Using Relational Databases with MarkLogic

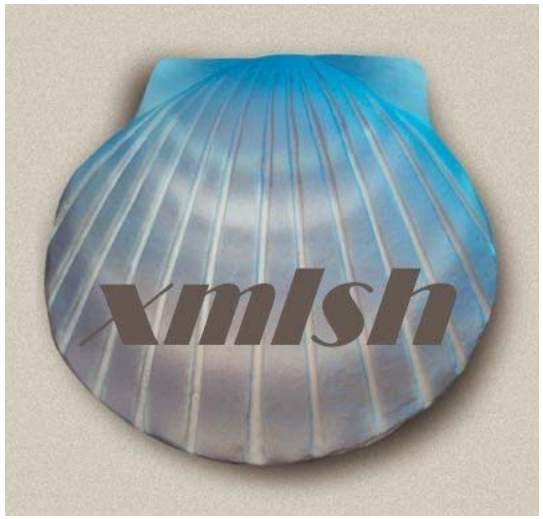
David A. Lee, Lead Engineer, MarkLogic
2012-05-03

Traditional RDBMS Integration

- Custom Code
  - Hand written SQL or language code to query DB
  - Must generate XML prior to load
  - Ingest with XCC, Information Studio, Record Loader, WebDav
- ML/SQL
  - “Pull” interface driven from within MarkLogic App.
  - Requires a J2EE server constantly running
  - Preprocessing within ML
- Information Studio
  - Must prepare files prior to ingestion

MarkLogic

MarkLogic + xmlsh



xmlsh

Motivation

- Unix and Unix Shells were a radical “Paradigm Shift”
  - Vastly simplified access to data and processing
  - Set of small simple core tools
  - Create complexity with hierarchy instead of linearly.

- 40 years later and the core design fundamentals are being eroded
  - Predominant data type is no longer byte/line streams – it is complex content, often XML.
  - Tools and shells have not evolved with the data (XML) – making working with XML too complicated for legacy toolkits.

MarkLogic

xmlsh - Philosophy

- Based on the design principles of the Unix Shells
- Largely backwards compatible syntax to /bin/sh
  - Use cases equivalent to /bin/sh
- Where the Unix Shells use strings and byte streams, xmlsh targets XML documents and XDM values.
- *Scripting with XML data should be as easy and natural as scripting with text files.*
- ***Someday, all data should be XML ...***
  - But until then, intermixing Text, binary and XML is necessary

MarkLogic

XMLSH

Project

- Open Source Project
  - Open Source / Closed Development
  - BSD License “Free Software”
  - No commercial restrictions
  - Hosted on Sourceforge: **xmlsh.sourceforge.net**
  - Main project site: **www.xmlsh.org**
- Extension Modules
  - MarkLogic
  - Relational
  - Amazon AWS
  - others …

MarkLogic

Syntax and Features

- Core syntax equivalent to /bin/sh
  - if ... then ... else ... fi
  - while/until ... do ...
  - case ... in
  - functions
  - variable assignment
  - Pipes
  - subshells and background processes/threads
  - IO redirection
  - script and external process execution

Syntax and Features

- New syntax specific to xmlsh
  - XML expressions and variables

    <[*xquery expr*]>

    $<(*xml producing command*)

  Example:

    foo=<[“hi” , 123 , <elem attr=“attr”>body</elem>]>

    bar=$<(xls)

    for $i in <[1,2,3,<node/>,”hi”]> ; do echo $i ; done

MarkLogic

Architecture

- Variables
  - Dynamically typed variables (text , xml , java object)
  - No limit to size
  - Take on the type of their assignment expression
    - x=“foo” # string
    - x=<[“foo”]> # xml
  - XML type is really a Saxon “XdmValue”
    - atomic type
    - item type
    - sequence
    - Anything that XQuery can produce
  - Java native objects and direct access to Java classes, methods etc.

