[6]Es recomendable estudiar la documentación para referencias específicas: (`http://www.postgresql.org/docs/9.4/interactive/index.html`).

[7]Aunque en este curso no se tocó el tema, la instalación más recomendada en linux es compilar PostgreSQL, definir correctamente el espacio de almacenamiento, y configurar correctamente el usuario postgres desde el sistema operativo.

[8]El lenguaje utilizado es SQL. Si se requiere revisar algún concepto, PostgreSQL ofrece un apartado en su documentación: `http://www.postgresql.org/docs/9.4/interactive/sql.html`

IT, Commitment Unlimited

```
--Creacion de dos schemas: upload y datoscdmx
create schema upload;
create schema datoscdmx;
--Creacion de tabla del schema upload
create table upload.datos (hits varchar(20),concepto
    text,fecha varchar(20));
--Validamos:
\l
\dn
```

En el apartado anterior se crearon la base y el schema. En Postgres, se pueden tener varias bases de datos, y cada base puede tener varios schemas. Y en los schemas es donde se alojan las tablas. Al final se validó listando las bases de datos con `\l` y los schemas con `\dn`. Este tipo de instrucciones (que no forman parte de SQL), permiten ejecutar instrucciones generales, de consultas, de entrada / salida e informativas. Para una lista completa, ver `\?`.
NOTA:Para no teclear siempre schema.tabla, se puede establecer en el path del schema, el orden en el cual se va a buscar una tabla en los schemas existentes.
Ahora se va a cargar la información a la base. Al momento de cargar, hay que considerar:

- qué tanto tiempo nos puede lleva la carga de grandes volúmenes de datos, y
- qué tantos cambios / limpieza tenemos que realizar con los mismos

. En muchos casos, la velocidad de la red es un factor determinante para el tiempo de carga. Por tanto, nos importa subir los datos lo más rápido posible y procesarlos en la base. Y generalmente, el engine de la base será lo suficientemente rápido para procesar los datos adquiridos, por lo que se puede aprovechar la velocidad para subir una versión de los datos *en sucio* tal cual se encuentran a un esquema separado, y de ahí importar los datos a otro esquema, siendo *limpios* mediante consultas.
Por esta razón se creó la tabla *datos* en el esquema *upload*. Nótese que son únicamente campos de caracteres (incluidos los hits y la fecha) y el concepto es de tipo text, para garantizar que se suba todo lo que trae el archivo. Adicionalmente, la base se creó desde un inicio con encoding UTF-8, para soportar bastantes caracteres adicionales a los estándar en ASCII.

```
--Importacion de los datos
copy upload.datos from
    '<directorio>/Hits_Informativos....csv' delimiter
    ','
csv;
```

```
select * from upload datos limit 20;
--Creacion de la tabla correcta en datoscdmx;
drop table if exists datoscdmx.datos;
create table datoscdmx.datos as
        select
            regexp_replace(hits,'^[0-9]','')::integer
            as hits,
        concepto,
        to_date(fecha,'YYYY-MM-DD') as fecha
        from upload.datos;
```

Ya que se tiene la tabla, se van a crear los índices respectivos. Esto es importante siempre que se trabaja con datos de buen tamaño, ya que reduce el tiempo de ejecución de una consulta. Una buena medida es indizar todo sobre lo que se realiza una consulta frecuentemente. La sintaxis para construir un índice es la siguiente:

```
--Generacion de Indices
create index <nombre_indice> on <tabla> (<columna(s)>)
[where <condicion(es)]
```

Window Functions

Las funciones ventana permiten realizar cálculos a registros que están relacionados con el registro actual. Es parecido a una función de agregación, sin embargo en una función de ventana los registros no se agrupan en un solo registro. La sintaxis es:

```
--Funcion de Ventana
select <columnas>, <funcion agg(columna)> over
    (partition by <col>) from
<tabla>;
```

La referencia para las Funciones de Ventana se encuentra en: `http://www.postgresql.org/docs/9.1/static/tutorial-window.html`

Ejercicios

Sobre la tabla cargada *datoscdmx.datos*, realizar lo siguiente:

1. ¿Cuál es el número total de hits en febrero 2014?
2. ¿Cuál es el concepto que más hits tuvo en todo el periodo (2006-2015)?
3. Comparar, para cada rubro de Uniformes, los hits para los últimos tres años (columnas: concepto, hits2013,hits2014,hits2015)
4. Realizar una consulta para generar sumarizados de hits por concepto

5. Obtener los tres conceptos más vistos por año, utilizando Window Functions y la función rank()
6. ¿Cuál de los conceptos se ha mantenido
7. Generar índices para: concepto, fecha únicamente de 2015 y concepto-fecha
8. Repetir los ejercicios anteriores.