# ESTRUCTURAS DE DATOS

**PRÁCTICA 2 - 2016/2017**

Lucía Asencio Martín y David García Fernández
Grupo 1201

# CAMBIOS REALIZADOS EN EL DISEÑO

Durante esta práctica, hemos cambiado el formato de algunas de nuestras tablas con el fin de hacer más correcta la base de datos.

- En primer lugar, hemos eliminado la columna “Precio final” de la tabla “Compra” porque, a pesar de simplificar las querys, no dejaba de ser una columna redundante cuyo contenido podía obtenerse a partir de las otras tablas. La query empleada para calcularlo se puede ver en compra.c, dividida en casos según si se aplica o no oferta al producto adquirido.
- Además hemos añadido, en la tabla de compra, una columna (“id”) que nos permite distinguir unívocamente esa venta de todas las demás. Esta columna es un tipo de dato serial, que va autoincrementándose para cada tupla que insertamos en la tabla. OJO: este id corresponde a la venta de un sólo ISBN, no es un “número de pedido” que corresponda a varias ediciones a la vez.
- Otro cambio en nuestras tablas ha sido eliminar la columna “número de edición” de la tabla edición, ya que no se hacía referencia a ella en ningún punto de la práctica y los datos proporcionados para poblar tablas no nos permitían dar valor a esta columna.
- Por último, para poder ajustar nuestras tablas al formato que requieren los programas c que debíamos entregar, hemos tenido que modificar la tabla usuario: hemos añadido las columnas full_name, screen_name y un tipo boolean (“borrado”) que nos indique si el usuario está activo (False) o no (True).

Así quedaron nuestras tablas:

| afectaoferta | |
|---|---|
| ofertaid | isbn |

| autor | |
|---|---|
| autornombre | dni |

| compra | | | | |
|---|---|---|---|---|
| isbn | nousuario | metodo | fecha | idcompra |

| edicion | | | | | | |
|---|---|---|---|---|---|---|
| isbn | editor | tapa | idioma | precio | titulonombre | fecha_ed |

| escribe | |
|---|---|
| titulo | dni |

| oferta | | | |
|---|---|---|---|
| descuento | id | fecha1 | fecha2 |

| titulo |
|---|
| nombre |

| usuario | | | | | |
|---|---|---|---|---|---|
| nousuario | ccard | screen_name | full_name | fecha | borrado |

# MÉTODO PARA POBLAR LAS TABLAS

Para poblar las tablas seguimos los consejos del documento que Santini subió a Moodle.

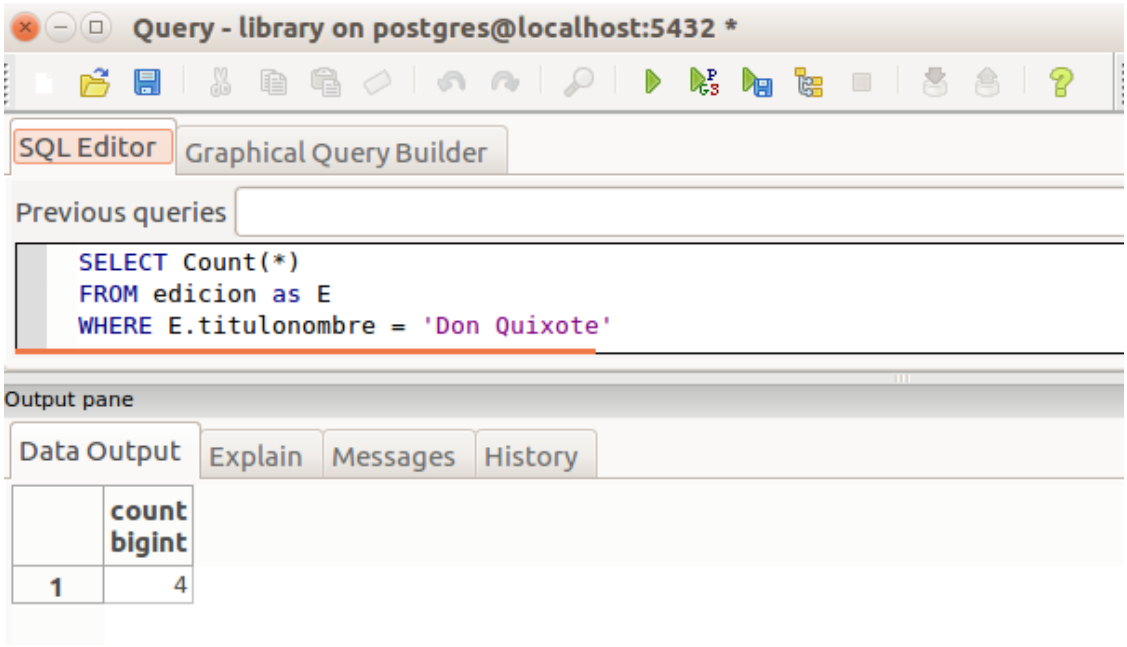
- En primer lugar, trabajando desde la terminal cambiamos el encoding de los ficheros con la información a UTF-8, y luego introdujimos todos los datos en tablas temporales cuyas columnas eran cadenas de caracteres, para luego modificarlas.
- De estas temporales hubo que eliminar información (por ejemplo, todas las tuplas de la tabla isbn_precio donde el precio estaba vacío, o todas las tuplas de libros donde la fecha era 0000).
- Como ISBN es una PK no podíamos tener un mismo ISBN asociado a distintos libros , así que con group by y con ayuda de la función min eliminamos también de las tablas temporales todos los ISBN repetidos.
- Eliminamos de la tabla de compra todas las tuplas que se asocian a un ISBN que no está en la tabla de edición.
- También damos valor a todas las columnas que están en nuestras tablas pero para las cuales no encontramos información en los ficheros: ya sea con valores constantes (por ejemplo, damos a la columna “metodo” de la tabla compra el valor de ccard), o variables (añadimos como “dni” a la tabla autor el número de fila en el que queda cada autor tras hacer un select distinct de todos los autores).
- Una vez tenemos en nuestras temporales toda la información necesaria para rellenar las tablas definitivas, tenemos que pasar nuestras cadenas de caracteres al tipo de dato que necesitamos y, por último, con muchos inner joins, vamos combinando la información de las tablas temporales para insertarla en las definitivas.

