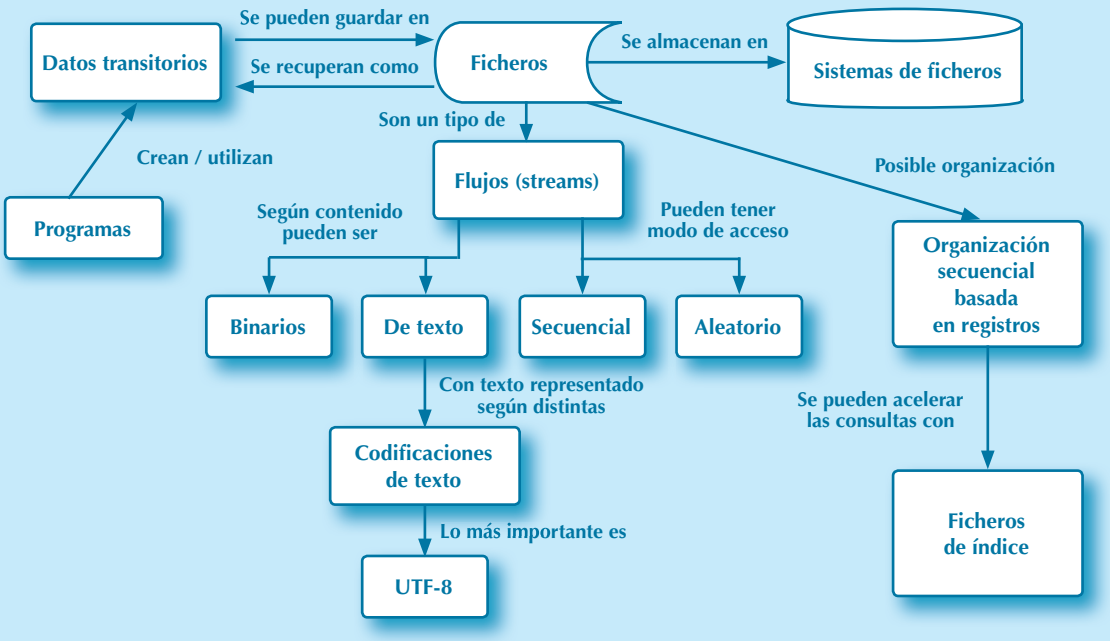
2

Ficheros

Objetivos

- ✓ Conocer las diferencias en la gestión de ficheros de texto y ficheros binarios.
- ✓ Diferenciar entre acceso secuencial y aleatorio a ficheros.
- ✓ Aprender las principales clases de Java para manejo de ficheros (y flujos en general) y su uso.
- ✓ Comprender el mecanismo de *buffering,* cómo permite acelerar las operaciones de lectura y escritura en ficheros tanto binarios como de texto, y cómo permite la lectura y escritura por líneas en ficheros de texto.
- ✓ Acceder correctamente a los contenidos de ficheros de texto con distintas codificaciones.
- ✓ Analizar algunas organizaciones sencillas de ficheros y la manera en que se pueden utilizar para la persistencia de datos.

Mapa conceptual



Glosario

Acceso aleatorio. Tipo de acceso a un fichero que permite acceder directamente a los datos situados en cualquier posición del fichero.

Acceso secuencial. Tipo de acceso a un fichero con el que la única manera de acceder a los datos situados en una posición determinada es leer todos los contenidos desde el principio hasta dicha posición.

Codificación de texto. Una manera particular de representar una secuencia de caracteres de texto mediante una secuencia de *bytes*.

Fichero de texto. Fichero que contiene texto.

Fichero. Unidad fundamental de almacenamiento. Consiste en una secuencia de *bytes*. Con una adecuada organización, se puede utilizar para almacenar cualquier tipo de información.

Índice. Fichero que permite recorrer los contenidos de otro fichero en un orden determinado.

Registro. Estructura para representar información. Consta de una serie de campos, en cada uno de los cuales se puede almacenar un dato particular.

2.1. Persistencia de datos en ficheros

Los ficheros son el método de almacenamiento de información más elemental. De hecho, en última instancia, todos los métodos de almacenamiento, por sofisticados que sean, almacenan los datos en ficheros.

