Escuela Politécnica Superior, UAM

Estructura de Datos 2016-2017

Práctica 2. Bases de datos y ODBC.

Fecha límite de entrega de la práctica

Grupo	Fecha de entrega
1201 (miércoles 18-20)	**16 de Noviembre**
1202 (martes 18-20)	**15 de Noviembre**
1211 (lunes 9-11)	**14 de Noviembre**
1212 (viernes 11-13)	**18 de Noviembre**
1261 (martes 18-20)	**15 de Noviembre**
1271 (miércoles 18-20)	**16 de Noviembre**
1272 (viernes 16-18)	**18 de Noviembre**
1273 (viernes 18-20)	**18 de Noviembre**

Las prácticas se entregarán mediante el programa Moodle (http://www.uam-virtual.es). El retraso en la entrega se penalizará con 1 punto por cada día natural después de la fecha límite.

CONSEJOS ÚTILES:

1. Las memorias, cuando se piden, son importantes: no sólo sirven para que vuestro profesor tenga una idea de lo que habéis hecho y cómo lo habéis organizado; también proporciona indicaciones importantes sobre el razonamiento que habéis seguido para hacer las cosas de una determinada manera. Esto nos ayudará a evaluar mejor vuestro trabajo. Además, escribir una buena memoria es importante para los que, como vosotros, trabajarán en campo técnico: hacer que los colegas, los jefes y los clientes entiendan vuestro trabajo será una de vuestras actividades más importantes.
2. Puede parecer banal, pero… ¡poned el nombre de los componentes del grupo en TODOS los ficheros! No os podéis ni siquiera imaginar cuantas maneras hay de perder el rastro de quien ha hecho algo.

Objetivos

Hoy en día, es muy raro que el usuario final interactúe directamente con una base de datos. Lo normal es que se acceda a la base de datos a través de un programa, que

constituye nuestra interfaz, y que traduce nuestros comandos y nuestras acciones en consultas para la base de datos. Esto es lo que hacemos, por ejemplo, cada vez que accedemos a una página web en un sitio de cierta envergadura que contiene muchos datos. En esta práctica experimentaremos con la conexión entre programas en C y bases de datos. En particular, usaremos la librería ODBC para crear una serie de programas que actuarán sobre la base de datos de la librería que hemos diseñado en la primera práctica.

Parte I: Refinar el diseño de la base de datos

En la primera práctica hemos creado un diseño de una base de datos para gestionar una librería. En esta práctica poblaremos la base de datos con datos sacados de unos ficheros de texto (están en Moodle). Antes de cargar los datos, revisaremos el diseño para hacerlo más "concreto".

Revisión del diseño

Cuando se pasa de un diseño conceptual a su implementación en una base de datos, es necesario a veces renunciar a la abstracta y diáfana pureza del diseño y ensuciarse las manos con el barro de la practicidad, para conseguir una base de datos más manejable y eficiente.

Ya hemos visto en clase (supongo...) algunas de estas soluciones prácticas. Por ejemplo, hemos visto como algunas relaciones del modelo ER (relaciones 1:n o 1:1) pueden no generar relaciones en el modelo relacional, y se pueden representar simplemente poniendo la clave de una entidad en la representación de la otra. Ahora verificaremos también otras cosas.

Por ejemplo, en el diseño ER algunas entidades pueden tener una clave formada por varios atributos de texto. En este caso puede ser conveniente introducir un campo adicional que se utilice como clave. Esto se puede hacer creando una columna con tipo de datos SEQUENTIAL, es decir, un tipo de datos cuyo valor se incrementa automáticamente cada vez que se añade una tupla, de manera tal que cada tupla tendrá en este campo un valor diferente.

Una relación (del diagrama ER) con multiplicidad 0..1 no habrá sido eliminada al crear el diseño relacional correspondiente, dado que hay entidades que podrían no estar en esta relación (p.ej., una venta se hace con una y sólo una tarjeta de crédito, pero hay

ventas que se hacen en efectivo y por tanto no se relacionan con ninguna tarjeta; por tanto, la relación entre venta y tarjeta se habrá representado como una tabla independiente). Estas relaciones se pueden eliminar usando el valor NULL para indicar que una entidad no participa en ella. El uso de valores NULL no está exento de riesgos, y hay que tener cuidado a la hora de usarlos. En este contexto puede estar justificado su utilización, pero debe hacerse con buen juicio y sentido común.

Poblar las tablas[1]

Tras refinar el diseño, hay que insertar los datos en las tablas. Aquí tendremos que resolver un problema muy común a la hora de poblar bases de datos con datos existentes: el esquema de los datos tal y como aparece en el fichero no se corresponde con las tablas que hemos definido en el diseño relacional. Será necesario "convencer" los datos para que se metan en nuestra tabla. Esta parte será claramente diferente por cada pareja, dado que cada pareja tendrá su diseño individual con tablas ligeramente distintas. Las bases son, por otro lado, iguales para todos, y las podemos dividir en cinco pasos:

1. se crea una colección de tablas temporales, una tabla por cada fichero de datos. Cada tabla tendrá un número de columnas igual al número de campos del fichero y de los tipos adecuados para recoger los datos del fichero;
2. mediante la instrucción COPY se insertan los datos de los ficheros en las tablas temporales;
3. se crean las tablas del diseño relacional elegido;
4. se transforman los datos: por cada tabla del diseño relacional se crea una consulta que, partiendo de los datos de las tablas temporales, crea la tabla del diseño relacional; los resultados de la consulta se insertan físicamente en la tabla usando la consulta dentro de un INSERT INTO;
5. con un DROP TABLE se borran las tablas temporales.

