Let-free Style and Streaming

March 19, 2012 at 12:34 PM | categories: XQuery, MarkLogic | 0 Comments

If you are familiar with Lisp or Scheme, you know that a function call can replace a variable binding, and function calls can also replace most loops. This is also true in XQuery.

```
xquery version "1.0";

let $x := 1
return 1 + $x
```

gistfile1.xq hosted with ♥ by GitHub view raw

In XQuery this leads to a style of coding that I call "let-free". In this style, there are no FLWOR expressions. Really this is "FLWOR-free", not "let-free", but that's too much of a mouthful for me.

But why would you write let-free code?

The answer is scalability - you knew it would be, right? This breaks out into concurrency and streaming. Let's talk about concurrency first. In the MarkLogic Server implementation of XQuery, every `let` is evaluated in sequence. However, other expressions are evaluated lazily with concurrency-friendly "future values". So a performance-critical single-threaded request can sometimes benefit from let-free style. You can see this technique in use in some of my code: the semantic library or the task-server forest rebalancer. Both of these projects try to benefit from multi-core CPUs.

The let-free style can also help with query scalability by allowing the results to stream, rather than buffering the entire result sequence. If you need to export large result sets, for example, this technique can help avoid `XDMP-EXPNTREECACHEFULL` errors. Those errors result when your query's working set is too large to fit in the expanded tree cache, a sort of scratch space for XML trees. But streaming results don't have to fit into the cache.

For example, let's suppose you need to list every document URI in the database. But you do not have the URI lexicon enabled, and you cannot reindex to create it.

```
xquery version "1.0-ml";

(: FLWOR expressions do not stream.
 : If the database is too large,
 : this fails with XDMP-EXPNTREECACHEFULL.
 :)
for $i in doc()
return xdmp:node-uri($i)
```

gistfile1.xq hosted with ♥ by GitHub view raw

```
xquery version "1.0-ml";

(: let-free style,
 : using function-mapping instead of FLWOR.
 : This can stream to avoid XDMP-EXPNTREECACHEFULL errors.
 :)
declare function local:uri($i as document-node())
{
  xdmp:node-uri($i)
};
```

Latest blog posts

- Deduplicating Search Results
- Introduction to Multi-Statement Transactions
- External Variables (Code Review, Part II)
- AlbumMixer v1.13
- rsyslog and MarkLogic

Recommended

- BeerAdvocate
- Jeff Thomas
- marklogic@StackOverFlow
- xquery@StackOverFlow

Apps

- AlbumMixer
- MrCombo

Projects

- Taskbot — Pure XQuery map-reduce processing for MarkLogic
- XQYSP — Customizable pure XQuery search parser for MarkLogic
- cprof — Conditional profiling for MarkLogic XQuery modules
- xquery-mode — Simple XQuery mode for emacs
- threx — Throttled reindexing for MarkLogic
- Corb — Bulk content reprocessing for MarkLogic
- task-rebalancer — Forest rebalancing for MarkLogic
- cq — Content query console for MarkLogic
- PerformanceMeters — Test harness for MarkLogic
- presta — Code management for MarkLogic
- RecordLoader — Bulk load tool for MarkLogic
- semantic — Triple storage for MarkLogic

```
local:uri(doc())
```

gistfile1.xq hosted with ❤ by GitHub view raw

Note that nested evaluations cannot stream, either. So even a let-free query may throw XDMP-EXPNTREECACHEFULL in cq or another development tool. To test this query, use an http module instead. This is ideal for web service implementations too.

In this example we used function mapping, a MarkLogic extension to XQuery 1.0. If a function takes a single argument but is called using a sequence, the evaluator simply maps the sequence to multiple function calls. This is somewhat faster than a FLWOR, and it can stream.