Hasta que fueron relegados en los años ochenta por las bases de datos relacionales, los ficheros fueron el principal medio de almacenamiento de datos. El nombre *fichero* se utilizó por analogía con los antiguos ficheros que contenían fichas de papel, todas con la misma estructura, consistente en un conjunto fijo de campos. En el caso de los libros de una biblioteca, por ejemplo, los campos podían ser el título, nombre del autor, tema, etc. Los ficheros que contenían fichas de papel fueron reemplazados por ficheros de ordenador que contenían registros, equivalentes a las antiguas fichas de papel. Un registro contiene un conjunto de campos de longitud fija. Para acelerar las búsquedas se empezaron a utilizar ficheros auxiliares de índice que permitían acceder a los registros según un orden determinado. IBM desarrolló un avanzado sistema de gestión de ficheros llamado ISAM *(indexed sequential access method)*. El lenguaje COBOL, creado en 1959, pero aún hoy ampliamente utilizado en determinados ámbitos (por ejemplo, el sector bancario), proporciona un excelente soporte para ficheros indexados.

Los ficheros como medio de almacenamiento masivo de datos fueron relegados progresivamente en favor de las bases de datos relacionales. En realidad, las bases de datos relacionales utilizan internamente sofisticados sistemas de gestión de ficheros para el almacenamiento de los datos.

En este capítulo se presentarán algunas organizaciones sencillas de ficheros, y se explicará todo lo necesario para escribir sencillos programas en Java para consultar, añadir, borrar y modificar la información contenida en ellos.

TOMA NOTA

El almacenamiento de datos en ficheros de texto con organizaciones sencillas puede ser una solución perfectamente válida para algunas aplicaciones, pero nunca hay que perder de vista sus limitaciones intrínsecas y su limitada escalabilidad. Si hay que realizar consultas complejas o que requiere relacionar mucha información diversa, será difícil escribir un programa para realizarlas. Si el volumen de datos para manejar es muy grande, o si es necesario realizar con mucha frecuencia operaciones de borrado o modificación de datos, el rendimiento será muy pobre. Permitir que varios programas realicen a la vez operaciones de consulta y actualización puede introducir inconsistencias en los datos e incluso dañar los ficheros. Para evitar estos problemas son necesarios elaborados mecanismos de control de acceso que añaden mucha complejidad al sistema, y mucho más complicado es añadir soporte para transacciones. Es difícil establecer y hacer que se cumplan restricciones de integridad sobre los datos. Y también es difícil evitar redundancias en los datos que, además de desperdiciar espacio de almacenamiento, puede hacer que surjan inconsistencias cuando se añaden o modifican datos. Porque si la misma información está en más de un lugar, puede acabar teniendo un valor distinto en cada uno.

Se siguen utilizando ficheros para la persistencia de datos en muchas aplicaciones, y muy importantes. Las bases de datos relacionales han reemplazado los sistemas basados en ficheros para grandes colecciones de datos con una estructura muy regular, lo que es muy importante, pero no lo es todo. Hoy en día se utilizan mucho los ficheros en formato XML (al que se dedicará un capítulo posterior) para el almacenamiento de todo tipo de datos. Los sistemas de

correo electrónico suelen mantener sus datos en ficheros con un formato relativamente sencillo. Muchos procesos masivos que se ejecutan periódica o puntualmente se realizan basándose en datos proporcionados en ficheros de texto con una estructura muy sencilla. También se usan ficheros de texto sencillos para exportación e importación de datos entre sistemas, y también para copias de seguridad. Aparte de eso, está la infinidad de aplicaciones que almacenan los datos con los que trabajan en ficheros con infinidad de formatos diferentes.

2.2. Tipos de ficheros según su contenido

Un fichero es simplemente una secuencia de *bytes*, con lo que en principio puede almacenar cualquier tipo de información (véase figura 1.3).

Un fichero se identifica por su nombre y su ubicación dentro de una jerarquía de directorios.

Los ficheros pueden contener información de cualquier tipo, pero a grandes rasgos cabe distinguir entre dos tipos: los ficheros de texto y los ficheros binarios:

1. *Ficheros de texto:* contienen única y exclusivamente una secuencia de caracteres. Estos pueden ser caracteres visibles tales como letras, números, signos de puntuación, etc., y también espacios y separadores tales como tabuladores y retornos de carro. Su contenido se puede visualizar y modificar con cualquier editor de texto, como por ejemplo el bloc de notas en Windows o gedit en Linux.
2. *Ficheros binarios:* son el resto de los ficheros. Pueden contener cualquier tipo de información. En general, hacen falta programas especiales para mostrar la información que contienen. Los programas también se almacenan en ficheros binarios.

