Universidad Autónoma de Madrid

Departamento de Informática

Estructura de Datos
Práctica - 1

Roberto Marabini Ruiz

Índice

1. Objetivos

A lo largo de esta práctica diseñaremos, desarrollaremos y haremos consultas en una base de datos de modelos de coches clásicos. Las bases de datos se crearán en el gestor de bases de datos PostgreSQL y las consultas se llevaran a cabo usando el lenguaje SQL.

2. Familiarizarse con la Base de Datos

Como base de datos se usará la base de datos de dominio público creada para el MySQL-tutorial (`https://www.mysqltutorial.org/mysql-sample-database.aspx`)

En *Moodle* podéis encontrar los ficheros `Makefile` y `classicmodels.sql`. Bajadlos y guardadlos en la misma carpeta. El comando `make` además de utilizarse para compilar código puede ser usado para automatizar otras tareas. En nuestro caso lo usaremos para crear y borrar bases de datos así como para ejecutar las consultas a las mismas. En nuestro `Makefile` hemos definido los comandos que se especifican en la Tabla 1.

comando	**función**
createdb	crea una base de datos llamada `classicmodels`
dropdb	borra la base de datos
dump	crea una copia de respaldo de la base de datos
restore	carga la copia de respaldo
shell	invoca el cliente de linea de comandos *psql*
all	ejecuta dropdb, createdb y restore

Tabla 1: listado de las tareas que se pueden ejecutar usando make.

Cread y poblad la base de datos usando `make all`, examinad la base de datos creada usando un cliente bien sea de línea de comandos (`make shell`) o gráfico `pgAdmin4`) y proporcionad, en la memoria de la práctica, la información siguiente:

1. Clave primaria de cada tabla.

2. Claves extranjeras que aparezcan en cada tabla.

3. Diagrama del modelo relacional de la base de datos implementada.

   ...

Para responder a las dos primeras preguntas escribid el esquema de la base de datos como en el ejemplo siguiente:

nombreTabla(**attrbA**, attrbB, ...)
otraTabla(**attrb1**, attrb2 → nombreTabla.attrA, attr3, ...)
...

donde *nombreTabla* y *otraTabla* son los nombres de las tablas, la clave primaria está en negrita (en este caso es el atributo llamado *attrbA* en la tabla *nombreTabla* y el atributo llamado *attrb1* en la tabla *otraTabla*); las claves extranjeras (en este caso *attrb2*) apuntan al atributo que referencia (en este caso el atributo *attrA* de la tabla *nombreTabla*).

Vuestro diagrama del modelo relacional debe seguir un modelo parecido al de la Figura 1 donde tanto las claves primarias como las extranjeras están marcadas explícitamente (PK → clave primaria, FK → clave extranjera). Así mismo, las tablas que contienen atributos relacionados están unidas mediante flechas.

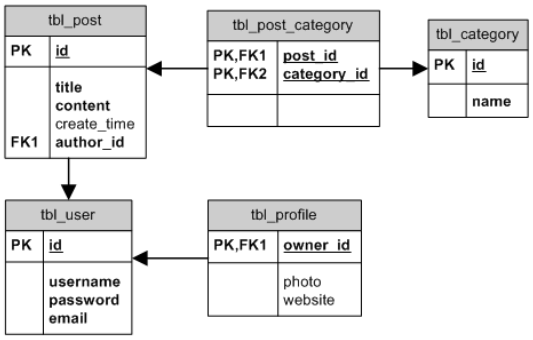

Figura 1: Ejemplo de diagrama relacional

3. Consultas

A continuación, se enumeran un conjunto de consultas a la base de datos que deberéis implementar usando SQL (leed las notas relativas a la implementación de las consultas antes de implementar esta parte).

1. Muestra la cantidad total de dinero abonado por los clientes que han adquirido el "1940 Ford Pickup Truck" (el dinero puede haber sido abonado para comprar otros modelos). Ordena el resultado por la cantidad de dinero abonada de mayor a menor cantidad. Cada línea debe mostrar: "customernumber", "customername" y la cantidad total de dinero pagada.

2. Tiempo medio transcurrido entre que se realiza un pedido (orderdate) y se envía el pedido (shippeddate) agrupado por tipo de producto ("productline"). Cada línea debe mostrar el "productline" y el tiempo medio correspondiente.

3. Empleados que reportan a otros empleados que reportan al director. El director es aquella persona que no reporta a nadie. El listado debe mostrar el "employeenumber" y el "lastname".

4. Oficina que ha vendido el mayor número de objetos. Nota: en un pedido ("order") se puede vender más de una unidad de cada producto, cada unidad se considerará un objeto. La salida debe mostrar el "officecode" y el número de productos vendidos.

5. Países que tienen al menos una oficina que no ha vendido nada durante el año 2003. La salida debe mostrar dos columnas conteniendo el nombre del país y el número de oficinas que no han realizado ninguna venta. Ordena las salidas por el número de oficinas de forma que la primera línea muestre el país con más oficinas que no han realizado ninguna venta.

