Index File Structure (.idx)

Visual Studio .NET 2003

Index files contain one header record and one or many node records. The header record contains information about the root node, the current file size, the length of the key, index options and signature, and printable ASCII representations of the key[1] and FOR expressions. The header record starts at file position zero.

The remaining node records contain an attribute, number of keys present, and pointers to nodes on the left and right (on the same level) of the current node. They also contain a group of characters encompassing the key value and either a pointer to a lower level node or an actual table record number. The size of each record that is output to a file is 512 bytes.

Index Header Record

Byte offset	Description
00 – 03	Pointer to the root node
04 – 07	Pointer to the free node list (-1 if not present)
08 – 11	Pointer to the end of file (file size)
12 – 13	Length of key
14	Index options (any of the following numeric values or their sums): 1 – a unique index 8 – index has FOR clause
15	Index signature (for future use)
16 – 235	Key expression (uncompiled; up to 220 characters)[1,3]
236 – 455	FOR expression (uncompiled; up to 220 characters ending with a null value byte)
456 – 511	Unused

Index Node Record

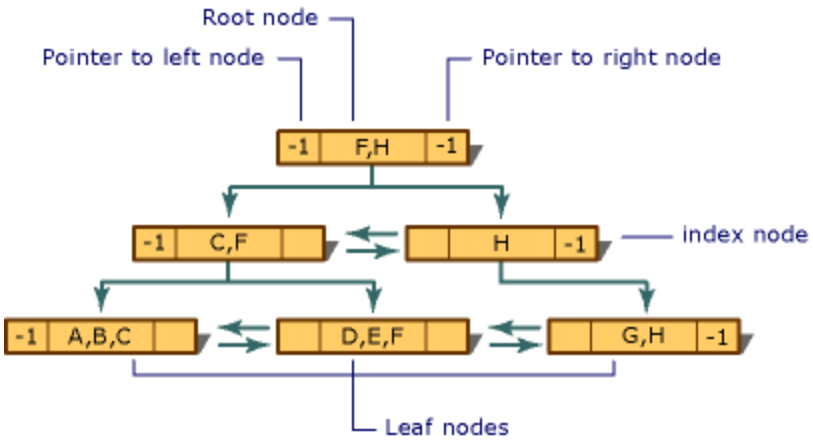
Byte offset	Description
00 – 01	Node attributes (any of the following numeric values or their sums): 0 – index node 1 – root node 2 – leaf node
02 – 03	Number of keys present (0, 1 or many)
04 – 07	Pointer to the node directly to left of the current node (on same level; -1 if not present)
08 – 11	Pointer to the node directly to right of the current node (on same level; -1 if not present)
12 – 511	Up to 500 characters containing the key value for the length of the key with a four-byte hexadecimal number stored in normal left-to-right format: If the node is a leaf (attribute = 02 or 03) then the four bytes contain an actual table number in hexadecimal format; otherwise, the 4 bytes contain an intra-index pointer.[2] The key/four-byte hexadecimal number combinations will occur the number of times indicated in bytes 02 – 03.

1 The type of the key is not stored in the index. It must be determined by the key expression.
2 Anything other than character strings, numbers used as key values, and the four-byte numbers in the leaf node are represented in reversed bytes (Intel 8086 format).
3 Numbers are a special case when used as a key. They are converted through the following algorithm so they can be sorted using the same ASCII collating sequence as characters:

- Convert the number to IEEE floating point format. For details, see Visual FoxPro System Capacities.
- Swap the order of the bytes from Intel 8086 order to left-to-right order.
- If the number was negative, take the logical complement of the number (swap all 64 bits, 1 to 0 and 0 to 1); otherwise, invert only the leftmost bit.

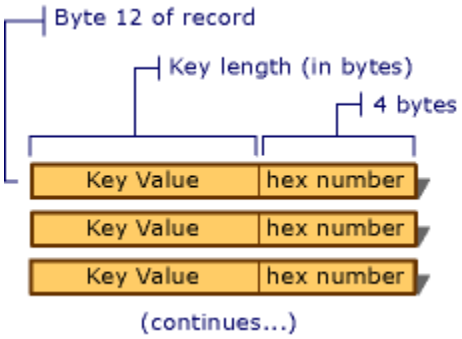
Example of an Ordered Tree Structure

Finding a key in the structure below requires searching a single path between the root and leaf nodes. Nodes at the lowest level are leaf nodes. Because the keys are sorted, all keys in the sub-tree are less than or equal to the parent node.



In the illustration above, the letters are used as the key values. Each key would also have a four-byte hexadecimal number. The numbers associated with the keys in the *leaf* nodes would be actual table numbers — all keys in other nodes would have intra-index pointers associated with them.

Bytes 12-511 in the index node record could be viewed as follows:



The key value/hexadecimal number combination occurs in bytes 12 – 511 n times where n is the number of keys present.

See Also

Compact Index File Structure (.idx) | Compound Index File Structure (.cdx) | Table File Structure (.dbc, .dbf, .frx, .lbx, .mnx, .pjx, .scx, .vcx) | Table Structures of Table Files (.dbc, .frx, .lbx, .mnx, .pjx, .scx, .vcx) | Memo File Structure (.FPT) | Macro File Format (.fky) | File Extensions and File Types