# CONSULTAS DE LA PRÁCTICA 1

**1)Dado un título, ¿Cuántas ediciones tiene? ¿En cuántos idiomas?**

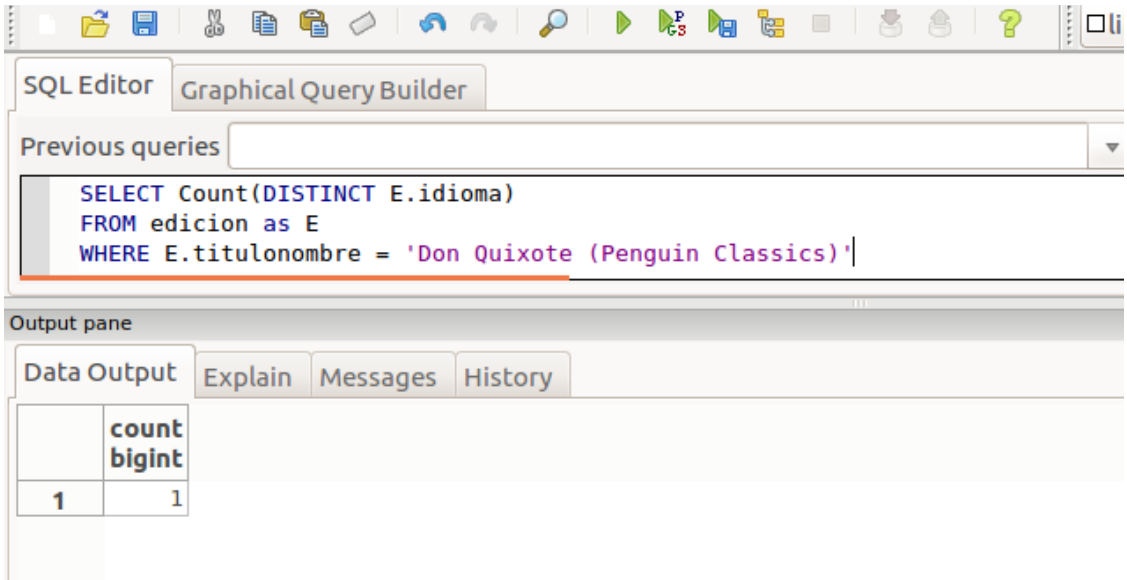
**a)**

```
SELECT Count(*)
FROM edicion as E
WHERE E.titulonombre = 'The Dumas Club'
```




**b)**

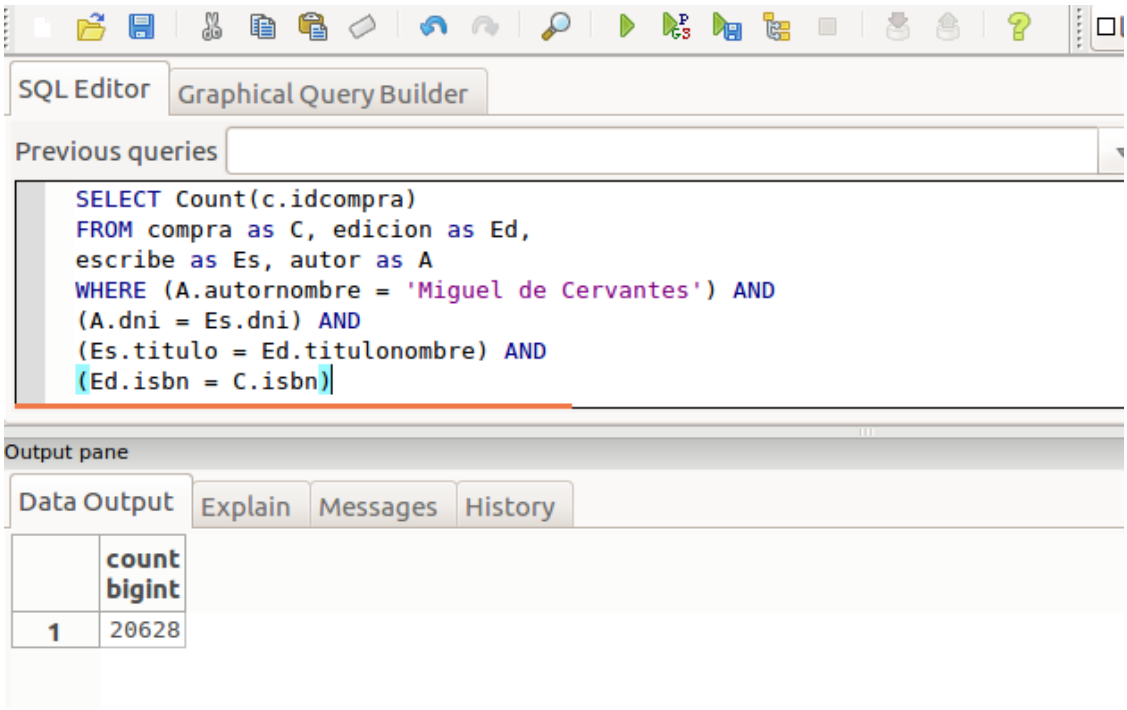
```
SELECT Count(DISTINCT E.idioma)
FROM edicion as E
WHERE E.titulonombre = 'Don Quixote (Penguin Classics)'
```




