UNPSJB

LICENCIATURA EN SISTEMAS OPGCPI

BASES DE DATOS II

Trabajo Práctico 1

PL/pgSQL, procesamiento y optimizacón de consultas

Cátedra

Lic. Gabriel Ingravallo

Lic. Cristian Parisse

Integrantes:

Luciano Serruya Aloisi

29 de marzo de 2018

Índice

1. Carga masiva de la Base de Datos **2**
1.1. Modelos de avión 2
1.2. Trabajadores 3
1.3. Aviones 3
1.4. Reparaciones 3

1. Carga masiva de la Base de Datos

Agregar tuplas en forma masiva a las tablas modeloAvion y Avion de forma tal que por cada modeloAvion insertado se agreguen 10000 aviones y que la estadística V(capacidad, modeloAvion) >= 1000. Además por cada avión agregado agregar una reparación del mismo (con algún DNI de trabajador ya preexistente). Se debe presentar el script de la solucion.

Los scripts a los que se haga referencia se pueden encontrar en el directorio scripts/

1.1. Modelos de avión

Los modelos de avión utilizados para la carga masiva de la base de datos fueron tomados del *dataset* disponible en [2]. Los tipos de naves del *dataset* fueron extraídos mediante el *script* de Python **extraer_modelos.py**

A continuación, mediante otro *script* en Python (**generar_script.py**), generamos un *script* en SQL para hacer la carga de los modelos de avión a la base de datos (**script_carga_modelos.sql**). De esta forma, insertamos en la base el total de **2449** modelos de avión, siendo su clave primaria un número secuencial y su capacidad un valor aleatorio entre 50 y 2000.

```
aviones=# SELECT COUNT(*) FROM "modeloAvion";
 count
-------
  2449
(1 row)
```

Los valores cargados para los modelos de avión cumplen con la estadística V(capacidad, modeloAvion) ≥ 1000

```
aviones=# SELECT COUNT(DISTINCT capacidad) FROM "modeloAvion";
 count
-------
  1389
(1 row)
```

1.2. Trabajadores

Para la carga de los trabajadores, se utilizó el sitio `http://www.generatedata.com/` utilizando valores aleatorios para los DNIs y nombres de ascendencia latina disponibles en el repositorio de *GitHub* [1]. El script generado por el sitio se encuentra bajo el nombre de **script_carga_trabajadores.sql**

```
aviones=# SELECT COUNT(*) FROM trabajador;
 count
-------
   105
(1 row)
```

1.3. Aviones

Una vez ejecutados exitósamente los dos scripts de carga, se procede a insertar masivamente registros en la tabla de aviones.

Por cada modelo de avión existente en la base de datos, se cargan 10000 aviones, con una cantidad de horas de vuelo aleatoria entre 100 y 5000, y un año aleatorio entre 1970 y 2018. El *script* de carga correspondiente es **script_carga_aviones.sql**.

La función que realiza la carga de aviones está compuesto por dos bucles *FOR .. LOOP* anidados; el más externo itera sobre los modelos de avión, mientras que el interno sobre enteros entre 1 y 10000.

```
aviones=# SELECT COUNT(*) FROM avion;
  count
----------
 24490006
(1 row)
```

1.4. Reparaciones

Por último, se cargan las reparaciones de los aviones. El *script* correspondiente (**script_carga_reparaciones.sql**) consiste también de dos bucles *FOR..LOOP* anidados; el primero itera sobre todos los aviones existentes, mientras que el segundo itera sobre un rango de valores aleatorios entre 1 y 10, que corresponden a la cantidad de reparaciones registradas para ese avión (puede haber registrado entre 1 a 10 repa-

raciones). El DNI del trabajador también va a ser un valor aleatorio entre todos los registrados

```
SELECT INTO dni_trabajador dni FROM trabajador ORDER BY random()
    LIMIT 1;
```

La fecha de inicio de la reparación se obtiene aleatoriamente entre un intervalo de tiempo, empezando desde 01/01/2010, hasta el 01/03/2018. Por último, la fecha de fin de reparación, que por más que no es obligatoria, se obtiene a partir de sumar una cantidad de días aleatoria a la fecha de inicio (entre 2 y 30 días).

Debido a los valores aleatorios, puede ocurrir que se intenten insertar dos veces una misma tupla (una reparación que tenga la misma clave primaria, compuesta del DNI del trabajador, el ID del avión, y la fecha de inicio de la reparación), por lo tanto se decidió que, frente a un intento de insertar un registro repetido, no hacer nada (no insertar la tupla conflictuante ni actualizar la existente).

```
INSERT INTO "trabajadorReparacion" VALUES(dni_trabajador,
   nro_avion, fecha_inicio, fecha_fin, tipo_falla) ON CONFLICT DO
    NOTHING;
```

```
aviones=# SELECT COUNT(*) FROM "trabajadorReparacion";
   count
-----------
 134712980
(1 row)
```

Referencias

[1] 2012. Dataset de apellidos y nombres. https://github.com/marcboquet/spanish-names.

[2] 2013. Dataset de accidentes aéreos, 1908-2013. https://opendata.socrata.com/Government/Airplane-Crashes-and-Fatalities-Since-1908/q2te-8cvq.