TP Exportación/Importación de Esquemas y Datos de una Base de Datos Postgresql

Migración de una Base de Datos

Si disponemos de un DBMS fuente que posee cierta base de datos relacional y deseamos "transferirla" hacia otro DBMS destino (quizás el mismo pero en otro momento) debemos encargarnos de migrar dos cosas:

- El esquema de la base de datos relacional (catálogo o esquema)
- La instancia de dicho esquema (los datos propiamente dichos)

Una vez que se obtienen ambas, desde el DBMS destino se procede a "importarlas". Existen distintas formas para poder realizar esto, y consisten en utilizar alguno o varios de los siguientes utilitarios:

- ***pg_dump***: Permite generar el esquema de una base de datos o tabla y puede extraer tuplas. Toda la información se vuelca en un archivo que luego puede ejecutarse con psql.
- ***Copy..From, Copy..To***: Permite exportar o importar los datos que contiene el esquema o una tabla.

  Este comando puede utilizarse sólo con privilegios de superuser. Para los privilegios que tenemos asignados podemos utilizar por consola o mediante un script, un comando equivalente: `\copy`
- ***Pg_dumpall*:** Extrae todas las bases de datos de una cluster a un archivo externo que luego podrá ejecutarse con psql
- ***Pg_restore*:** Restaura una base de datos a partir de un archivo que proviene de un pg_dump (no los de tipo texto).

➢ PG_DUMP:

La sintaxis de uso de este comando se encuentra en:

http://www.postgresql.org/docs/9.2/static/app-pgdump.html

pg_dump se ejecuta desde el file system, no desde el promt de postgres.
Para exportar varias tablas, es necesario usar la opción –t tantas veces como tablas se quieran exportar.
Es importante destacar que los datos exportados con pg_dump a un archivo de texto, deben restaurarse usando **psql** (NO pg_restore), ya que la salida es un script sql.

➢ COPY

Para importar datos a nuestras tablas a partir de archivos de texto o csv (comma value separated), Postgresql utiliza el comando COPY ..FROM.

Existe también el comando COPY…TO que, de manera análoga, exporta el contenido de una tabla a un archivo estándar del file system.

La sintaxis es la siguiente;

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | PROGRAM 'command' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]

COPY { table_name [ ( column_name [, ...] ) ] | ( query ) }
    TO { 'filename' | PROGRAM 'command' | STDOUT }
    [ [ WITH ] ( option [, ...] ) ]

donde option puede ser:

    FORMAT format_name
    OIDS [ boolean ]
    DELIMITER 'delimiter_character'
    NULL 'null_string'
    HEADER [ boolean ]
    QUOTE 'quote_character'
    ESCAPE 'escape_character'
    FORCE_QUOTE { ( column_name [, ...] ) | * }
    FORCE_NOT_NULL ( column_name [, ...] )
    FORCE_NULL ( column_name [, ...] )
    ENCODING 'encoding_name'
```

Alguna de las opciones que vamos a utilizar

table_name

Nombre de una tabla existente (opcionalmente prefijada con el nombre del esquema)

column_name

Lista opcional de columnas a copiar. Si no se especifican, se copian todas las columnas de la tabla.

query

Una consulta (SELECT o VALUES) cuyo resultado será copiado. Esta consulta debe estar encerrada entre paréntesis.

filename

El nombre (incluyendo el path) del archivo de entrada o salida. El path del archivo de entrada puede ser absoluto o relativo. El path del archivo de salida, debe ser absoluto.

`FORMAT`

Selecciona el formato de los datos a ser leídos o escritos: `text`, `csv` (Comma Separated Values), o `binary`. El default es `text`.

`DELIMITER`

Determina el caracter que separa las columnas dentro de cada fila. En los archivos de texto, el default es el tab, en los CSVs, la coma. Tiene que ser un único de character de un byte. Esta opción no se permite para el format `binary`.

`HEADER`

Especifica que el archivo contiene encabezado con los nombres de las columnas. En modo salida, la primera línea contiene los nombres de las columnas de la tabla y en la entrada, la primera línea sólo se ignora. Esta opción sirve únicamente para el formato `CSV`.

`QUOTE`

Especifica qué carácter se usa como comillas cuando el valor del dato está encomillado. El default es la comilla doble. Debe ser un carácter de un byte. Esta opción sirve únicamente para el formato `CSV`.

IMPORTANTE: como para usar COPY hay que tener privilegios de Superuser, vamos a utilizar la opción \COPY por consola, en la cual se pueden utilizar los parámetros antes mencionados

Por ejemplo

\COPY –u username nombretabla FROM arch.csv csv header delimiter ';'

El resto de los parámetros puede consultarse en:

http://www.postgresql.org/docs/9.3/static/sql-copy.html

- **PG_DUMPALL**

http://www.postgresql.org/docs/9.3/static/app-pg-dumpall.html

- **PG_RESTORE**

http://www.postgresql.org/docs/9.3/static/app-pgrestore.html

Como este TP se debe hacer por consola (no usando el PGAdmin ni DBVisualizer), indicamos aquí cómo conectarnos al server Pampero si estamos fuera del ITBA desde linux:
En una terminal, escribir:

```
ssh nombreusuario@pampero.itba.edu.ar
```

Para conectarse a Postgres:

```
psql –h bd1.it.itba.edu.ar –U nombreusuario PROOF
```

Ejercicio 1

El objetivo de este ejercicio es exportar un dataset y volverlo a importar a otro conjunto de tablas. Para eso trabajaremos con línea de comandos desde Pampero. En este caso vamos a exportar el dataset **LIBRERÍA** a un archivo .sql que contendrá el script para su creación.

Debemos utilizar para ello el comando pg_dump indicando cuáles tablas queremos exportar y el nombre del archivo .sql en dónde se encontrará el script.

1.1)

Para poder realizar la exportación del dataset debemos indicar al comando **pg_dump** los siguientes parámetros:

- -h: Host al que nos conectaremos, en nuestro caso bd1.it.itba.edu.ar
- -p: Port (si no es el default 5432)
- -U: Nombre de usuario
- Base de datos de la que extraeremos las tablas, en nuestro caso PROOF.
- -t: tablas que conforman el dataset Librería, en nuestro caso son: EDITORIAL, LIBRO, FACTURA y DETALLE_FACTURA. Para poder exportar dichas tablas debemos enumerarlas explícitamente repitiendo cada vez el parámetro -t.

Ejecutaremos entonces desde el prompt de Pampero

```
pg_dump –h bd1.it.itba.edu.ar -U nombreusuario -t detalle_factura -t
factura -t libro -t editorial –f libreria.sql PROOF
```

Si todo funcionó correctamente, debería haberse creado el archivo **librería.sql**.

1.2)

Utilizando el script **librería.sql** generado en el punto anterior, vamos a crear otro dataset similar al LIBRERÍA sólo que en este caso los nombres de las tablas serán: EDITORIAL2, LIBRO2, FACTURA2 y DETALLE_FACTURA2.

Abrirlo y revisar su contenido. Debería tener los scripts necesarios para crear las 4 tablas.

Reemplazar los nombres de las tablas por su homólogo + 2. Es decir, por ejemplo, buscar todas las apariciones de la palabra LIBRO y reemplazarlo por LIBRO2. Sus apariciones estarán en el **CREATE TABLE**, en **ALTER TABLE** o también en el **COPY**.

El script debería quedar similar a:

```
CREATE TABLE detalle_factura2 (
    codfact integer NOT NULL,
    nrolinea integer NOT NULL,
    isbn character varying(8) NOT NULL,
    cantidad integer
);

ALTER TABLE detalle_factura2 OWNER TO salerno;

--
-- Name: editorial2; Type: TABLE; Schema: salerno; Owner: salerno
--

CREATE TABLE editorial2 (
    codedit character varying(3) NOT NULL,
    direccion character varying(30),
    nombre character varying(20),
    pais character varying(15)
);


ALTER TABLE editorial2 OWNER TO salerno;

--
-- Name: factura; Type: TABLE; Schema: salerno; Owner: salerno
--

CREATE TABLE factura2 (
    codfact integer NOT NULL,
    fecha date,
```

No olvidar reemplazar TODAS las apariciones incluyendo los nombres restricciones (constraints).

Grabar los cambios.

1.3)

Crearemos ahora las tablas con sus datos corriendo el script modificado en el punto anterior.
Para ello ejecutamos desde el prompt del sistema operativo:

```
psql -h bd1.it.itba.edu.ar -U nombreusuario -f libreria.sql PROOF
```

Verificar que se hayan creado las nuevas tablas y que las mismas cuenten con sus datos.

Ejercicio 2

Se quiere exportar el ***Dataset: PEDIDOS*** pero en este caso, exportaremos la estructura separada de los datos ya que queremos guardarlos en formato **csv** (comma separated values).
Debido a esto, la exportación la realizamos en dos pasos (ítems 2.1 y 2.2)

2.1)

Exportar las tablas de dataset PEDIDOS (proveedor, producto, cliente y pedido) sin exportar los datos. El comando es similar al del ejercicio 1.1 pero se debe utilizar el parámetro **-s**.
Volcar el resultado en el archivo **pedidos.sql.**

2.2)

Ahora exportaremos los datos de las tablas en formato csv (comma separated values). Para ello es necesario ejecutar el comando COPY. Debido que no somos administradores del sistema, debemos usar el comando \COPY.

Este comando puede ejecutarse dentro de la línea de comando de Posgres (psql) o creando un script y luego ejecutarlo con el parámetro -f y el nombre del archivo .sql que contiene los comandos.

El comando \COPY …TO debe ejecutarse una vez por cada una las tablas.
Se muestra aquí el comando para realizar la exportación de los datos de la tabla proveedor:

```
\copy proveedor to proveedor.csv header delimiter ';' csv;
```

Repetir el proceso para el resto de las tablas: PRODUCTO, CLIENTE y PEDIDO.
Una vez creado editar los archivos y verificar que se hayan creado con los atributos solicitados.

2.3)

Volvemos a crear un dataset similar al PEDIDOS. Siguiendo el mismo procedimiento que el utilizado en el ejercicio **1.2**, crear las tablas PRODUCTO2, PROVEEDOR2, CLIENTE2, PEDIDO2

2.4)

Utilizando el comando \COPY… FROM insertar los datos en cada una de las tablas creadas en el ítem **2.3**.

Ejercicio 3

Dataset: EXAMEN

El objetivo de este ejercicio es migrar los datos desde el formato xml, a la base de datos.

Los archivos xml para este ejercicio, se encuentran en CAMPUS en ***datos_examen.zip***.

3.1)

Crear un dataset con las mismas tablas que el dataset EXAMEN diferenciando sus nombres con un número 2 tal como se explicó en el ejercicio **1.2**. Asegurarse que las nuevas tablas estén vacías.

3.2)

Descomprimir el archivo ***datos_examen.zip***. Los nombres de los archivos descomprimidos se corresponden con los de las tablas. Analizar la estructura de cada uno de ellos.

3.3)

Se pide transformar mediante una plantilla **xslt**, los archivos xml a formato **texto** en dónde los datos de las columnas estén delimitados por ';'.
http://www.w3schools.com/xsl/xsl_intro.asp

La salida de este ejercicio deben ser 4 archivos de texto con el siguiente contenido:

Alu.txt

```
12000;Andrea Garcia
12500;Pedro Lima
11500;Ana Salina
10500;Sol Roy
```

Materia.txt

```
50;Programacion I
20;Estructura y Algoritmos
80;Base de Datos I
10;Programacion IV
```

Profesor.txt

```
533;Anibal Solano;12/10/1960
239;Martin Knuth;22/03/1974
448;Sara Warxel;17/08/1980
329;Paula Lorca;12/10/1960
```

Examen.txt

```
12000;239;2;03/03/1997;80;111
12000;533;3;15/03/1997;20;676
12500;533;4;15/03/1997;20;676
12500;239;5;03/03/1997;80;111
12000;239;8;21/03/1998;80;133
11500;533;9;15/03/1997;20;676
11500;239;6;03/03/1997;10;144
10500;239;8;03/03/1997;10;144
10500;533;2;03/03/1997;20;144
11500;533;7;03/03/1997;80;111
12000;448;6;21/03/1998;10;887
12000;448;8;21/03/1998;50;888
11500;448;10;08/04/1999;50;900
```

***Sugerencia**: para que la salida sea en formato texto y no tenga el encabezado xml, en xslt, se puede utilizar el elemento **output** indicando su atributo **method =”text**”.*

http://www.w3schools.com/xsl/el_output.asp

Recordatorio: Para parsear documentos xml usando xml schema se utiliza el siguiente comando:

java net.sf.saxon.Transform -s:*archivo*.xml -xsl:*plantilla*.xsl -o:*salida*.txt

3.4)

Utilizar el comando **\copy** para importar los datos generados en el punto anterior, a las tablas generadas en **3.1**.

Verificar que las siguientes sean las tuplas que están insertadas en las tablas ALU2, MATERIA2, PROFESOR2 y EXAMEN2:

ALU2	
Legajo	nombre
12000	Andrea Garcia
12500	Pedro Lima
11500	Ana Salina
10500	Sol Roy

MATERIA2	
Código	nombre
50	Programacion I
20	Estructura y Algoritmos
80	Base de Datos I
10	Programacion IV

(fecha con formato DD/MM/YYYY)

PROFESOR2		
legProf	nombre	antiguedad
533	Anibal Solano	12/10/1960
239	Martin Knuth	22/03/1974
448	Sara Warxel	17/08/1980
329	Paula Lorca	12/10/1960

(fecha con formato DD/MM/YYYY)

EXAMEN2					
legajo	legProf	nota	Fecha	Codigo	nroActa
12000	239	2	03/03/1997	80	111
12000	533	3	15/03/1997	20	676
12500	533	4	15/03/1997	20	676
12500	239	5	03/03/1997	80	111
12000	239	8	21/03/1998	80	133
11500	533	9	15/03/1997	20	676
11500	239	6	03/03/1997	10	144
10500	239	8	03/03/1997	10	144
10500	533	2	03/03/1997	20	144
11500	533	7	03/03/1997	80	111
12000	448	6	21/03/1998	10	887
12000	448	8	21/03/1998	50	888
11500	448	10	08/04/1999	50	900