Ejecutar las consultas

Ahora que las tablas están bien bien -1488292591 *llenitas* de datos, podemos ejecutar en ellas las 9 consultas que hemos diseñado en la práctica 1. Si todo va bien, es probable que una mitad de las consultas funcionen bien, mientras una mitad necesitarán algún tipo de ajuste. No hay que preocuparse: cuando se hace una consulta basándose simplemente en que esquema de la base de datos, así como hicimos en la práctica 1, uno se concentra principalmente en los aspectos conceptuales de la consulta (¿tengo toda la información disponible?), cuando uno la ejecuta los detalles

[1] No… no estamos hablando de enviar gente a vivir a Damiel...

acaban teniendo su importancia. -1488292592 Como se dice; *the devil is in the details*...

¡Viva el C!

Como se decía en los párrafos de introducción, hoy en día nadie (casi nadie... si trabajáis en Oracle las cosas ya cambian) interactúa con una base de datos escribiendo directamente consultas SQL. En este apartado usaremos ODBC para crear unos programas en C que ejecuten ciertas operaciones en la base de datos. ODBC es una librería para enviar consultas a una base de datos y recibir resultados. En clase veremos los conceptos fundamentales de ODBC y como usarla para interactuar con la base de datos.

Se crearán los programas siguientes:

usuario

Este programa tiene dos funciones: añadir y eliminar un usuario fidelizado. Para añadir un usuario se ejecutará

usuario + <screen_name> "<full name>"

el programa imprimirá un mensaje de confirmación o un mensaje de error su el screen_name ya está asignado; para eliminar un usuario se ejecutará

usuario - <screen_name>

el programa imprimirá un mensaje de confirmación o un mensaje de error su el screen_name no existe en la base de datos.

Atención: el usuario no se borrará físicamente de la base de datos. Si se hiciera esto, todas las compras que el usuario ha efectuado en el pasado quedarían sin usuario de referencia y (según el diseño que habéis elegido) sin tarjeta de crédito asociada. Lo que hay que hacer es introducir una columna con un valor BOOLEAN que nos diga que el usuario ha sido borrado.

oferta

Este programa permite la introducción de una oferta. Para añadir una oferta se ejecutará:

oferta <descuento> <de> <a> <isbn> <isbn> .... <isbn>

donde descuento es un entero entre 0 (ningún descuento) y 100 (regalamos el libro). La lista de isbn tiene longitud arbitraria, y representa todos los libros a que se aplica el descuento.

Atención: en el diseño cada oferta tendrá un identificador único; por otro lado, en la lista de parámetro no se da ningún identificador: el mismo programa tendrá que generar un identificador para la oferta, asegurándose que ya no exista en la tabla (para esto hay un simple truco que veremos en clase).

compra

Con este programa un usuario registrado podrá comprar libros. Para esto se ejecutará:

compra <screen_name> <isbn> <isbn> .... <isbn>

La lista de isbn tiene longitud arbitraria, y representa todos los libros que el usuario quiere comprar. El programa imprimirá en pantalla el coste de cada libro (uno por línea) teniendo en cuenta el descuento, y el coste total de la compra.

best_seller

Este programa enseña los libros más vendidos. Se ejecuta como

best_seller <n>

El programa enseña el título y el número de copias vendidas de todas las ediciones de los <n> libros más vendidos

Qué hay que entregar

1. Las tablas definitivas de vuestro diseño relacional. Si no habéis cambiado nada respeto al diseño de la práctica 1, entregad otra vez las tablas de la práctica 1 sin más; si, como es probable, habréis tenido que introducir algún cambio, entregad las nuevas tablas y una breve descripción de los cambios que habéis introducido;
2. todos los comandos que habéis usado para poblar las tablas (en un fichero txt es suficiente) y los ficheros de texto que habéis usado (los ficheros de texto con los datos son necesarios porque a veces se hacen pequeños cambios en ellos, de forma que no sería posible reproducir vuetros resultados empleando los comandos que figuren en la memoria);
3. las consultas de las práctica 1 en su forma final y los resultados de su ejecución en la base de datos poblada; si alguna de esas consultas diera un resultado muy

largo (no creo, pero no las he probado, así que no lo sé) entregad sólo las primeras 10 tuplas del resultado;

4. Todos los códigos fuente de los programa en C, así como el fichero Makefile necesario para su compilación. Se recuerda que es necesario que el código esté debidamente documentado y depurado mediante valgrind.

Enlaces de utilidad sobre ODBC en C:

1. http://www.unixodbc.org/
2. http://www.easysoft.com/developer/languages/c/index.html
3. http://www.easysoft.com/developer/languages/c/examples/index.html