Besides using function mapping, let-free style can use XPath steps. However, this technique only works for sequences of nodes.

```
xquery version "1.0-ml";

(: iterate using XPath steps :)
doc()/xdmp:node-uri(.)
```

gistfile1.xq hosted with ❤ by GitHub view raw

While these techniques are useful, they can make for code that is hard to read and tricky to debug. Function mapping is especially prone to errors that are difficult to diagnose. If a function signature specifies an argument without a quantifier or with the `+` quantifier, and the runtime argument is empty, the function will not be called at all. This is surprising, since normally the function would be called and would cause a strong typing error.

```
xquery version "1.0-ml";

declare function local:test($i as document-node())
{
  xdmp:node-uri($i)
};

local:test(doc('does-not-exist'))
```

gistfile1.xq hosted with ❤ by GitHub view raw

```
xquery version "1.0-ml";

declare option xdmp:mapping "false";

declare function local:test($i as document-node())
{
  xdmp:node-uri($i)
};

local:test(doc('does-not-exist'))
```

gistfile1.xq hosted with ❤ by GitHub view raw

The first expression returns the empty sequence, while the second throws the expected strong typing error `XDMP-AS`. This behavior is annoying, but in some applications the benefits of function mapping outweigh this drawback. We can make debugging easier if we weaken the function signature to `document-node()?` so that the function will be called even when the argument is empty. If needed, we can include an explicit check for empty input too.

Another let-free trick is to use module variables. These act much like `let` bindings, but they can stream.

```
xquery version "1.0-ml";

declare variable $URIS := doc()/xdmp:node-uri(.) ;

$URIS
```

- XQSync — Import and export tool for MarkLogic
- XQUT — XQuery unit tests for MarkLogic

Categories

- AWS (rss) (2)
- Java (rss) (1)
- Javascript (rss) (1)
- Linux (rss) (8)
- MarkLogic (rss) (34)
- New Zealand (rss) (12)
- OS X (rss) (1)
- Performance (rss) (3)
- Thailand (rss) (3)
- UK (rss) (9)
- Uncategorized (rss) (1)
- XQuery (rss) (26)
- beer (rss) (7)
- food (rss) (3)
- home (rss) (21)
- housekeeping (rss) (1)
- iOS (rss) (15)
- jazz (rss) (3)
- recipes (rss) (2)
- travel (rss) (22)
- wine (rss) (6)

Archives

- June 2014 (1)
- June 2013 (1)
- September 2012 (1)
- June 2012 (1)
- May 2012 (2)
- March 2012 (3)
- January 2012 (1)
- December 2011 (2)
- November 2011 (6)
- October 2011 (1)
- September 2011 (1)
- August 2011 (1)
- July 2011 (1)
- June 2011 (1)
- March 2011 (1)
- February 2011 (1)
- January 2011 (1)
- December 2010 (4)
- November 2010 (2)
- October 2010 (3)
- August 2010 (2)
- June 2010 (1)
- May 2010 (3)
- December 2009 (1)
- October 2009 (3)
- September 2009 (4)
- August 2009 (1)
- June 2009 (2)
- May 2009 (1)
- April 2009 (1)
- March 2009 (1)
- February 2009 (2)

gistfile1.xq hosted with ❤ by **GitHub** **view raw**

This example is a bit contrived, since the module variable doesn't add anything. But if you find yourself struggling to refactor a `let` as a function call or an XPath step, consider using a module variable. Module variables are also excellent tools for avoiding repeated work, since the right-hand expression is evaluated lazily and is never evaluated more than once. If the evaluation does not use the module variable, then the right-hand expression is never evaluated. In contrast, the right-expression of a `let` is evaluated even when the `return` does not use its value.

As always, do not optimize code unless there is a problem to solve. There are also some situations where the let-free style isn't appropriate. Aside from making your code harder to read and more difficult to debug, let-free style simply doesn't work in situations where your FLWOR would have an `order by` clause. And after all, streaming won't work for that case anyway. The evaluator can't sort the result set without buffering it first.

0 Comments Where am I? Login

Sort by Best Share Favorite

Start the discussion…

Be the first to comment.

Subscribe Add Disqus to your site

- January 2009 (2)
- December 2008 (1)
- September 2008 (2)
- June 2008 (3)
- May 2008 (3)
- April 2008 (3)
- March 2008 (2)
- February 2008 (1)
- January 2008 (2)
- October 2007 (1)
- August 2007 (1)
- July 2007 (1)
- June 2007 (1)
- May 2007 (2)
- April 2007 (4)
- February 2007 (4)
- January 2007 (2)
- December 2006 (3)
- November 2006 (2)
- October 2006 (2)
- September 2006 (1)
- August 2006 (1)
- July 2006 (2)
- June 2006 (1)
- May 2006 (11)
- April 2006 (1)
- September 2004 (1)

Atom | RSS | Comments