**2)¿Cuántos libros se han vendido de un autor dado?**

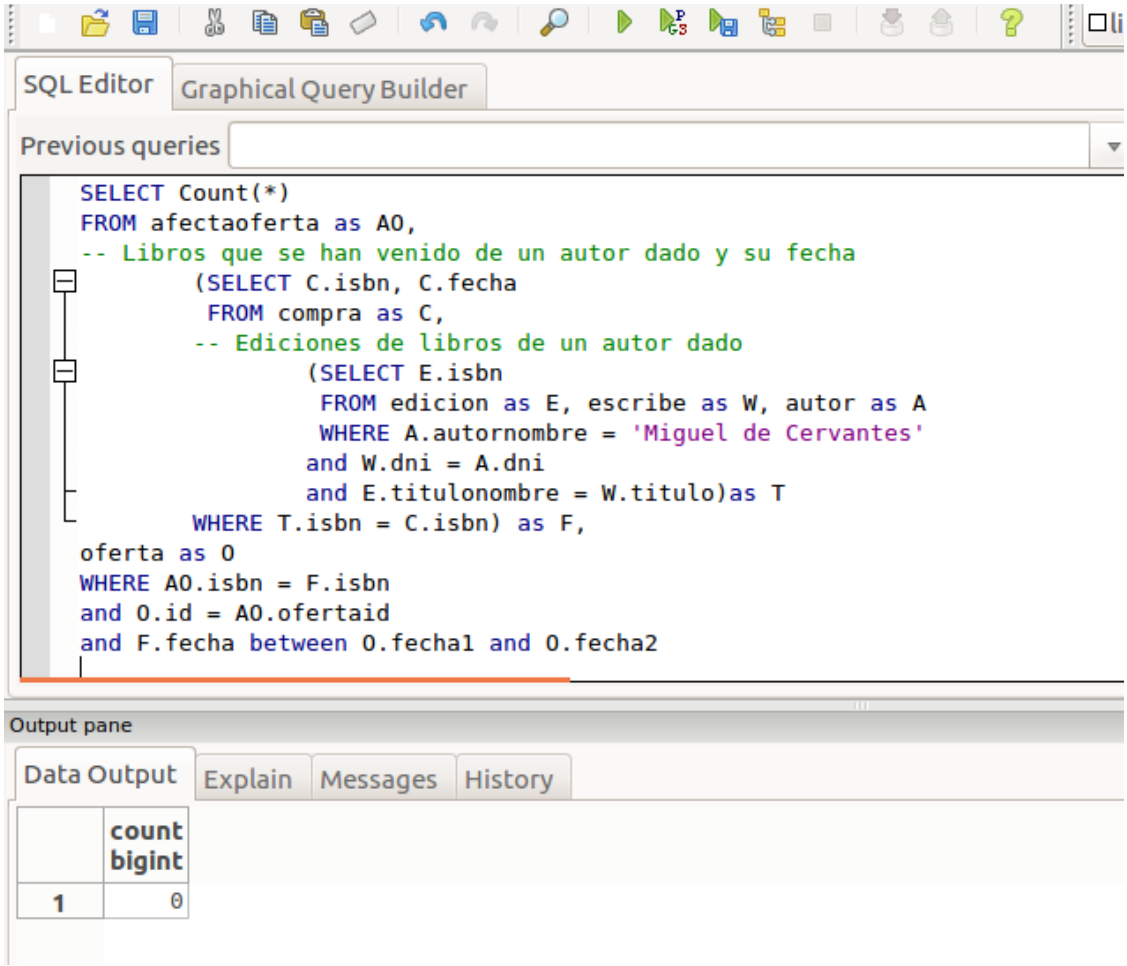
```
SELECT Count(c.idcompra)
FROM compra as C, edicion as Ed,
```

```
escribe as Es, autor as A
WHERE (A.autornombre = 'Miguel de Cervantes') AND
(A.dni = Es.dni) AND
(Es.titulo = Ed.titulonombre) AND
(Ed.isbn = C.isbn)
```



