An Overview of Berkeley DB XML (2)

Dan Brian
Conceptuary, Inc.
dan@brians.org

O'Reilly Open Source Convention
August 1 - 5, 2005

About the Presentation

- I'm not going to offer extensive examples for every language.
- I'm not going to delve into some of DB XML's deeper features.
- I'm not going to try to convince you that a native XML database is superior to a relational implementation.
- I'm not going to assume that you are an XML expert.
- I'm not going to say much you don't already know if you've used DB XML.

Why Use an XML Database?

O'REILLY®

Why I Use and Like XML DBs

- Textual markup feels simple and solid; binary protocols impose client and server dev requirements.
- I enjoy working with language-independent (!), text-formatted data.
- A lot of the data I work with is hierarchical.
- XML has tools support.
- XPath, XSLT, XQuery, and associated technologies make a lot of sense to me.
- I'm a Relational DB bigot.

About the Example Project

- I dislike mock and generic code examples.
- Throughout, examples are provided using samples from the Glass Bead Network.
- A game played with game pieces that represent anything: movies, music, places, people, things.
- Each “bead” has a graphical representation, and a corresponding XML file that defines its relationships to other beads.

A Brief Explanation of the Sample Data

- WordNet 2.1, a psycholinguistic lexicon
- “Synsets” - synonyms grouped
- “Hypernyms”
- “Meronyms”

XML, XPath, XQuery in 60 (?) seconds

```
/bead

/bead/*/name

/bead[text/@lang="en"]

/bead/text[@lang="en"]/name

/bead/text[@lang="en"][name="D.H. Lawrence"]/name

/bead[text/@lang="en"][text/name="D.H. Lawrence"]/@id

distinct-values(/bead/text/name)

/bead/text[matches(name, "Lawrence")]/name

for $i in /bead order by $i/text/@lang return $i
```

O'REILLY

Indexing XML files with DB_File

```
use XML::LibXML;
use DB_File;

my $parser = new XML::LibXML;
my %btree;
tie %btree, 'DB_File', "$indexdir/words.index", O_RDWR|O_CREAT, 0666,
$DB_BTREE ...

opendir my $dir, "$datadir";
while (my $file = readdir($dir)) {
  next if ($file =~ /^\./);
  my $dom = $parser->parse_file("$datadir/$file");
  foreach my $node ($dom->findnodes("/bead/text/name")) {
    my ($textnode) = $node->findnodes("text()");
    my $lcname = lc($textnode->getData);
    if ($textnode) { $btree{$lcname} = $file; }
  }
}
untie %btree;
closedir($dir);
```

O'REILLY

Indexing XML files with DB_File

- File named “bead-text-name.db”
- Name/value pairs: “string value” => filenames
- A query script (or module) to return an array of filenames
- New indexes created for queries “off-line”
- Obviously has severe limitations, since you can’t really query a collection at all
- Proof of my DB bigotry

DB XML Background

- John Merrells
  - heavy LDAP involvement
  - lead engineer on Netscape Directory Server
  - designed and implemented DB XML on top of Berkeley DB
  - has now moved involvement outside of Sleepycat with Parthenon Computing
- Gareth Reakes
  - cofounder Parthenon
  - VP Apache organization
  - chair of Xerces project

DB XML Background

- Sleepycat
  - owns and continues investment in DB XML
  - sells licensing and support
- DB XML project
  - has an active open source community
  - has an active mailing list with rapid answers to questions
  - product blog with FAQs, build instructions, and tips

DB XML Features

- Embedded, meaning it's a library (via C++, Perl, Python, PHP, Java) operating on files, rather than a client/server
- XQuery is the interface, providing rapid development
- Query optimization, query pre-compilation
- Transactions, logging, replication, recovery, on-disk data encryption
- Better for content-oriented applications than a local relational DB; content is comprised of *documents and metadata*, after all
- Text means editor freedom
- Text protocols are here to stay; *why translate your data to and from XML?*
- No limit to data size; > 1M documents, > 100M documents
- Built on Berkeley DB

The Case for Embedding

- If you're working with XML anyway, it might as well be embedded
- Embedding tends to keep applications simple
- Embedding keeps control with the program
- Embedding reduces the process[ing] overhead
- Embedding means no network traffic for data
- Embedding makes DBAs mad

Installing and Using DB XML

Installation

- buildall.sh
- Package contains:
  - db-4.3.x
  - xerces-2.6
  - xquery-1.0.0
  - pathan
- End result: libs, bins, and docs in ‘install/’
- Tips:
  - Provide to buildall.sh -c (C compiler), -x (C++ compiler), and -m (make)
  - Build with provided packages
  - Build without Perl/Python support (do this manually)

Installation

- Tips (cont.):
  - On Mac OS X
    - Use Tiger and XCode 2.1 (with gcc 4.0)
    - Correct symbol errors with “otool” and “install_name_tool”
    - Repeat for Perl and Python “bundles”
  - On FreeBSD
    - Use 5.3 or later
    - gcc 4.0 used to have problems
    - LDFLAGS usually needs tweaking to have -liconv and -L/usr/local/lib

