Sistemas de Información

Práctica 6 Curso 2015/16 3º curso

Otros paradigmas de almacenamiento

Introducción

Durante las prácticas anteriores nos hemos centrado en la utilización de bases de datos que siguen el modelo lógico relacional. Este tipo de bases de datos configuran el núcleo de la asignatura, porque son, con mucha diferencia, los sistemas de almacenamiento de información más utilizados hoy en día.

Además, por suerte, el estándar SQL se ha impuesto como lenguaje de interacción con estas bases de datos, de tal forma que no es necesario aprender otros lenguajes para poder interactuar con cualquier SGBD relacional de los existentes hoy en día.

Sin embargo, tal como ya sabemos, el relacional no es el único paradigma de almacenamiento y gestión de información existente. Por tanto, vamos a dedicar esta última práctica a experimentar un poco con algunos sistemas que sigan otros paradigmas.

De entre las posibles opciones entre las que podíamos elegir, hemos seleccionado el paradigma orientado a objetos y el modelo semiestructurado de XML.

Para muchos autores, el modelo orientado a objetos (incluyendo, como paso previo el modelo objeto-relacional) representa el siguiente paso en la evolución de las bases de datos actuales, por lo que consideramos que no está de más conocerlo mínimamente en su estado actual.

Por otra parte, XML es el paradigma por antonomasia actual para trabajar con datos semiestructurados, por lo que también consideramos relevante un mínimo conocimiento sobre sus posibilidades por parte del alumno.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Que el alumno experimente el uso práctico de sistemas de almacenamiento de información que no utilizan el modelo lógico relacional.
- Aprender a realizar consultas sobre los datos almacenados en documentos XML.
- Comprender los fundamentos de las bases de datos orientadas a objetos.

Entorno

Para realizar esta práctica es necesario instalar el siguiente software, disponible tanto para Linux como para Windows y Mac OS X:

- API de base de datos orientada a objetos *db4o* (se puede descargar de Faitic)
- Editor de XML *editix* (http://www.editix.com)

Bases de datos orientadas a objetos

Introducción

Las bases de datos orientadas a objeto, aún necesitan evolucionar en diferentes aspectos para gestionar eficientemente grandes cantidades de información. Cuando esta evolución se produzca, para muchos autores, sustituirán a las bases de datos relacionales como principal modelo de gestión de información.

Hoy en día ya existen gestores de bases de datos orientados a objetos que se pueden utilizar cuando la cantidad de información que queremos manejar no es muy elevada, ya que algunos de los principales problemas que presentan actualmente estos sistemas son de eficiencia a la hora de gestionar grandes cantidades de datos.

Como es fácil imaginar, las bases de datos que siguen este paradigma almacenan objetos, que se habrán definido utilizando algún lenguaje. Nosotros vamos a utilizar el gestor de bases de datos *db4o*, que es software libre y que se utiliza como una api de Java, de tal manera que el gestor trabaja directamente con objetos Java.

Su instalación es muy sencilla: sólo hay que descargarlo de Faitic e incluirlo como librería en aquellos proyectos Java en que queramos utilizarlo.

Creando una base de datos

Las bases de datos orientadas a objetos, como acabamos de indicar, almacenan objetos. En este primer ejemplo vamos a crear una base de datos sencilla que almacene objetos de un solo tipo.

La secuencia típica de trabajo con *db4o* es:

1. Crear una configuración para la conexión a la base de datos. Por ahora utilizaremos una configuración por defecto.
2. Conectar a la base de datos utilizando la configuración anterior.
3. Operar sobre la base de datos
4. Cerrar la conexión.

Es decir, una sesión de trabajo, tendrá la forma:

```
Configuration config = Db4o.configure();
ObjectContainer db =
        Db4o.openFile(config,"student.db4o");
…
…
db.close();
```

En el segundo paso, si la base de datos indicada existe se abre, si no existe se crea un nueva con el nombre de fichero indicado.

