Estructuras de Datos y de la Información

Práctica 6

Indexación

OBJETIVO: El objetivo de la práctica es implementar la técnica de indexación sobre un fichero de registros utilizando un índice simple implementado en almacenamiento secundario.

FECHA: Esta práctica se entregará los días 18, 19 y 20 de enero.

ENUNCIADO: Implementar el tipo de dato `FileIndex` que hereda las operaciones definidas para el tipo `FileBuf`, y mantiene los registros del fichero de datos indexados por una clave primaria. Para ello, la clase `FileIndex` encapsula un índice primario implementado sobre un `FileSort` (fichero de registros ordenados).

El campo `ISBN` se usará como clave primaria del tipo `Libro` para implementar el índice primario.

Modificar el programa desarrollado para la práctica 4 para que trabaje con objetos `FileIndex<Libro, KeyISBN>` y presente el siguiente menú:

```
1.- Abrir/crear fichero.
2.- Mostrar todos los registros.
3.- Insertar un registro.
4.- Borrar un registro. (Opcional)
5.- Cerrar fichero.
0.- Salir.
```

A continuación se describe el comportamiento del programa al seleccionar las opciones.

- En la opción 1 se solicita el nombre del fichero de datos y si existe se abre. Si no existe se crea el fichero de datos y el fichero índice.
- En la opción 2 se muestran todos los registros del fichero por pantalla ordenados por el valor de la clave primaria.
- En la opción 3 se solicitan los datos de un registro desde teclado. Si no existe se inserta en el fichero.
- En la opción 4 se solicita el valor de ISBN del registro a eliminar. Se borra el registro colocando una marca en el índice primario.
- En la opción 5 se cierra el fichero de trabajo y se coloca una marca en el registro cabecera del fichero de datos para indicar que el índice está sincronizado. De esta forma, si al abrir un fichero de datos no se encontrara la marca de sincronización habría que generar el índice desde el fichero de datos.

NOTAS DE IMPLEMENTACION:

- Definir el tipo de dato abstracto `PrimaryIndex` que describe las operaciones para implementar un índice primario. Se trata de un tipo de dato genérico (`template`) que depende del tipo de la clave primaria (`KEY`). Además es una clase abstracta que sólo tiene métodos puros.
- Un caso particular de índice primario (`PrimaryIndex`) es la clase Simple`FileIndex<KEY>` que se implementa mediante un fichero ordenado `FileSort<PrimaryIndexEntry>`.
- Se define la clase `KeyISBN` para encapsular un ISBN (char[11]) y las operaciones necesarias para utilizarlo como clave.

```
class KeyISBN {
  char ISBN[TAM_ISBN];
  KeyISBN(const char* key) {strncpy(ISBN, key, TAM_ISBN);}
  bool operator==(const KeyISBN& key) {
    return strncmp(ISBN, key, TAM_ISBN) == 0;
  }
 ...
};
```

Estructuras de Datos y de la Información

- Se define la estructura `PrimaryIndexEntry` para representar una entrada en el índice primario `{KEY, TPointer}`, siendo `KEY` un objeto de tipo `KeyISBN`, y por tanto, un registro de longitud fija que se almacena de forma ordenada en el fichero índice.

```
template <class KEY>
class PrimaryIndex {
  public:
    // Inserta una nueva entrada (key, ref) en el índice.
    virtual void Insert(const KEY& key, const TPointer ref) = 0;

    // Elimina la entrada con valor de clave key del índice.
    // Retorna false si no se encuentra la entrada.
    virtual bool Remove(const KEY& key) = 0;

    // Busca la clave key y devuelve en el parámetro ref la referencia
    // asociada. Posiciona un apuntador a la entrada seleccionada.
    // Retorna true si la encuentra, y false en otro caso.
    virtual bool Search(const KEY& key, TPointer& ref) = 0;

    // Avanza el apuntador y devuelve en el parámetro ref la referencia
    // asociada a la siguiente entrada del índice.
    // Retorna false cuando alcanza el final del índice.
    virtual bool Next(TPointer& ref) = 0;

    // Se posiciona un apuntador sobre la primera entrada del índice.
    virtual void Rewind() = 0;
};

template <class KEY>
class SimpleFileIndex: public PrimaryIndex<KEY> {
  public:
    // Constructores y destructores
    SimpleFileIndex(const char *file);
    ~SimpleFileIndex();

    // Código para los métodos virtuales
    ...
  private:
    struct PrimaryIndexEntry {
      KEY key;
      Tpointer ref;
    };
    FileSort<PrimaryIndexEntry> Index;   // Fichero indice
    ...
};


template <class RECORD, class KEY>
class FileIndex: public FileBuf<RECORD> {

  int Open(char *file, int mode = ios::in | ios::out |
                     ios::binary | ios::nocreate) {
    FileBuf::Open(file);
    char file_idx[50];
    strcpy(file_idx, file);
    strcat(file_idx, ".idx");
    Index = new SimpleFileIndex(file_idx);
  }
  protected:
    PrimaryIndex<KEY> *Index;
};
```