The Shell: Adding Data

```
$ dbxml
dbxml> createContainer beads.dbxml
Creating document storage container

dbxml> putDocument 1048.xml '
<?xml version="1.0"?>
<bead id="1048">
  <version date="TBD" editor="mbrian@glassbead.net" status="editing" seed="158 (43.
Authors)">0.0</version>
  <text lang="en">
    <name>D.H. Lawrence</name>
    <short>English writer of novels, short stories, poetry, and plays.</short>
  </text>
  <mappings>
    <wikipedia>D._H._Lawrence</wikipedia>
  </mappings>
...
</bead>
'
Document added, name = 1048.xml

dbxml>
```

O'REILLY

The Shell: Querying Data

```
dbxml> query '
collection("beads.dbxml")/bead/text/name/text()
'
1 objects returned for eager expression '
collection("beads.dbxml")/bead/text/name/text()
'
dbxml> print
D.H. Lawrence

dbxml> query ‘
collection(“beads.dbxml”)/bead[starts-with(version/@editor,
“mbrian”)][text/short=””]/@id/text()
‘
1 objects returned for eager expression '...’

dbxml> print
print
1048
```

O'REILLY

The Shell: Getting Help

```
dbxml> help

Command Summary
---------------
#                - Comment. Does nothing
abort            - Aborts the current transaction
addAlias         - Add an alias to the default container
addIndex         - Add an index to the default container
append           - Append to nodes specified in the query expression
commit           - Commits the current transaction, and starts a new
one
contextQuery     - Execute query expression using the last results
as the context item
...

dbxml> help addIndex
...
```

DB XML Library Objects

- XmlManager
  - High-level class for managing containers
  - Creation of document, query, and transaction objects
  - Preparing and running queries
  - Directories as environments (allowing logging and transactions)
  - Access flags and permissions
- Container
  - File that contains documents and indexes (metadata, too)
  - Various flags available (DB_CREATE/RDONLY) for access and default behaviors
  - Various container types; “Wholedoc”, “Node”
  - Can be created, deleted, and renamed

Populating a Database (C++)

```
#include "DbXml.hpp"
...

using namespace DbXml;
int main(void)
{
    std::string fileName = "./xml-src/1048.xml";
    std::string docName = "1048.xml";

    XmlManager myManager;

    myManager.setDefaultContainerType(XmlContainer::WholedocContainer);
    XmlContainer myContainer = myManager.openContainer("beads.bdbxml");
    XmlUpdateContext theContext = myManager.createUpdateContext();

    try {
        XmlInputStream *theStream =
            myManager.createLocalFileInputStream(fileName);

        myContainer.putDocument(docName,      // The document's name
                                theStream,    // The actual document.
                                theContext,   // The update context
                                              // (required).
                                0);           // Put flags.
    } catch (XmlException &e) {

    }
    return(0);
}
```

O'REILLY

Populating a Database (Perl)

```
use Sleepycat::DbXml 'simple' ;
my $datadir = “./xml-src/”;

eval
{
    my $mgr = new XmlManager;
    my $container = $mgr->openContainer("beads.dbxml");
    opendir my $dir, "$datadir";
    while (my $file = readdir($dir)) {
        next if ($file =~ /^\./);
        die "File not found: $datadir/$file" unless -e "$datadir/$file";
        open my $xmlfile, "$datadir/$file" or die "Couldn't open file: $!";
        my $id = $container->putDocument($file, (join '', <$xmlfile>));
    }
    closedir $dir;
};

if (my $e = catch std::exception) { warn $e->what(), "\n"; }
elsif ($@) { warn $@ }
```

O'REILLY

Adding to and Querying the Database (Python)

```
from bsddb.db import *
from dbxml import *

    ...
    mgr = XmlManager()
    uc = mgr.createUpdateContext()
    container = mgr.openContainer("beads.dbxml")
    container.putDocument(book_name, book_content, uc)

    ...
    qc = mgr.createQueryContext()
    results = mgr.query("collection('beads.dbxml')/bead/text/name", qc)
    results.reset()
    for value in results:
        document = value.asDocument()
        print document.getName(), "=", value.asString()
```

O'REILLY

Querying the Database (Java)

```
import java.io.*;
import com.sleepycat.dbxml.*;

class queryForDocumentValue
{
    ...
    public static void main(String args[])
        try {
            env = new myDbEnv(path2DbEnv);
            XmlManager theMgr = env.getManager();
            openedContainer = theMgr.openContainer(theContainer);
            XmlQueryContext context = theMgr.createQueryContext();
            getDetails(theMgr, "/beads/text[@lang='en'][name = 'apple'], context);
            getDetails(theMgr, "/beads[text/version/@editor = 'dbrian']",
                context);
        } catch ...
```