El objeto `db` se utiliza para realizar todas las operaciones sobre la base de datos (añadir, modificar, eliminar y consultar). En concreto, para almacenar objetos se utiliza el método `store()`, al que se le pasa como parámetro el objeto que queremos almacenar.

Ejercicio 1:

Crea un programa java con una clase `Peliculas` que tenga 4 atributos: un ID entero, un título, una nacionalidad y un número real entre 0.0 y 5.0 que representará tú interés por la película.

Esta clase debe tener los siguiente métodos: Un constructor al que se le pasen los 4 parámetros, métodos "`get`" y "`set`" para todos los atributos, y un método "`toString`".

Crea una base de datos y almacena en ella varias instancias de la clase anterior.

Consultas sencillas

Disponemos de diversos métodos para realizar consultas. Nosotros, con el fin de conocer otro paradigma de definición de consultas distinto del algebra relacional en que está basado SQL, vamos a utilizar el paradigma "*Query by example*" (QBE), en el que las consultas se realizan especificando valores de búsqueda para algunos de los atributos que forman nuestras tuplas o, en este caso, objetos.

Para ello, *db4o* nos proporciona un método de la clase `ObjectContainer` denominado `queryByExample()`.

Si a este método le pasamos como parámetro el nombre de la clase de objetos que queremos recuperar, nos devolverá todos los objetos de esa clase almacenados.

Existen varias versiones del método, aunque la más útil es la que utiliza *generics* de Java y nos devuelve una lista de objetos de la clase que indiquemos.

Ejercicio 2:

Escribe un programa que consulte la base de datos que creaste en el ejercicio anterior e imprima su contenido.

Consultas con valores

Si en vez de buscar todos los objetos de una determinada clase, queremos obtener sólo los que cumplan un determinado criterio de búsqueda, también podemos hacerlo con el método `queryByExample()`, pero, en este caso, pasándole como parámetro un objeto concreto de la clase correspondiente, que tendrá valores para aquellos atributos que sean significativos para la búsqueda.

Que un atributo no es significativo para la búsqueda se indica con valores 0 ó 0.0 para los campos numéricos y con el valor `null` para referencias a objetos, como puede ser a un String)

Ejercicio 3:

Crea un programa que imprima las películas españolas.

Modificar y eliminar objetos

Modificar y eliminar objetos en db4o es bastante sencillo.

En ambos casos, lo primero que tenemos que hacer es recuperar el objeto a modificar o eliminar.

Una vez tenemos una referencia al objeto, si lo que queremos es realizar una modificación, realizamos la modificación sobre ese objeto y lo volvemos a almacenar en la base de datos llamando a `store()`.

Para eliminar un objeto de la base de datos, una vez recuperado dicho objeto, simplemente tenemos que llamar al método `delete()`, pasando dicho objeto como parámetro.

Ejercicio 4:

Crea una programa que gestione la base de datos de películas, permitiendo añadir y eliminar películas, listarlas todas y modificar sus valoraciones.

Bases de datos con asociaciones

En una base de datos orientada a objetos, las entidades se representan mediante objetos y las asociaciones mediante atributos de dichos objetos. Así:

- Una asociación 1-1 se resuelve mediante un atributo simple que sea una referencia al objeto de la otra clase con el que el objeto actual se relaciona.
- Una asociación 1-N se resuelve asociando al objeto correspondiente un atributo que referencie un conjunto de objetos de la otra clase (se recomienda utilizar la clase Java parametrizable `Vector`)
- Una asociación M-N se resuelve definiendo una nueva clase con referencias a las dos clases que representan los conjuntos entidad participantes.

Ejercicio 5:

Crea una base de datos orientada a objetos que almacene el esquema de la base de datos que hemos estado utilizando en las dos últimas prácticas, y que almacena: `Películas`, `Directores` y `Actores`.

Popula dicha base de datos con algunos datos.

