Escuela Politécnica Superior, UAM

Estructura de Datos 2016-2017

Práctica 1. Diseño de una base de datos relacional.

Fecha límite de entrega de la práctica

Grupo	Fecha de entrega
1201 (miércoles 18-20)	**12 de Octubre**
1202 (martes 18-20)	**11 de Octubre**
1211 (lunes 9-11)	**10 de Octubre**
1212 (viernes 11-13)	**14 de Octubre**
1271 (miércoles 18-20)	**12 de Octubre**
1272 (viernes 16-18)	**14 de Octubre**
1273 (viernes 18-20)	**14 de Octubre**

Las prácticas se entregarán mediante el programa Moodle: en la sección de práctica habrá varios enlaces (uno por grupo) para la entrega. Por favor, entregad la práctica en el enlace de vuestro grupo. El retraso en la entrega se penalizará con 1 punto por cada día natural después de la fecha límite.

CONSEJOS ÚTILES (SÍ: SON ÚTILES DE VERDAD):

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.
3. Entregad, por cuanto sea posible, todos los documentos **en formato pdf**. Es muy mala práctica entregar ficheros "editables" o ficheros en el formato del programa de videoescritura, incluso si se trata de word. Los ficheros de texto los podéis entregar así como están.

Objetivos

En esta práctica llevaremos a cabo un simple diseño relacional en tres fases: primero, crearemos un diagrama ER que capture las características del diseño; luego transformaremos este diagrama en un diseño relacional razonablemente optimizado (es decir, eliminando las relaciones claramente redundantes) y finalmente usaremos este diseño para expresar algunas consultas sencillas.

El problema es crear un modelo de una librería para gestionar el catálogo de los libros, de los autores, las ventas, las ofertas, etc. Algunos requisitos son los siguientes:

1. La librería vende, claramente, libros, pero un libro es una cosa bastante complicada. Hay que distinguir entre un título (el libro como objeto abstracto: por ejemplo *De Ratones y Hombres*), que tiene uno o más autores (Steinbeck, en este caso) y sus varias ediciones: hay ediciones en distintos idiomas (*Of mice and men*, *Uomini e topi*, *Des souris et des hommes*, etc.), ediciones de capa dura o blanda, etc. Cada edición del mismo libro puede tener un editor diferente y un precio diferente.
2. La librería tiene ofertas. Una oferta es un descuento (fijo: depende de la oferta pero es igual para todos) que se aplica a un grupo de libros si se compran entre dos fechas. La librería quiere tener un registro de todas las ofertas que hace.
3. La librería tiene cierto número de usuarios “fidelizados”, registrados cada uno con su identificador y una (sólo una) tarjeta de crédito. Cada usuario recibe un descuento del 10% en cada libro que compra, incluso en libros en oferta; la librería quiere tener un registro de lo que ha gastado cada usuario.
4. Otras personas (los que no son usuarios identificados) pueden naturalmente comprar libros: la librería quiere tener un registro de todas las ventas, registrando por cada una que edición de que libro se ha comprado, cuanto ha costado, si se ha pagado en efectivo o con tarjeta, la fecha de la compra, etc. En la compra hay que indicar si se ha aplicado alguna oferta. Esta misma información debe ser registrada también para los usuarios “fidelizados”

Para validar el modelo, averiguar que se proporciona suficiente información y suficientes relaciones para contestar las preguntas siguientes:

1. Dado un título, ¿Cuántas ediciones tiene? ¿En cuántos 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?
5. ¿Cuántos libros han comprado los usuarios fidelizados?
6. ¿Cuántos usuarios fidelizados han comprado libros en inglés?
7. ¿Cuánto dinero se ha ganado vendiendo libros en Francés?
8. ¿En que días hubo ofertas de libros de la editorial Adelpi?

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

Una vez validado el modelo habrá que convertirlo a relacional. Para esto, primero se creará un modelo siguiendo mecánicamente el procedimiento canónico es decir, una relación por cada entidad y cada relación del modelo. Llamaremos este el modelo relacional *tontorrón*. A continuación habrá que mejorar el problema eliminando esas relaciones que, en base a la multiplicidad de las relaciones del modelo ER, son redundantes. Este será nuestro modelo *optimizado*.

Que hay que entregar:

a. Un diagrama ER con el diseño del sistema de gestión de la librería. Podéis dibujar el diagrama ER usando Powerpoint o algún programa disponible en red. Para evitar problemas de formato, por favor **entregad este diagrama en formato pdf**
b. Un breve informe en que se analiza el diseño. En particular, el informe debería analizar posibles redundancias en el diseño y puntos en que se podría dar una inconsistencia en los datos
c. En el informe, se haga un breve análisis de las consultas propuestas; por cada una, se describan las relaciones y las entidades que hay que cruzar para contestar la pregunta
d. La lista de las cabeceras de las relaciones de los dos modelos relacionales, el tontorrón y el optimizado.
e. El texto de las consultas SQL correspondientes a las consultas 1-8 (la pregunta 9 es más complicada, y es opcional); estas consultas habrá que escribirlas sólo para el modelo optimizado.

