Prácticas de Estructuras de Datos

David Arroyo Guardeño

Escuela Politécnica Superior de la Universidad Autónoma de Madrid

Información general

- ✓ Correo $\leftrightarrow$ xxx.yyy@estudiante.uam.es
- ✉ [EDAT]
- ✓ Nota: $0.7 \times N_T + 0.3 \times N_P$
- ☠ COPIAS

Requisitos de las entregas

Retraso Penalización de 1 punto por día de retraso

Entrega moodle Límite: una hora antes del comienzo de la primera sesión de la siguiente práctica

Nombre entregable Gxxx_Pyy_z.tgz
en caso contrario, ↓ 1 punto

Contenido del fichero

1. Fichero de texto: nombre alumnos, email, grupo y fecha
2. Documentación en formato pdf
   - ✗ Respuesta a preguntas breves
   - ✗ Cómo se ha realizado cada ejercicio
   - ✗ Documentación ejercicio final:
3. Listado de código fuente documentado
   - ✗ Programación modular y estructurada
   - ✗ Makefile
   - ☠ Compilación no correcta: 0!!!

Práctica 1

1. Crear un diagrama E-R
2. Optimización del primer diseño
3. Consultas SQL

Gestión de un catálogo de libros I

1. Libro: uno o varios autores, una o varias ediciones, uno o varios idiomas, tapa dura o bolsillo
2. Cada edición puede tener un editor o editores distintos; cada edición puede tener un precio distinto
3. Existen ofertas para un grupo de libros dentro de una cierto rango de fechas

Gestión de un catálogo de libros II

4. Existen usuarios fidelizados: identificador y número (único) de tarjeta de crédito
5. Los usuarios fidelizados tienen un descuento de un 10% (también en libros en oferta)
6. Hay registro de compras para cada usuario fidelizado. La compra puede ser en metálico o con tarjeta

Gestión de un catálogo de libros III

7. Los usuarios no fidelizados compran (en metálico o con tarjeta) y el registro se lleva sobre los artículos adquiridos

Consultas I

1. Dado un título, ¿Cuántas ediciones tiene? ¿En cuantos idiomas?
2. ¿Cuántos libros se han vendido de un autor dado?
3. ¿Cuántos libros de un autor dado se han vendido en oferta?
4. ¿Cuánto dinero se ha ganado vendiendo libros de un editor dado?

Consultas II

5. ¿Cuántos libros han comprado los usuarios fidelizados?
6. ¿Cuántos usuarios fidelizados han comprado libros en inglés?

```
sudo apt-get install postgresql postgresql-contrib
sudo -i -u postgres createuser edat -d -l
sudo -i -u edat createdb -U edat libreria
man createuser
man createdb
man dropuser
```

Comandos útiles de PostgreSQL

- ✓ **psql −U <usuario> <base de datos>** para conectar un usuario a una base de datos
- ✓ **\l** lista todas las bases de datos
- ✓ **\du** lista todos los usuarios
- ✓ **\d** muestra la estructura de una tabla (su descripción)
- ✓ **\q** para salir de psql
- ✓ **pgadmin3** para acceder en modo gráfico

Introducir un fichero en una tabla de la base de datos

```
\copy users from '/tmp/user.txt' CSV header delimiter E'\t' encoding 'ISO
    -8859-1' NULL as 'NULL'
```

- ✓ **CSV header** elimina la cabecera del fichero (los descriptores de columna)
- ✓ **with delimiter E'\t'** establece que las columnas están separadas mediante un tabulador
- ✓ **encoding 'ISO-8859-1'** establece que la codificacion del fichero de importación es 'ISO-8859-1'
  - ✗ Si no se usa la codificación adecuada puede haber problemas con la visualización de caracteres especiales
  - ✗ Para ver la codificación de un fichero desde la shell de Linux, basta hacer **file fichero.txt**
  - ✗ Por defecto PostgreSQL asume que el fichero está codificado siguiendo la norma UTF-8
  - ✗ Se puede cambiar la codificación de un fichero mediante iconv ⇒ **man iconv**

Realizar un backup de la base de datos y cargar desde un dump

```
pg_dump –U username –f backup.sql database_name
psql –d database_name –f backup_sql
```

Más comando útiles

```
# Mostrar un listado con todas la bases de datos
sudo -u postgres psql --list

# Realizar un backup
sudo -u postgres pg_dump [database_name] > dumpl.sql

# Eliminar una base de datos
sudo -u prostgres dropdb [database_name]

# Crear una base datos
sudo -u postgres createdb [new_databse]

# Restaurar una base de datos
sudo -u postgres psql [new_database] < dump.sql

##################################################

# Respaldar una unica tabla
sudo -u postgres pg_dump --table [table_name] [database_name] > dump.sql

# Restaurar solo una tabla
sudo -u postgres psql -f dump.sql [database_name]
```