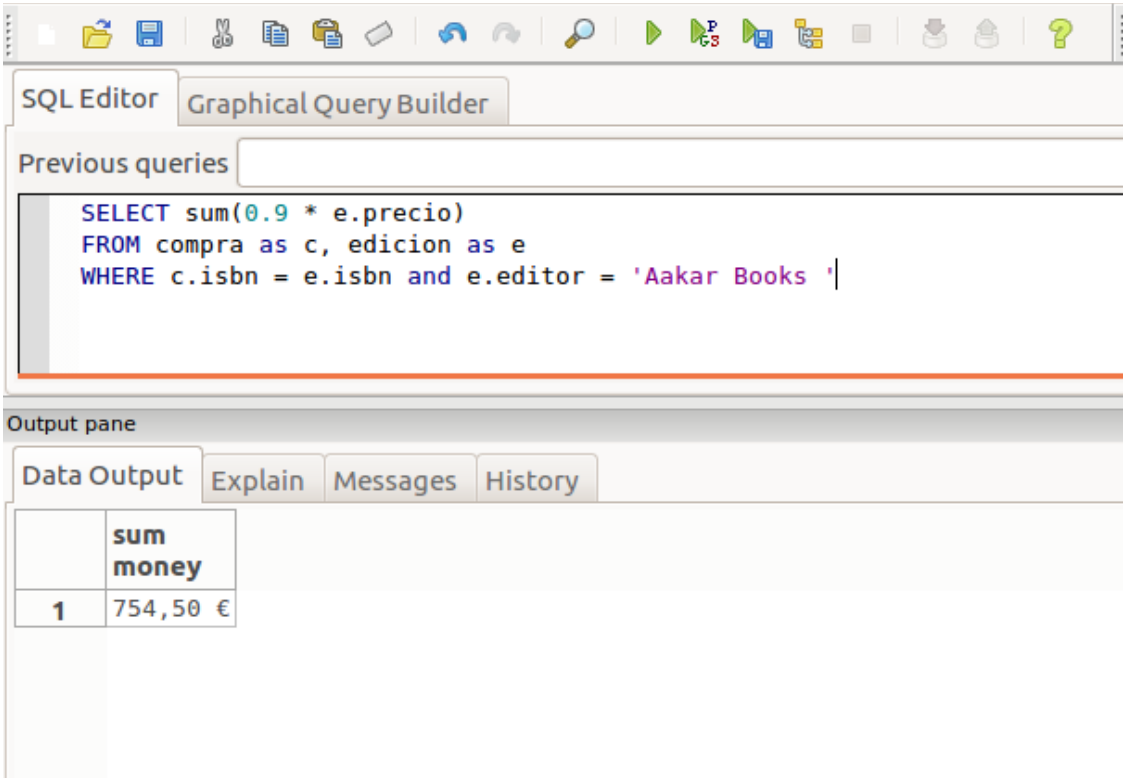

**3)¿Cuántos libros de un autor dado se han vendido en oferta?**

```
SELECT Count(*)
FROM afectaoferta as AO,
-- Libros que se han venido de un autor dado y su fecha
     (SELECT C.isbn, C.fecha
      FROM compra as C,
     -- Ediciones de libros de un autor dado
          (SELECT E.isbn
           FROM edicion as E, escribe as W, autor as A
           WHERE A.autornombre = 'Miguel de Cervantes'
          and W.dni = A.dni
          and E.titulonombre = W.titulo)as T
     WHERE T.isbn = C.isbn) as F,
oferta as O
WHERE AO.isbn = F.isbn
and O.id = AO.ofertaid
and F.fecha between O.fecha1 and O.fecha2
```



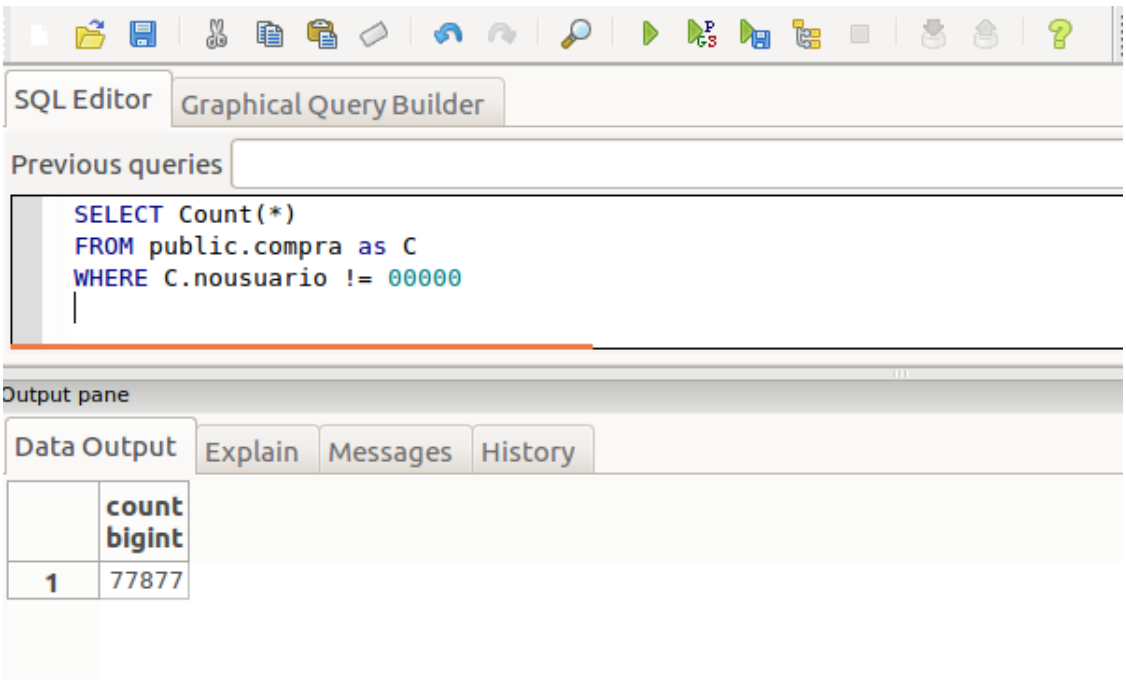

**4)¿Cuánto dinero se ha ganado vendiendo libros de un editor dado?**

```
-- Como todos los usuarios son fidelizados y todavia no
tenemos ofertas,
--podemos simplificar as la query
SELECT sum(0.9 * e.precio)
FROM compra as c, edicion as e
WHERE c.isbn = e.isbn and e.editor = 'Aakar Books '
```