Consultas nativas

Ya hemos visto cómo realizar consultas que sigan el paradigma QBE. Este tipo de consultas son muy sencillas de definir, pero presentan limitaciones a la hora de realizar operaciones de consulta complejas. Así, no permiten utilizar los valores, 0, 0.0 o `null` en las consultas, ni tampoco permiten combinar predicados con "`and`", "`or`" o "`not`".

db4o soporta otro tipo de consultas con más posibilidades, y que se denominan *nativas*. Estas consultas permiten ejecutar código sobre todas las instancias de una clase almacenadas en la base de datos y seleccionar el subconjunto de instancias que satisfacen el criterio expresado por el código.

Para ello debemos utilizar el método `query()` del objeto `ObjectContainer`. Este método toma como parámetro un objeto que debe encapsular un método que se denomine `match()` y que devuelva un valor `Boolean`. El gestor de bases de datos ejecutará dicho método sobre todas las instancias almacenadas en la base de datos y devolverá una lista con aquellas para las que `match()` devuelva verdadero.

Puesto que este tipo de consultas se implementan mediante código Java, éstas podrán ser tan complejas como queramos.

Ejercicio 6:

Realiza la siguiente consulta sobre tu base de datos utilizando consultas nativas: Obtener todas las películas estadounidenses con más de 3 actores y cuyo título tenga más de 10 caracteres.

Consultas SODA (*Simple Object Database Access*)

Un último método de consulta soportado por *db4o* permite consultas dinámicas y se basa en el acceso al grafo de datos de la base de datos orientada a objetos.

Para realizar este tipo de consultas se utilizan instancias de la clase `Query`, que posteriormente se ejecutan sobre la base de datos.

Estas consultas se definen a bajo nivel y se utilizan bastante menos que las consultas nativas, que son, con diferencia, las más utilizadas.

Ejercicio 7:

Realiza la siguiente consulta sobre tu base de datos utilizando consultas SODA: Obtener todos los actores y directores que no sean estadounidenses.

Almacenamiento semiestructurado (XML)

Introducción

Tal como hemos visto en teoría, el lenguaje más utilizado hoy en día para modelar datos de forma semiestructurada es XML. En esta parte de la práctica veremos cómo podemos trabajar sobre los datos que almacenan estos ficheros, considerándolos como sistemas de información.

Obviamente, como los ficheros XML son ficheros de texto, el añadir, eliminar y modificar datos en un fichero XML son operaciones que se realizan de manera inmediata y directa mediante cualquier procesador de texto. Por tanto, en esta práctica nos vamos a centrar en algunos ejemplos (básicos) de consultas.

Para esta parte de la práctica vamos a necesitar un editor de XML que soporte consultas. Por desgracia no existen editores con las funcionalidades que necesitamos y que sean software libre. Por tanto, vamos a utilizar uno de los mejores editores comerciales que existen, denominado *editix* y que dispone de una versión de evaluación de 30 días que nos permitirá hacer la práctica. De este editor existen versiones para Linux, Windows y Mac OS, y se puede descargar de[1]: http://www.editix.com/download.html.

[1] Existe una versión gratuita de este editor, pero no soporta las funcionalidades que necesitamos.

Bases de datos XML

Mediante los elementos y atributos que proporcionan estructura a la información podemos modelar entidades y atributos, aunque con unas posibilidades de introducir restricciones de integridad muchos más limitadas que con el modelo relacional.

Así, podemos modelar entidades como elementos XML y asociaciones entre dos entidades mediante elementos y/o atributos XML.

Ejercicio 8:

Descarga de la plataforma Faitic el fichero *empresa.xml*, que contiene la descripción de una empresa en XML.

Observa la estructura que define dicho fichero y dibuja el diagrama E-A de la base de datos que está modelando.

Consultas con XPATH

XPath es una especificación que permite navegar y localizar nodos en un documento XML. No es, por tanto, un lenguaje de consulta en sentido estricto, pero las expresiones que se definen se utilizan en otros lenguajes de consulta como XQuery.