Nota importante: Para crear el diagrama ER, existen varias posibilidades. La más inmediata es utilizar Powerpoint: los instrumentos gráficos que pone a disposición son ampliamente suficientes para dibujar un diagrama como este. En red hay también programas específicos para el diseño de diagramas ER. Muchos de ellos son de pago, pero los hay gratis. También es aceptable dibujar el diagrama en un papel y escanearlo (no vale fotografiarlo con el móvil: la calidad es generalmente insuficiente), siempre y cuando el diagrama sea bien hecho, preciso y legible: ¡una foresta de líneas dibujada sobre un papel con restos de pizza no vale!

Nota aun más importante: Todos los ficheros mencionados en esta práctica se recogerán en un *único* fichero en formato .tgz o .zip con nombre EDAT1617_p1_XXX_AAA_BBB.zip donde XXX es el número del grupo de prácticas, AAA y BBB son los apellidos de los miembros de la pareja. Este fichero se entregará mediante moodle, usando el enlace que se publicará en la página de la asignatura. Sólo tendrá que entregar el fichero uno de los miembros de la pareja. Por favor, evitad las

entregas dobles: no os preocupéis, ¡os vamos a dar la nota a los dos!, sobre todo si seguís el consejo de la:

Nota tremendamente importante: el fichero .tar o .zip ya nos permite saber quién sois, pero siempre es buena norma *poner vuestro nombre en todos los ficheros de textos que entreguéis*. Todos los formatos permiten un campo de comentario: al principio del fichero poned el nombre. Este consejo no se limita a EDAT: vale para todas las asignaturas, y os puede evitar un montón de problemas.

PostgreSQL en los laboratorios

Los ordenadores de los laboratorios de la Escuela tienen instalado PostgreSQL en Linux. Asimismo, disponen de varias aplicaciones front-end, tales comoTora en y pgAdmin.

En los laboratorios, el acceso a PostgreSQL se realiza con los siguientes usuarios y contraseñas:

- *Linux*: usuario **alumnodb**, sin contraseña

PostgreSQL

A continuación presentamos una breve introducción de cómo PostgreSQL, basado en un modelo cliente-servidor, ejecuta las órdenes impartidas por el usuario desde una aplicación front-end. Una sesión de PostgreSQL consiste de los siguientes procesos:

1. Demonio *postmaster*: es el proceso principal y se ejecuta en el servidor.

2. Procesos *postgres (back-end)*: se trata de los procesos que manejan los ficheros de la base de datos, aceptan conexiones a la base de datos desde las aplicaciones cliente y realizan acciones sobre la base de datos, sin que el cliente se percate de ello.

3. Entorno sobre el que trabaja el usuario (front-end), por ejemplo pgAdmin. El cliente interactúa con el usuario, envía sus comandos a los procesos *postgres* y recibe los resultados.

Una aplicación front-end envía un paquete de inicio al postmaster, donde incluye el nombre del usuario y la base de datos a la que el usuario quiere conectarse. El postmaster utiliza esta información junto con la contenida en el fichero pg_hba.conf para determinar qué información adicional de autentificación necesita del front-end y responde al front-end en concordancia. A continuación, el front-end envía la información de autentificación requerida.

Una vez que el postmaster valida esta información responde al front-end que está autentificado y entrega una conexión a un proceso back-end. El back-end entonces envía un mensaje indicando arranque correcto (caso normal) o fallo (por ejemplo, un nombre de base de datos inválido o contraseña incorrecta).

Las comunicaciones siguientes son paquetes de consulta y resultados intercambiados entre el front-end y el back-end. El postmaster no interviene ya en la comunicación ordinaria de consultas/resultados. Sin embargo, el postmaster puede involucrarse ya que vigila las solicitudes y está al tanto de los movimientos. Un caso típico de actuación en el postmaster podría ser cuando el front-end desea cancelar una consulta que se esté efectuando en su back-end.

Como típica aplicación cliente-servidor, el cliente y el servidor pueden estar en diferentes máquinas. En este caso la comunicación entre ambos es a través del protocolo TCP/IP.

El servidor postgreSQL puede manejar varias conexiones concurrentes con los mismos datos, tanto realizando consultas como actualizando las tablas de la base de datos. El gestor es el responsable de coordinar todas estas peticiones asegurando la integridad de los datos.

PostgreSQL en los laboratorios

Desde el terminal/consola de comandos (*usuario* = usuario PostgreSQL, *bd* = nombre base de datos, *fichero* = nombre fichero):

- Creación de una base de datos **createdb -U** *usuario bd*
- Acceso a una base de datos **psql -U** *usuario bd*
- Eliminación de una base de datos **dropdb -U** *usuario bd*
- Hacer un volcado de una base de datos **pg_dump --inserts -U** *usuario bd* > *fichero*

Desde la línea de comandos de PostgreSQL:

- Sentencias SQL Se ejecutan finalizándolas con ;
- Salir \q
- Ayuda \h
- Mostrar comandos internos de psql \?
- Entrada de órdenes desde un fichero \i nombre_fichero

Enlaces de interés

- **Descarga de PostgreSQL**
  - Windows, http://www.postgresql.org/download/windows
  - Linux, http://www.postgresql.org/download/linux
  - Mac OS, http://www.postgresql.org/download/macosx
- **Descarga de aplicaciones front-end para PostgreSQL**
  - Navicat Lite, http://www.navicat.com/en/download/download.html
  - pgAdmin, http://www.pgadmin.org
  - Tora, http://torasql.com/Download