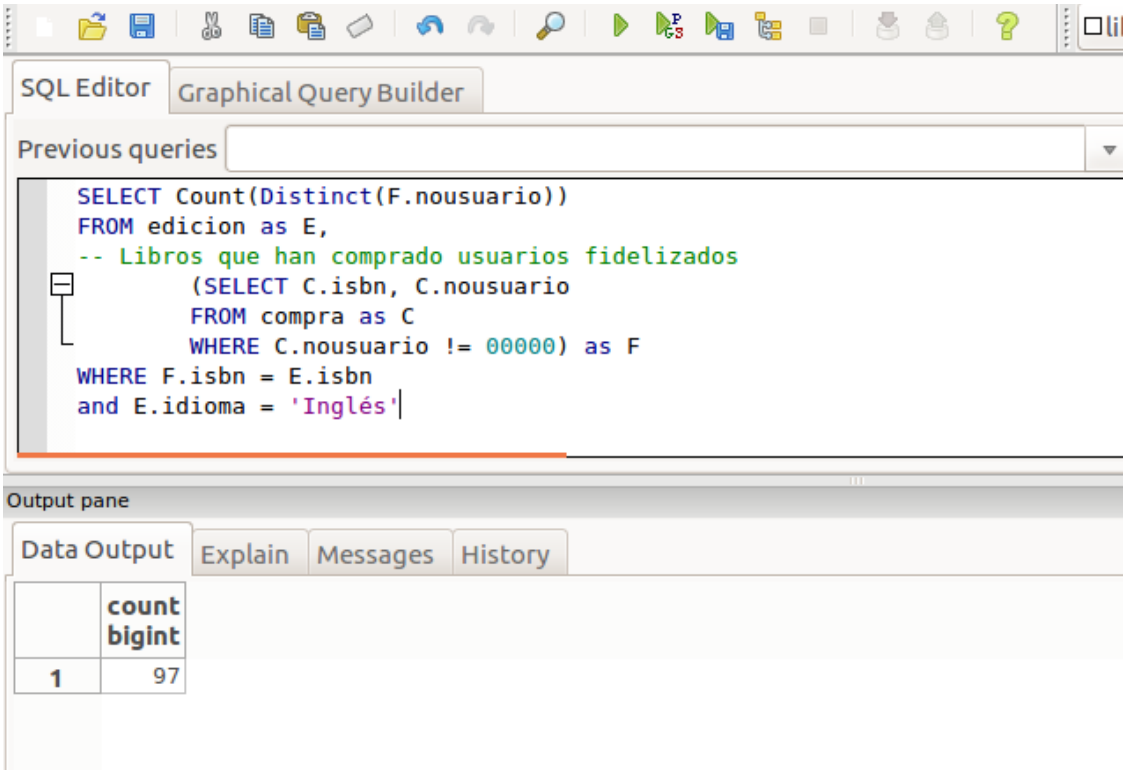

5)¿Cuántos libros han comprado los usuarios fidelizados?

```
SELECT Count(*)
FROM public.compra as C
WHERE C.nousuario != 00000
```



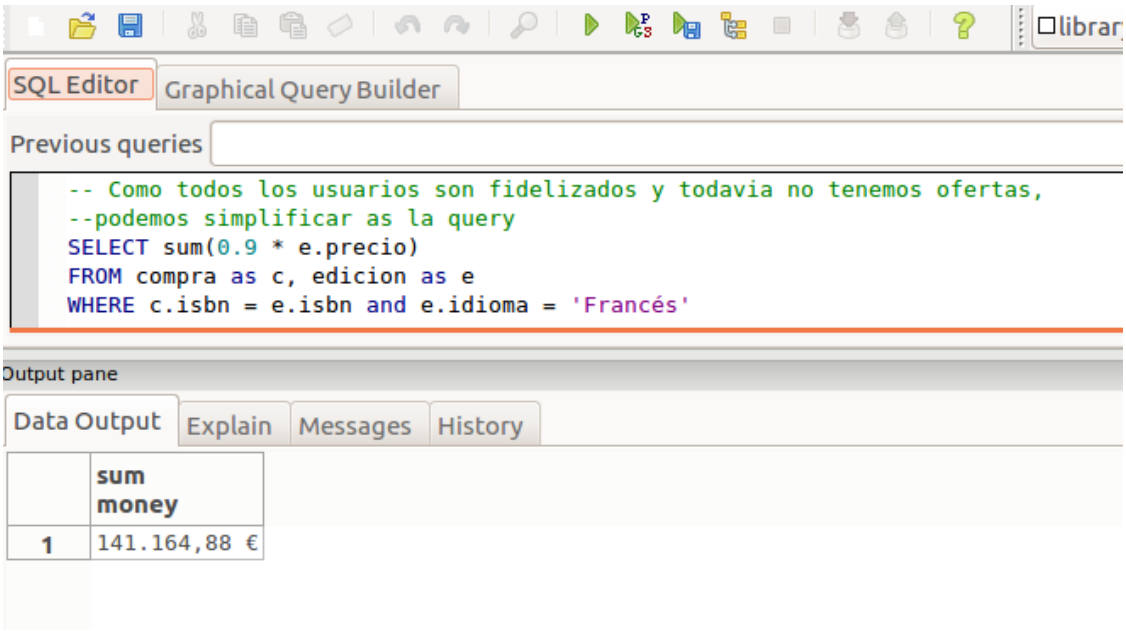
**6)¿Cuántos usuarios fidelizados han comprado libros en inglés?**

```
SELECT Count(Distinct(F.nousuario))
FROM edicion as E,
-- Libros que han comprado usuarios fidelizados
     (SELECT C.isbn, C.nousuario
     FROM compra as C
     WHERE C.nousuario != 00000) as F
WHERE F.isbn = E.isbn
and E.idioma = 'Inglés'
```