6. Definimos el carro de la compra como el conjunto de todos los productos comprados usando la misma "order". Se desea un listado de todas las parejas de productos que aparezcan en más de un carro de la compra. La salida debe mostrar tres líneas conteniendo el identificador de ambos productos y el numero

de carros de la compra en el cual aparecen. Nota, cada pareja de productos debe aparecer una única vez, en particular la pareja de productos con identificadores (id1, id2) no es distinta de la pareja (Id2, Id1) que tiene los mismos identificadores pero en otro orden.

3.1. Notas relativas a la implementación de las consultas

Makefile

Se desea que las consultas se puedan ejecutar usando `make`. En particular `make query1` debería ejecutar la primera consulta, `make query2` debería ejecutar la segunda consulta y así sucesivamente.

Para conseguirlo, deberéis crear un fichero por cada consulta llamado `queryX.sql` (donde X es el número de la consulta) que contenga el código SQL y añadir en el fichero `makefile` las siguientes líneas (por cada consulta):

```
query1:
@echo consulta-1: "Cantidad␣total␣de␣dinero␣abonado..."
@cat consulta1.sql | $(PSQL) | tee consulta1.log
```

Recordad que los espacios a la izquierda de cada orden en un fichero `makefile` son tabuladores.

Formateo de Consultas

Un buen código debe ser legible tanto para el programador que lo crea como para cualquier otra persona que revise el código. Por favor escribid las consultas de forma consistente usando sangrías y saltos de línea para mejorar su legibilidad. Antes de entregar el código pasarlo por un "formateador". Se sugiere usar `http://www.dpriver.com/pp/sqlformat.htm`. Si no os gusta buscad otro que os satisfaga y tenga una funcionalidad similar, e incluid el URL en la primera línea del `makefile` como un comentario.

3.1.1. ¿Cuándo es una consulta correcta?

Una consulta es correcta si satisface los siguientes requerimientos:

1. Se puede ejecutar en los ordenadores de los laboratorios usando los comandos definidos en el fichero `makefile` y descritos en esta memoria.
2. Proporciona el resultado correcto para **cualquier** instancia de la base de datos.
3. El código es eficiente y legible. Se acepta como código legible el obtenido usando la utilidad accesible en `http://www.dpriver.com/pp/sqlformat.htm` u otra utilidad semejante y accesible on-line en la dirección dada en la primera línea del fichero `makefile`. La consulta será considerada eficiente si el tiempo necesario para ejecutarla no es superior a 20 veces el tiempo requerido por la implementación de la consulta desarrollada por los profesores de esta asignatura.

Si cualquiera de estos puntos no se satisface, la consulta se calificará con cero puntos.

4. Rediseño de la Base de Datos

El diseño de la base de datos es en general bastante pobre y muestra múltiples deficiencias. Por ejemplo: si un empleado se mueve de una oficina a otra se pierde la información de la/s oficina/s donde había trabajado en el pasado, un cliente solo puede relacionarse con un único empleado o los pagos no están asociados a una compra en concreto.

Diseña una base de datos que evite los inconvenientes citados. Incluye en la memoria: (1) el nuevo diagrama relacional y comenta cómo tus cambios solucionan los problemas planteados y (2) añade un nuevo comando en el fichero `makefile` que se ejecute con `makefile nuevabase` y que borre todas las tablas y datos de la base de datos y cree las tablas de tu nuevo diseño (los comandos SQL necesarios para crear las tablas se almacenarán en un fichero llamado nuevabase.sql).

5. Resumen del material a entregar

Subid a *Moodle* un fichero único en formato zip que contenga:

1. Fichero `makefile` y todos los ficheros necesarios para crear la base de datos, borrarla, poblarla y ejecutar cada consulta. IMPORTANTE: **NO incluyáis el fichero** `classicmodels.sql`.

2. Memoria en formato pdf que incluya: el esquema de la base de datos, el diagrama relacional de la base de datos y responda a las preguntas realizadas en esta práctica. Adicionalmente, para cada consulta en SQL incluid un listado con el código y una breve descripción de la implementación. No os olvidéis de incluir el material solicitado en el apartado 4

6. Criterios de corrección

Para aprobar es necesario:

- Que tanto el esquema como el diagrama relacional de la base de datos sean correctos. (Se admitirán un máximo de tres errores).
- Tener 3 consultas totalmente correctas.

Recordad que los criterios para decidir si una consulta es correcta están definidos en la subseccion 3.1.1

Para obtener una nota en el rango 5-7 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- Tener 5 consultas correctas

Para obtener una nota en el rango 7-8 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.

- Qué tanto el esquema como el diagrama relacional de la base de datos sean totalmente correctos.
- Igualmente, se espera que la memoria esté correctamente redactada y demuestre vuestra comprensión de la práctica.

Para obtener una nota en el rango 8-9 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- Que todas las consultas sean correctas.

Para obtener una nota en el rango 9-10 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- Haber rediseñado la base datos tal y como se solicita en la práctica.

NOTA: Cada día (o fracción) de retraso en la entrega de la práctica se penalizará con un punto.