Configuración de pgadmin I

Editar el fichero

/etc/postgresql/9.x/main/pg_hba.conf

Cambiar

```
#Database administrative login by Unix domain socket
local    all             postgres                                md5
```

por

```
#Database administrative login by Unix domain socket
local    all             postgres                                trust
```

Re-arrancar el servidor

sudo service postgresql restart

Hacer

Configuración de pgadmin II

```
psql –U postgres
ALTER USER postgres with password 'new password';
```

Volver a la configuración inicial del fichero de configuración de PostgreSQL

```
#Database administrative login by Unix domain socket
local    all                postgres                                md5
```

Finalmente

sudo service postgresql restart

Permitir hacer login a cualquier usuario en local y dado de alta en postgres

Editar pg_hba.conf

```
# Database administrative login by Unix domain socket
#sólo para postgres
#local    all             postgres                                md5
local    all             all                                     md5
```

Shell: comandos útiles I

Analizador de comandos shell: **ExplainShell**

- ✓ **head −1 LIBROS.txt**: muestra la primera fila del fichero, en este caso la cabecera CSV
- ✓ **cut −d $'\t' −f1−3,5 LIBROS.txt**: muestra las columnas 1 a 3 y la columna 5 del fichero; las columnas están separadas por un tabulador (el parámetro **−d** define el separador entre columnas)
- ✓ **paste −d $'\t' user_id.txt isbns_vendidos.txt > ventas.txt**: crea un nuevo fichero (ventas.txt) anexionando las columnas de los ficheros de entrada (en este caso sólo dos, pero pueden ser tantos como precisemos); las columnas se separan mediante tabulador
- ✓ **wc −l fichero.txt**: número de líneas de un fichero
- ✓ **find ./ −name *.txt −type f | xargs −I{} grep candidato {}**

Shell: comandos útiles II

- **find ./ −name ∗.txt −type f**: encuentra todo los ficheros con extensión txt y que estén contenidos en el directorio actual o en alguno de los directorios que éste contiene
- **xargs −I{}**: considera como argumento de entrada la salida proporcionada por el comando a la izquierda de la tubería
- **xargs −I{} grep candidato {}**: busca la palabra “candidato” en cada una de los ficheros que devuelve el comando **find**

Shell: comandos útiles III

- ✓ **sed –i 's/0000–00–00/1984–01–01/g' LIBROS.txt**: sustituir la cadena "0000-00-00" por "1984-01-01" en LIBROS.txt (el parámetro **–i** hace que **sed** guarde el resultado en el fichero de entrada)

```
cut –d $'\t' –f8 LIBROS_FINAL.txt | sed '1d' | sort | uniq –c | tr –s " "
```

- ✗ **cut –d $'\t' –f8 LIBROS_FINAL.txt**: extrae la columna 8 del fichero de entrada
- ✗ **| sed '1d'**: elimina la primera fila del resultado anterior
- ✗ **| sort**: ordena las filas obtenidas en el anterior paso

Shell: comandos útiles IV

- ✗ **| uniq -c**: en base al resultado de la operación previa, obtiene los elementos que son únicos y, además, muestra cuántas veces aparece cada uno de ellos
- ✗ **| tr -s " "**: sustituye múltiples ocurrencias de un espacio en blanco por una única ocurrencia

```
cat /dev/urandom | tr -cd "0-9" | fold -w 3 |
| xargs -I{} date -d "2016-09-28 + {} days" +"%d-%m-%Y"
```

- ✓ **cat /dev/urandom | tr -cd "0-9" | fold -w 3**: Genera aleatoriamente números de 3 dígitos
- ✓ **| xargs -I{} date -d "2016-09-28 + {} days" +"%d-%m-%Y"**: suma el número aleatorio a una fecha de partida

Comandos útiles SQL I

Además de los ya conocidos . . .

- ✓ Conversión de tipos (*casting*): **select 1::boolean;** (conversión de entero en booleano);
  select 1 as boolean; ; **select now()::date;** (muestra la hora actual en formato de fecha sin hora); **select 1.56::money;** (convierte un número a formato money)
- ✓ Generación de valores aleatorios: **select random();** ;
  select random()∗10:int8::money; (generación de tipo money de modo aleatorio)
- ✓ Fecha aleatoria:

```
select (now()+ interval  '1'  minute ∗ round(365∗(random()::numeric
 ),0)):: timestamp (0);
```

- ✓ Evitar problemas por duplicación de claves primarias en la importación de datos (☠: sólo si es estrictamente necesario)

Comandos útiles SQL II

```
insert into libros (select isbn,titulo,formato,idioma from
 libros_temp)on conflict do nothing;
```

- ✓ Lo anterior sólo funciona para versiones de PostgreSQL superiores a la versión 9.4.5

```
select * from (select *, row_number()over (partition by isbn order
    by titulo)
as row_number from libros_temp02)as rows where row_number = 1;
```

- ✓ Funciones de ventana en PostgreSQL: `https://www.postgresql.org/docs/current/static/functions-window.html`