Básicamente, XPath permite especificar un conjunto de nodos de la estructura en árbol de un documento XML de manera similar a como se especifican ficheros en un sistema de ficheros como puede ser el de UNIX.

Las expresiones en XPath tienen, por tanto, el siguiente formato básico:

```
/BDempresa/departamentos/departamento
```

Este ejemplo referencia a todos los elementos "departamento" que son subelementos de "departamentos", que a su vez son subelementos del elemento raíz "BDempresa" de nuestro documento de ejemplo.

Algunos elementos básicos más de la sintaxis de XPath son:

- Las expresiones pueden ser absolutas (desde el elemento raíz, y empiezan por /) o relativas.
- Para acceder a un atributo se utiliza el símbolo `@` antes del nombre del atributo.
- Para acceder a los nodos de texto del documento XML se utiliza la función `text()`
- El símbolo * representa a cualquier elemento en una expresión.
- `//` significa "este nodo o su descendiente", lo que permite acceder a nodos que estén a cualquier nivel del árbol.
- Con `[]` se pueden expresar predicados que deben cumplir los nodos. Estos predicados pueden ser compuestos, utilizando "`and`", "`or`" y "`not`".

Con estos elementos sintácticos básicos (XPATH define más elementos y funciones), podemos realizar consultas básicas. Por ejemplo:

```
/BDempresa/proyectos/proyecto[@departamentoResponsable=6]
```

referencia todos los proyectos cuyo departamento responsable sea "Software"

Ejercicio 9:

Abre el documento de ejemplo con el editor *editix*, abre la vista XPath (menú XML) y realiza las siguientes búsquedas:

1. Encontrar todos los elementos que representan departamentos.
2. Encontrar los empleados que se llaman "Juan"
3. Encontrar los empleados que ganan más de 40.000 €.
4. Encontrar todos los empleados que vivan en Málaga[2].
5. Encontrar todos los empleados que trabajan para el departamento "Hardware" y son de sexo masculino y tienen un familiar de sexo femenino.

[2] La función `contains(elemento, String)` permite referenciar los elementos que contienen el string indicado.

Consultas con XQuery

XQuery es el lenguaje de consulta estándar del W3C para consultar datos XML. Es un lenguaje de alto nivel, que soporta expresiones aritméticas y lógicas, funciones predefinidas, asignaciones, … Es un lenguaje complejo del que sólo vamos a ver un ejemplo sencillo, ya que sólo introducir su sintaxis necesitaría más de una práctica completa.

Las expresiones más habituales para expresar consultas con XQuery se denominan "*expresiones FLWOR*"[3], que toman como base expresiones XPath, y que tienen una forma parecida a las consultas SQL.

Ejercicio 10:

Examina la siguiente consulta XQuery e intenta averiguar qué busca:

```
let $d:=doc("./empresa.xml")
for $e in $d/BDempresa/empleados/empleado
where starts-with($e/apellidos,"G")
order by $e/@dni
return <empleadoG>{$e/@dni} {$e/nombre}</empleadoG>
```

Abre el documento de ejemplo con el editor *editix*, abre la vista XQuery (menú XML) y ejecuta la consulta anterior para comprobar si tu conclusión era correcta.

Ejercicio 11:

Intenta construir las consultas del ejercicio 9 utilizando XQUERY.

[3] FLOWR = for, let, where, order by, return

Resumen

Los principales resultados esperados de esta práctica son:

- Conocer otros paradigmas de sistemas de información.
- Comprobar el funcionamiento del paradigma de bases de datos orientadas a objetos.
- Conocer las posibilidades de consulta sobre los datos de un documento XML.

Como trabajo adicional del alumno, se proponen las siguientes líneas:

- Experimentar con la opción *--xml* de *mysqldump* para obtener ficheros XML a partir de nuestras bases de datos relacionales.
- Profundizar en las posibilidades de XPath y XQuery.