Sistemas de Gestión de Bases de Datos

Guía Práctica 3
Motor Relacional

Ingeniería Informática
Universidad Nacional de Avellaneda

1. Introducción

Los objetivos de esta práctica son:

- Crear una base de datos relacional
- Realizar ABM de registros
- Comprender nociones básicas de SQL
- Integrar python y postgres

La bibliografía utilizada es:

1. *Manuales de postgres* (las referencias serán respecto a secciones de la versión 9.5: *postgresql-9.5-A4.pdf*)

2. Actividades preliminares

En esta práctica van a trabajar con el motor de base de datos relacional "postgres", para lo cual van a tener que instalarlo. Además será necesario que instalen el cliente "pgAdmin", la librería de interface python-postgres "psycopg2" y la librería de gráficos geográficos "geopandas"

3. Ejercicios

3.1. Ejercicio 1

Importar la base de datos del script world.sql:

1. Crear la base de datos "world" (sección 1.3)
2. Llenar la base de datos con los datos del script world.sql (sugerencia: usar psql -f)
3. Verificar la importación ejecutando una consulta desde algún cliente (pgAdmin, psql, etc.).

Referencia de cliente pslq: `https://www.postgresql.org/docs/devel/static/app-psql.html`

3.2. Ejercicio 2

Realizar las siguientes consultas en SQL.

Simples sobre una sóla tabla

1. La población de Argentina
2. Todos los continentes (sin repeticiones)
3. Nombres de los países de América del Sur con más de 15 millones de habitantes
4. Nombre y producto bruto de los diez países con mayor producto bruto (gnp)
5. Forma de gobierno y cantidad de paises con dicha forma de gobierno ordenados por cantidad de modo descendente (sugerencia: agrupar por forma de gobierno y contar)
6. Los nombres de los continentes con sus respectivas superficies ordenados de forma descendentes por superficie
7. Los continentes y la cantidad de paises que los componen de aquellos continentes con más de 15 países Idem g) pero que los países que se tengan en cuenta tengan una población de más de 20 millones de personas

Subqueries

1. ¿Qué hace la siguiente consulta?

```
select name, lifeexpectancy
from country
where lifeexpectancy = (select min(lifeexpectancy) from country)
```

2. Nombre del pais y la expectativa de vida de el/los países con mayor y menor expectativa de vida
3. Nombre de los países y año de independencia que pertenecen al continente del país que se independizo hace más tiempo
4. Nombres de los continentes que no pertenencen al conjunto de los continentes más pobres

Joins

1. Los paises y las lenguas de los paises de Oceania
2. Los paises y la cantidad de lenguas de los paises en los que se habla más de una lengua (ordenar por cantidad de lenguas de forma descendente)
3. Las lenguas que se hablan en el continente más pobre (sin considerar a Antarctica)
4. Los nombres de los países y sus respectivas poblaciones calculadas de formas distintas: 1) de acuerdo al campo de la tabla country y 2) como suma de las polaciones de sus ciudades correspondientes, ademas se pide calcular el porcentaje de población urbana (de las ciudades), ordenar por porcentaje de modo descendente

3.3. Ejercicio 3

Crear la siguiente tabla de estadísticas (utilizando la información de las tablas importadas):

- Nombre: **stats**
- Campos:
  - **countrycode**: primary key/forign key (tabla country)
  - **cant_lenguas**: cantidad de lenguas que se hablan en el país
  - **pop_urbana**: cantidad total de habitantes en las ciudades del país

Referencias

- `https://stackoverflow.com/questions/6083132/postgresql-insert-into-select`
- `https://stackoverflow.com/questions/6256610/updating-table-rows-in-postgres-using-subquery`

3.4. Ejercicio 4

El archivo "top-1m.csv" contiene la información ordenada del 1er. millón de sitios de internet con mayor tráfico (la medición corresponde al año 2012). La estructura del archivo es la siguiente:

- Cada línea es de la forma: $< nro.\ de\ orden >, < dominio >$
- Cada dominio es de la forma: XXXXX.YYY.ZZ

Se pide crear la siguiente tabla

- Nombre: **sitio**
- Campos:
  - **id**: int (es el número de orden que aparece en cada línea de la archivo)/primary key
  - **entidad**: varchar (1ra. parte del dominio: hasta el 1er. punto)
  - **tipo_entidad**: varchar (2da. parte del dominio)
  - **país**: varchar (3ra. parte dela dominio)
  - **countrycode**: foreign key a **country.code**

Hacer un programa en python que haga lo siguiente:

1. Leer los campos "code2" y "code" de la tabla country y generar un diccionario de la forma: d[code2] = code
2. Para cada línea de la archivo "top-1m.csv"separar el nro. de orden y el dominio
3. Separar el dominio en los tres (o dos) campos que lo componen
4. Hacer la inserción del registro correspondiente en la tabla "sitio" haciendo la traducción correspondiente del país

Referencia de interface python-postgres: `http://zetcode.com/db/postgresqlpythontutorial/`

3.5. Ejercicio 5

Los *índices* son estructuras de datos secundarias (o auxiliares) que permiten acceder a los datos de una tabla en menor tiempo.

La estructura de datos más popular para implementar índices se denomina *B-tree* (Ver `https://en.wikipedia.org/wiki/B-tree`).

1. De acuerdo a la tabla "sitio" del ejercicio anterior, analizar qué debería devolver la siguiente consulta:

```
select *
from sitio s1, sitio s2
where s1.countrycode = s2.countrycode
and s1.entidad like 'a%' and s2.entidad like 'b%'
limit 100
```

2. En general los motores de bases de datos relacionales proveen funcionalidades para analizar cómo se ejecuta una consulta (ver sección 14.1). Analizar la consulta anterior con EXPLAIN PLAN y luego ejecutarla
3. Crear un indice sobre la columna "countrycode" de la tabla "sitio" y repetir el análisis del punto 2
4. Ejecutar la consulta

3.6. Ejercicio 6

En base al ejemplo "prueba.py", realizar siguientes gráficos *Choropleth* (`https://en.wikipedia.org/wiki/Choropleth_map`):

1. Un mapa de la población mundial
2. Un mapa del producto bruto mundial
3. Un mapa de sitios web en el mundo