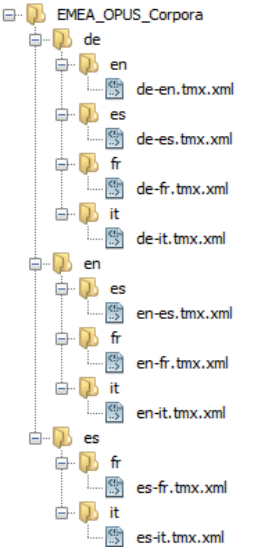


Figura 2.1 Ficheros en una jerarquía de directorios

2.3. Codificaciones para texto

Aunque la cuestión de las codificaciones para texto se ha incluido en este capítulo dedicado a los ficheros, es relevante para cualquier medio de almacenamiento de datos porque, allá donde se almacene un texto, debe hacerse con una codificación determinada.

Un texto es una secuencia de caracteres. Un texto, como cualquier tipo de información, se almacena en memoria o en cualquier dispositivo de almacenamiento como una secuencia de *bytes*. Una codificación es un método para representar cualquier texto como una secuencia de *bytes*. El mismo texto, según la codificación empleada, se puede representar como una secuencia de *bytes* distinta.

La variedad de codificaciones es un problema cada vez menos importante en la práctica, debido a la implantación general de Unicode y su codificación UTF-8, pero todavía es relativamente frecuente encontrar textos con otras codificaciones, sobre todo en entornos Windows.

Las más frecuentes son ISO 8859-1 y Windows-1252, que se puede considerar una variante no estándar de Microsoft del estándar ISO 8859-1. UTF-8 es compatible con el código ASCII,

lo que significa que cualquier texto codificado en ASCII se representa exactamente igual en UTF-8. Este ha sido un motivo fundamental para la adopción generalizada de UTF-8.

Para las nuevas aplicaciones, siempre hay que utilizar UTF-8 para almacenar texto, a no ser que haya alguna razón de peso para utilizar otra, que normalmente no la hay. A veces hay que importar u obtener textos de otras fuentes, desde donde podrían venir con otras codificaciones. Hay que identificar estas situaciones y hacer la conversión necesaria para recodificar los textos.

Recurso digital

En el anexo web 2.1, disponible en www.sintesis.com y accesible con el código indicado en la primera página del libro, encontrarás información sobre la codificación del contenido de ficheros.

RECUERDA

Cualquier editor de texto debería permitir visualizar correctamente un fichero de texto independientemente de su codificación. Por ejemplo, Notepad en Windows y gedit en Linux. Normalmente, con la opción "Guardar como…" se puede seleccionar la codificación que se va a utilizar, y UTF-8 suele aparecer como una de las opciones.

La manera más fiable y segura de confirmar el tipo de un fichero en Linux es con el comando `file`. Este analiza los contenidos del fichero e indica su tipo. Si es de texto, indica la codificación utilizada para el texto, típicamente: ASCII, UTF-8 o iso-8859-1.

El comando `iconv` de Linux permite recodificar ficheros de texto. El siguiente comando, por ejemplo, se podría utilizar para recodificar a UTF-8 un fichero codificado en ISO 8859-1.

```
iconv -f iso8859-1 -t utf-8 fichero_8859-1.txt > fichero_utf8.txt
```

2.4. La clase File de Java

La versión de Java utilizada para este libro es Java SE 8, una versión LTS (*long term support*, es decir, con soporte a largo plazo). En los servidores web de Oracle se puede consultar la documentación de la biblioteca estándar de clases de Java SE 8.

Las clases que permiten trabajar con ficheros están en el paquete `java.io`.

Recurso web

Se recomienda consultar en http://docs.oracle.com/javase/8/docs/api la documentación de Java SE8 para más información acerca de cualquier clase utilizada en este capítulo. Arriba, a la izquierda, aparece la lista de paquetes por orden alfabético. En ella se puede localizar el paquete `java.io`. Si se pulsa sobre él, aparece debajo la lista de interfaces y clases que contiene el paquete.