In Lianja v3.0 we have extended the xquery() functions to better handle the parsing of large XML documents and also handle element attributes.

The following functions have been added:

```
  xquery_open()
xquery_count()
xquery_node()
xquery_select()
xquery_decode()
xquery_close();
```

as well as the existing ones:

```
  xquery()
xquery_file()
```

e.g.

```
data = xquery_file("books.xml","/bookstore/book[2]/title")
```

Notice how I use book[2] to reference the second book.

You can iterate across the books in a loop. The last call to a non-existent book will return an empty string.

You can of course extract inner XML (and use that string with xquery()) insofar as that it does not exceed 64k.

```
  i = 1
```

```
do while .t.
  data = xquery_file("books.xml","/bookstore/book[&i]/title")
  if empty(data)
    exit
  endif
  i = i + 1
  // process the data
enddo
```

For better performance you can load the XML document into memory (any reasonable size) and perform queries on it.

```
  xquery_open("books.xml")
m_count = xquery_count("/bookstore/book")
for i=1 to m_count
  data = xquery_node("/bookstore/book[&i]/title")
  // process the data
endfor
xquery_close()
```

The xquery_decode(cExp) function returns an object representing the selected XML. Attributes are also handled. See below.

```
  xquery_open("books.xml")
xquery_select("book[@category='children']")
obj = xquery_decode()
? obj
  Dynarray (refcnt=1)
(
    [book] => Dynarray (refcnt=1)
    (
      [1] => Dynarray (refcnt=1)
      (
        [title] => Dynarray (refcnt=1)
        (
          [text] => Harry Potter
          [attributes] => Dynarray (refcnt=1)
          (
            [lang] => en
          )
        )
        [author] => Dynarray (refcnt=1)
        (
          [text] => J K. Rowling
```

```
        [attributes] => Dynarray (refcnt=1)
        (
        )
      )
      [year] => Dynarray (refcnt=1)
      (
        [text] => 2005
        [attributes] => Dynarray (refcnt=1)
        (
        )
      )
      [price] => Dynarray (refcnt=1)
      (
         [text] => 29.99
         [attributes] => Dynarray (refcnt=1)
         (
         )
      )
      [attributes] => Dynarray (refcnt=1)
      (
          [category] => CHILDREN
      )
    )
  )
)
```

You can now reference all the text and attributes of each XML element using standard OOPS notation. Notice how elements are always represented as an array so that you can process them easier.

```
? obj.book[1].title.text
```

Tip: Use the len() built-in function to get the length of an array.

```
 ? len( obj.book )
```

Querying by attributes and values.

To query all nodes that have a specific attribute use the notation [@name=value] e.g.

xquery_select("book[@category='children']")

As well as the = operator you can use:

```
 !=, >, >=,     (
    [1] => Dynarray (refcnt=1)
    (
      [title] => Dynarray (refcnt=1)
      (
        [text] => Harry Potter
        [attributes] => Dynarray (refcnt=1)
        (
          [lang] => en
        )
      )
      [author] => Dynarray (refcnt=1)
      (
        [text] => J K. Rowling
        [attributes] => Dynarray (refcnt=1)
        (
        )
      )
      [year] => Dynarray (refcnt=1)
      (
        [text] => 2005
        [attributes] => Dynarray (refcnt=1)
        (
        )
      )
      [price] => Dynarray (refcnt=1)
      (
         [text] => 29.99
         [attributes] => Dynarray (refcnt=1)
         (
         )
      )
      [attributes] => Dynarray (refcnt=1)
      (
          [category] => CHILDREN
      )
    )
  )
```

)

After the xquery_select() function completes the internal XML document is replaced by the result nodes and non-zero is returned if there are any matching nodes. If there are no matching nodes then 0 is returned. The next call to xquery_select() operates on the previously selected nodes. To reload the XML document use xquery_open() again. Use xquery_close() to free up any resources allocated by xquery_open().

As stated earlier, use xquery_decode() with no argument to return an object that represents the current XML. This provides you with a lot of flexibility for processing XML data as you can reference all the elements (nodes) and their attributes programmaticly using standard OOPS notation.