MarkLogic Extension to xmlsh

createdir	Creates directories
del	Deletes documents
deldir	Deletes directories
direxists	Tests whether a directory exists
exists	Tests whether a document exists
get	Gets documents from the server
get-permissions	List document permissions
invoke	Invokes a remotely stored query
list	Lists documents
list-collections	List all collection URI's
listdir	Lists directories
put	Puts one or more files to the server
move-forest	Move a document from one forest to another
query	Invokes an ad-hoc query
rename	Renames (moves) a document
set-permissions	Set document permissions

MarkLogic

MarkLogic Module Examples

Puts the file "test.xml" using the uri "myfile.xml"

```
import module ml=marklogic
ml:put -uri myfile.xml test.xml
```

Puts the result of an xquery (via stdin) to “test.xml”

```
xquery -q 'myquery.xquery' -i input.xml | ml:put -uri test.xml
```

Recursively puts a directory tree with batches of 100 files and 4 threads

```
ml:put -baseuri / -r -m 100 –t 4 modules
```

xsql command

Executes an SQL command using any JDBC connector and exports the results formatted as xml. May also insert data using XML as input.

MySQL example: select * from books

```
xsql -cp mysql-connector-java-5.1.7-bin.jar -c jdbc:mysql://host.com/xmlsh
    -root books –row book -u xmlsh -p password -d org.gjt.mm.mysql.Driver
    -q 'select * from books'
```

```
<books>
 <book>
  <TITLE>Pride and Prejudice</TITLE>
  <AUTHOR>Jane Austen</AUTHOR>
  <PUBLISHER>Modern Library</PUBLISHER>
  <PUB-DATE>2002-12-31</PUB-DATE>
  <LANGUAGE>English</LANGUAGE>
  <PRICE>4.95</PRICE>
  <QUANTITY>187</QUANTITY>
  <ISBN>679601686</ISBN>
  <PAGES>352</PAGES>
  <DIMENSIONS>8.3 5.7 1.1</DIMENSIONS>
  <WEIGHT>6.10</WEIGHT>
 </book>
...
```

MarkLogic

Put it together

Generate an XML document with all books and store as a single XML file

```
import module ml=marklogic

xsql –cp … –d … –u … –p …  -root books –row book -u xmlsh -p password
    -q 'select * from books‘        |
ml:put -uri /books/allbooks.xml
```

MarkLogic

Try some preprocessing

Preprocess the file with an xslt stylesheet before storing in MarkLogic

```
import module ml=marklogic

xsql –cp … –d … –u … –p …  -root books –row book -u xmlsh -p password
    -q 'select * from books‘        |
xslt –f transform.xsl               |
ml:put -uri /books/allbooks.xml
```

A bit more complex

- **Generate an XML document with all books**
- **Preprocess with a stylesheet**
- **Split it into multiple documents per book**
- **Store as a directory of XML files in batches of 3 using 2 threads**
- **Delete temporary files as transferred**

```
import module ml=marklogic

xsql options  -q 'select * from books‘          |
xslt –f transform.xsl                           |
xsplit –l                                       |
ml:put -r -baseuri /books/ -m 3 -maxthreads 2  -delete -f -
```

MarkLogic

Store to RDBMS

xsql can store data to RDBMS

- ml:get - get documents from MarkLogic
- xsql … –insert - Inserts rows to an RDBMS

Oracle example – store data to RDBMS from an XQuery in ML

```
ml:get /books/book.xml |
xsql -cp ojdbc14.jar -driver oracle.jdbc.OracleDriver -connect
"jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)(FAILOVER=ON)(ADDRESS=(PROTOCOL=
TCP)(HOST=myhost.com)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=myhost2.com)(PORT
=1521))(CONNECT_DATA=(SERVICE_NAME=MYSERVICE)))"
    -user myuser -password mypass -insert
```

Summary

- This is the tip of the iceberg
- Many more things you can do !
- Easy, Efficient, Fun !
- Future Plans – I need your suggestions !

Interested in more ?

Thank you !

Talk to me for Demos and more info!!!

David A. Lee

http://www.xmlsh.org

dlee@marklogic.com

MarkLogic®

Summary: xmlsh + MarkLogic + RDBMS

- Runs on any Java 1.6 VM (Unix,Windows,Mac)
- Support for MarkLogic 4.2 and later
- All operations in-process – single JVM
- Streaming and multithreaded capabilities
- Efficient batch and parallel stores to MarkLogic
- Access to any RDBMS with a JDBC Driver
- Local XML pre and post processing
- No dedicated server required
  - Only a single machine with access to both MarkLogic and RDBMS
  - May be the same as a MarkLogic node or a remote machine
- Full range of processing tools available
  - XSLT, XQuery, XPath, XInclude, Validation, XProc, Text, Binary, Filesystem access
- Integration Options:
  Command line (batch and interactive), Embedded Java, Servlet

MarkLogic