2. Multi-Linked Lists *(Section 4.5)*

Doubly-linked lists are a special case of Multi-linked lists; it is special in two ways:

- each node has just 2 pointers
- the pointers are exact inverses of each other

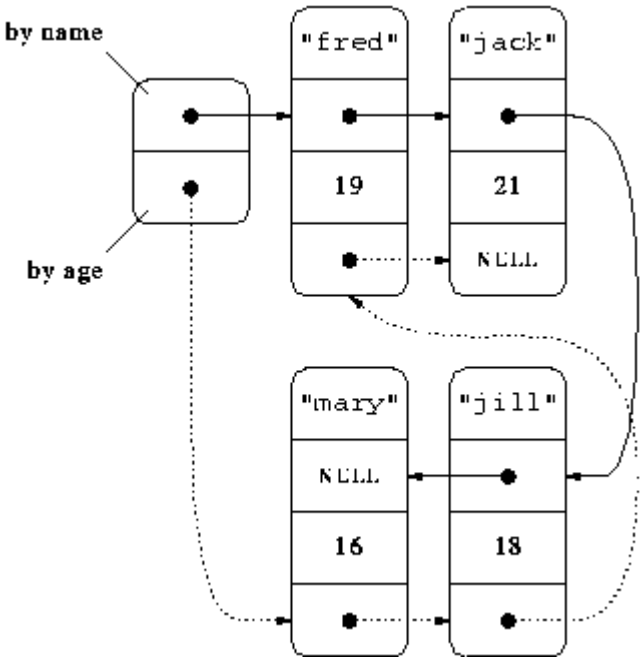
In a general multi-linked list each node can have any number of pointers to other nodes, and there may or may not be inverses for each pointer.

2.1. Example 1: Multiple Orders Of One Set Of Elements

The standard use of multi-linked lists is to organize a collection of elements in two different ways. For example, suppose my elements include the name of a person and his/her age. e.g.

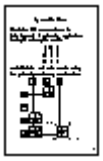
(FRED,19) (MARY,16) (JACK,21) (JILL,18)

I might want to order these elements alphabetically and also order them by age. I would have two pointers - *NEXT-alphabetically*, *NEXT-age* - and the list header would have two pointers, one based on name, the other on age.



Inserting into this structure is very much like inserting the same node into two separate lists. In multi-linked lists it is quite common to have back-pointers, i.e. inverses of each of the forward links; in our example this would mean that each node had four pointers.

2.2. Example 2: Sparse Matrices *(Section 6.4)*



A second very common use of multi-linked lists is sparse matrices. A sparse matrix is a matrix of numbers, as in mathematics, in which almost all the entries are zero. These arise frequently in engineering applications. the use of a normal Pascal array to store a sparse matrix is extremely wasteful of space - in an NxN sparse matrix typically only about N elements are non-zero. For example:

```
     X = 1  2  3
       ----------
Y=1  |  0 88  0
Y=2  |  0  0  0
Y=3  | 27  0  0
Y=4  | 19  0 66
```

We can represent this by having linked lists for each row and each column. Because each node is in exactly one row and one column it will appear in exactly two lists - one row list and one column. So it needs two pointers: *Next-in-this-row* and *Next-in-this-column*. In addition to storing the data in each node, it is normal to store the co-ordinates (i.e. the row and column the data is in in the matrix). Operations that set a value to zero cause a node to be deleted, and vice versa. As with any linked list in practice is common for every pointer to have a corresponding back pointer.