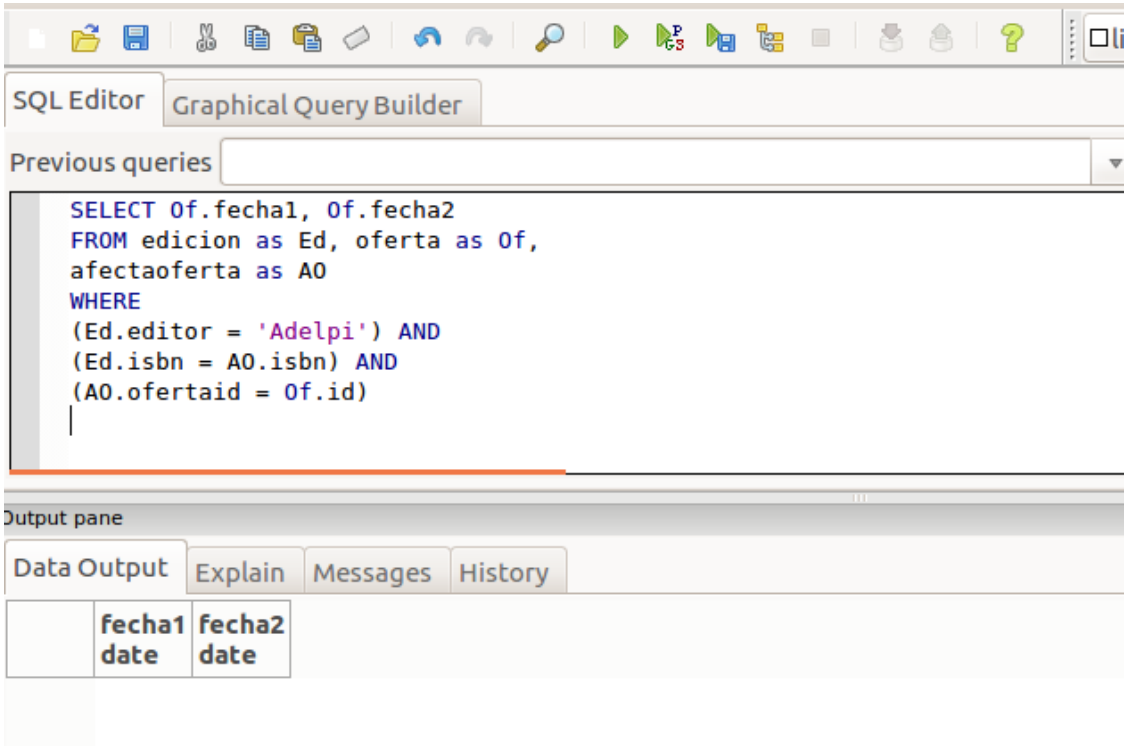

**7)¿Cuánto dinero se ha ganado vendiendo libros en Francés?**

```
-- Como todos los usuarios son fidelizados y todavia no
tenemos ofertas,
--podemos simplificar as la query
SELECT sum(0.9 * e.precio)
FROM compra as c, edicion as e
WHERE c.isbn = e.isbn and e.idioma = 'Francés'
```



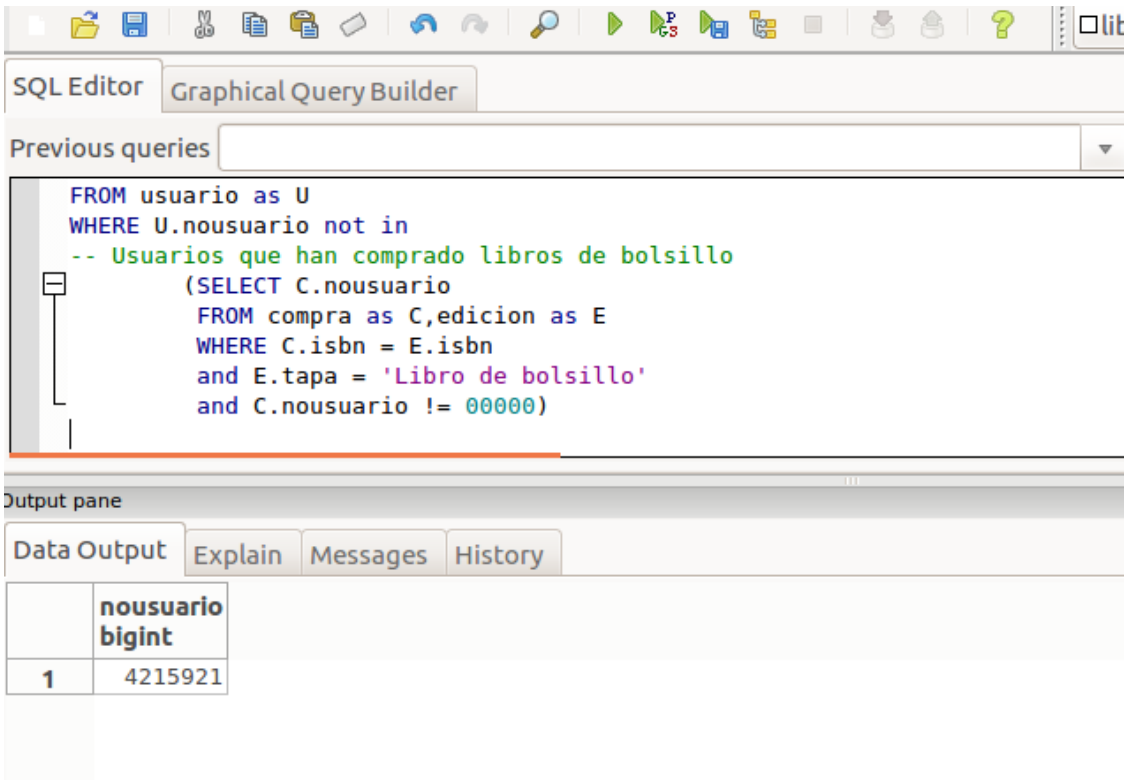

**8)¿En que días hubo ofertas de libros de la editorial Adelpi?**

```
SELECT Of.fecha1, Of.fecha2
FROM edicion as Ed, oferta as Of,
afectaoferta as AO
WHERE
(Ed.editor = 'Adelpi') AND
(Ed.isbn = AO.isbn) AND
(AO.ofertaid = Of.id)
```




**9)¿Qué usuarios fidelizados no han comprado nunca libros de bolsillo?**

```
SELECT U.nousuario
FROM usuario as U
WHERE U.nousuario not in
-- Usuarios que han comprado libros de bolsillo
     (SELECT C.nousuario
      FROM compra as C,edicion as E
      WHERE C.isbn = E.isbn
      and E.tapa = 'Libro de bolsillo'
      and C.nousuario != 00000)
```




# DOCUMENTACIÓN DE LA INTERFAZ

Para la realización de la interfaz propuesta en la práctica hemos diseñado cuatro programas distintos. El primero, “usuario.c” permite insertar y borrar usuarios, “oferta.c” crea ofertas y se las asigna a ediciones con ISBN seleccionados, “compra.c” permite realizar compras a usuarios fidelizados y calcula el precio final de la compra con los posibles descuentos aplicados. Por último “best_seller.c” nos muestra una lista con el número de libros más vendidos que desee el usuario.

Para su implementación hemos utilizado las librerias “SQL.h”, “SQLext.h” y “odbc.h”. Para hacer más sencilla la implementación hemos tenido en cuenta el control de errores solamente en las instrucciones SQLexecute (ejecuta una query) y en SQLfetch (extrae datos de una tabla).

En todos ellos en el caso de que los parámetros de entrada no sean los correctos el programa se cierra y muestra por pantalla la correcta sintáxis de los comandos.

**USUARIO**

Este programa gestiona los usuarios fidelizados de nuestra librería. Si deseamos añadir un usuario debemos ejecutar la siguiente instrucción en la consola de comandos:

```
> ./usuario + PedroG67 "Pedro González"
Usuario PedroG67 añadido con éxito.
```

Si el usuario no existe en la base de datos el programa nos avisará de que el usuario se ha añadido correctamente si no ha habido ningun problema, podemos ver que en nuestra base de datos aparece lo siguiente:

| | nousuario bigint | ccard bigint | screen_name character varying(100) | full_name character varying(100) | fecha date | borrado boolean |
|---|---|---|---|---|---|---|
| 1 | 4215922 | 0 | PedroG67 | Pedro González | 2016-11-19 | f |
| 2 | 4215921 | 9888266664171555 | jasonlk | Jason M. Lemkin | 2007-04-11 | f |
| 3 | 4188111 | 6336865874717188 | srcasm | Jesse Middleton | 2007-04-11 | f |
| 4 | 3889921 | 9371257335657443 | elbruno | Bruno Capuano | 2007-04-09 | f |

Nuestro usuario se ha añadido correctamente. Hemos tomado las siguientes decisiones de diseño al añadirlo: en primer lugar, como el número de usuario no es una variable autoincrementable hemos decidio asignarle al nuevo usuario el mayor de la base de datos mas 1, con el fin de evitar conflictos. En el caso de que no haya ningún número de usuario se le signará el 1. La tarjeta de crédito por defecto será la de número “0” ya que el programa no recibe un campo que sea tarjeta de crédito. La fecha es la actual, obtenida con la librería de c “time.h”.

En caso de ya se encuentre en la base de datos un usuario con el mismo screen name obtendremos el siguiente error:

```
> ./usuario + PedroG67 "Pedro García"
El usuario PedroG67 ya existe, escoja otro screen name.
```

Esto se debe a que en nuestra base de datos solo puede haber varios usuarios con el mismo screen name a la vez si sólo uno de ellos no está borrado.
Para borrar un usuario existente realizamos la siguiente llamada:

```
> ./usuario - PedroG67
Usuario PedroG67 borrado con éxito
```

Vemos que nuestra base de datos se actualiza:

| | nousuario bigint | ccard bigint | screen_name character varying(100) | full_name character varying(100) | fecha date | borrado boolean |
|---|---|---|---|---|---|---|
| 1 | 4215922 | 0 | PedroG67 | Pedro González | 2016-11-19 | t |
| 2 | 4215921 | 9888266664171555 | jasonlk | Jason M. Lemkin | 2007-04-11 | f |
| 3 | 4188111 | 6336865874717188 | srcasm | Jesse Middleton | 2007-04-11 | f |
| 4 | 3889921 | 9371257335657443 | elbruno | Bruno Capuano | 2007-04-09 | f |

En caso de que el usuario no exista o ya esté borrado recibimos el siguiente mensaje:

```
> ./usuario - PedroG67
El usuario introducido no se encuentra en la base de datos.
```

En caso de que el usuario que deseamos se encuentre en la base de datos borrado y el full name que introducimos en el programa coincide con el del usuario borrado simplemente actualizamos el

campo “borrado” a “false”:

```
> ./usuario + PedroG67 "Pedro González"
Usuario PedroG67 reactivado con éxito.
```

En este caso la base de datos volvería a estar en el mismo estado que en el de la primera captura.

Si, por el contrario los screen name coinciden pero el full name no, se inserta un nuevo usuario:

```
> ./usuario + PedroG67 "Pedro García"
Usuario PedroG67 añadido con éxito.
```

Nuestra base de datos quedaría así:

| | nousuario bigint | ccard bigint | screen_name character varying(100) | full_name character varying(100) | fecha date | borrado boolean |
|---|---|---|---|---|---|---|
| 1 | 4215923 | 0 | PedroG67 | Pedro García | 2016-11-19 | f |
| 2 | 4215922 | 0 | PedroG67 | Pedro González | 2016-11-19 | t |
| 3 | 4215921 | 9888266664171555 | jasonlk | Jason M. Lemkin | 2007-04-11 | f |
| 4 | 4188111 | 6336865874717188 | srcasm | Jesse Middleton | 2007-04-11 | f |

En este caso el usuario “Pedro González” ya no se podría reactivar ya que existe otro con su mismo screen name activado.

**OFERTA**

Este programa está divido en dos partes, la creación de una nueva oferta con un descuento y un intervalo de validez dado y la actualización de la tabla “afectaoferta” en la que relacionamos la oferta con unos ISBN también dados.
En este caso el número de idoferta ha sido asignado de la misma manera que el nousuario en el programa anterior. Otra decisión de diseño que cabe mencionar es el hecho de que aunque el programa reciba un descuento en formato de porcentaje nosotros lo guardaremos en la tabla como un número decimal que podamos multiplicar directamente al precio de un libro para aplicar la oferta y así facilitar a implementación de algunas querys. Por ejemplo: si el programa recibe un descuento de “60” (60%) nosotros lo guardaremos en la tabla oferta como “0.4”.

El funcionamiento del programa es el siguiente:

```
> ./oferta 20 2016-11-01 2017-01-26 1420949705 0992245338 123 45
El ISBN 3 no se encuentra en la base de datos.
El ISBN 4 no se encuentra en la base de datos.
Oferta añadida con éxito.
Se añadieron 2 ISBN a la oferta.
```

El programa indica si la oferta se ha añadido exitósamente y el número de ISBN con la que la hemos relacionado. El programa comprueba si se han los ISBN introducidos se encuentran en la base de datos, en caso contrario devolvemos un mensaje de advertencia e incluímos todos los ISBN que sí se encuentren en nuestra base de datos.

Nuestras tablas quedarían de la siguiente manera:

| | descuento double precision | id bigint | fecha1 date | fecha2 date |
|---|---|---|---|---|
| 1 | 0.8 | 1 | 2016-11-01 | 2017-01-26 |

*Tabla oferta*

| | ofertaid bigint | isbn character varying(24) |
|---|---|---|
| 1 | 1 | 1420949705 |
| 2 | 1 | 0992245338 |

*Tabla afecta-oferta*

En el caso de que no se haya añadido ningún ISBN a los parámetros de entrada nuestro programa crea la oferta pero no le asigna ningún ISBN, por lo que la tabla “afectaoferta” quedaría vacía.

**COMPRA**

Este programa permite realizar compras a los usuarios fidelizados. Imprime por pantalla los productos escogidos, el precio final de cada uno después de aplicarle los respectivos descuentos vigentes y el precio total de la compra, indicando qué producto estaba en oferta en el momento de realizar la compra. Como este programa sólo lo utilizan usuarios fidelizados todos los libros comprados tendrán un 10% de descuento independientemente del descuento que puedan tener por la influencia de una oferta.
En este caso el id de cada compra es una variable autoincrementable por lo que será asignada automáticamente.

Funcionamiento:

```
> ./compra PedroG67 1420949705 0992245338 145905928X 1234567890

Ha comprado: La Galatea
Precio final de esta edición: 11,68 €
Este producto se encuentra en oferta.

Ha comprado: Oceania: Neocolonialism, Nukes and Bones
Precio final de esta edición: 16,33 €
Este producto se encuentra en oferta.

Ha comprado: Castelvines y Monteses, tragi-comedia, tr. by F.W. Cosens
Precio final de esta edición: 11,06 €

El ISBN 4 no se encuentra en la base de datos.

Precio final de la compra: 39.07€
```

La tabla de compras realizadas por el usuario “PedroG67” quedaría:

| | isbn character varying(24) | nousuario bigint | fecha date | idcompra integer |
|---|---|---|---|---|
| 1 | 1420949705 | 4215923 | 2016-11-19 | 77969 |
| 2 | 0992245338 | 4215923 | 2016-11-19 | 77970 |
| 3 | 145905928X | 4215923 | 2016-11-19 | 77971 |

En caso de que el usuario no exista aparece el siguiente aviso:

```
> ./compra FalseUser 1420949705 0992245338 145905928X 1234567890
El usuario FalseUser no existe.
```

**BEST SELLER**

Este sencillo programa muestra la lista de los libros más vendidos. La longitud de la lista varía según el parámetro que imponga el usuario.

```
> ./best_seller 5
1º: Don Quixote
2º: DK Eyewitness Travel Guide: Seville & Andalusia
3º: Who Rules the World?
4º: DK Eyewitness Travel Guide: Spain
5º: Gaza in Crisis: Reflections on Israel's War Against the Palestinians
```

En caso de que se pida un número de best-sellers (por ejemplo 500) mayor al número de ejemplares distintos vendidos se obtiene este aviso:

```
344º: Inside Lebanon
345º: Selected Commercial Statutes (Selected Statutes)
346º: The Uses of Haiti
347º: L' Toile de S Ville;
348º: Don Quixote Of The Mancha (Everyman's Library Children's Classics)
No hay suficientes ventas como para tener 500 bestsellers
```