Querying the Database (Java) cont.

```
...
private static void getDetails(XmlManager mgr, String query, XmlQueryContext context)
      String fullQuery = "collection('beads.dbxml')" + query;
      XmlResults results = mgr.query(fullQuery, context, null);
      XmlValue value = results.next();
      while (value != null) {
          XmlDocument theDocument = value.asDocument();

          String name = getValue(mgr, theDocument, "/bead/text/name/text()", context);

          String short = getValue(mgr, theDocument, "/bead/text/short/text()",
                                  context);
          String version = getValue(mgr, theDocument,
                                  "/bead/version/text()", context);

          System.out.println("\t" + name + " : " + version + " (" + short + ")");

          value = results.next();
      }
```

O'REILLY

Setting and Querying Metadata (Shell)

```
dbxml> setMetaData 1048.xml '' editor string dbrian
MetaData item 'editor' added to document 1048.xml

dbxml> query '
collection("beads.dbxml")/*[dbxml:metadata("editor")="dbrian"]
'
1 objects returned for eager expression '
collection("beads.dbxml")/*[dbxml:metadata("editor")="dbrian"]
'
```

Setting Metadata (Perl)

```
my $mdURI = "http://dbxmlExamples/timestamp";
my $mdName = "timeStamp";

my $txn = $mgr->createTransaction();
my $xmlDoc = $mgr->createDocument();

$xmlDoc->setContent($xmlString);
$xmlDoc->setName($theName);

my $timeString = time ;

$xmlDoc->setMetaData($mdURI, $mdName, $timeString);

$container->putDocument($txn, $xmlDoc);
```

Setting and Querying Metadata (Shell)

```
dbxml> setMetaData 1048.xml '' timestamp decimal 1122150383
MetaData item 'timestamp' added to document 1048.xml

dbxml> query '
collection("beads.dbxml")/*[dbxml:metadata("timestamp") > 1122150380]'
1 objects returned for eager expression '
collection("beads.dbxml")/*[dbxml:metadata("timestamp") > 1122150380]'
```

Metadata Data Types

```
anyURI
base64Binary
boolean
date
dateTime
dayTimeDuration
decimal
double
duration
float
gDay
gMonth
gMonthDay
gYear
gYearMonth
hexBinary
NOTATION
QName
string
time
yearMonthDuration
untypedAtomic
```

Metadata Queries with Date Comparison

```
dbxml> setMetaData 1048.xml '' timestamp dateTime 2005-07-01T10:59:48
MetaData item 'timestamp' added to document 1048.xml

dbxml> query '
collection("beads.dbxml")/*[dbxml:metadata("timestamp") > xs:dateTime("2005-06-16T10:59:48")]
'
1 objects returned for eager expression '
collection("beads.dbxml")/*[dbxml:metadata("timestamp") > xs:dateTime("2005-06-16T10:59:48")]
'
```

Getting Metadata (Perl)

```
my $mdName = "timeStamp";

my $theDocument = $value->asDocument();

my $docName= $theDocument->getName() ;


my $metaValue = new XmlValue() ;
$theDocument->getMetaData( $mdURI, $mdName, $metaValue );
print "Document '$docName' stored on $metaValue\n";
```

Modifying (Replacing) Documents

```
doc.setContent("<a><b>random content</a></b>");

XmlUpdateContext uc = myManager.createUpdateContext();
myContainer.updateDocument(doc, uc);
```

Modifying Documents

```
XmlModify::addAppendStep()
XmlModify::addInsertAfterStep()
XmlModify::addInsertBefore()
XmlModify::addRemoveStep()
XmlModify::addRenameStep()
XmlModify::addUpdateStep()

(XmlQueryExpression, XmlObject type, std::string name, std::string
content)

XmlModify::execute()
```

Modifying Documents

```
XmlModify mod = myManager.createModify();
XmlQueryExpression select =
     myManager.prepare("/sampleDocument/node1/@attr1", qc);
mod.addRenameStep(select, "myAttribute");

std::string newChildContent = "<c1>some content</c1>";
select = myManager.prepare("/sampleDocument/node2", qc);

mod.addAppendStep(select,
                  XmlModify::Element,
                  "newChild",
                  newChildContent);

select = myManager.prepare("/sampleDocument/removeNode", qc);
mod.addRemoveStep(select);
```

Modifying Documents

```
<bead id="1048">
  <text lang="en">
    <name>D.H. Lawrence</name>
    ...
  </text>
  ...
</bead>

/bead/text
```

```
my $modify = $mgr->createModify();
$modify->addInsertAfterStep($mgr->prepare($tnx, "/bead/text", $context),
          XmlModify::Element, "text", "");
my $numOfModifications = $modify->execute($txn, $results, $context, $uc);

$modify->addAppendStep($mgr->prepare("/bead/text[2]", $context),
          XmlModify::Attribute, "lang", "jp");
```

O'REILLY

Constraining with Schemas

```
dbxml> createContainer validate.dbxml d validate
Creating document storage container, with validation

dbxml> putDocument 12231.xml '
<bead xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation=
        "http://conceptuary.com/lexicon/beads.xsd">
    <text @lang=”eh”>
        <name>Larry Wall</name>
    </text>
</bead>
' s

Document added, name = 12231.xml
```

Indices

```
dbxml> setVerbose 2 2

dbxml> query '
collection("beads.dbxml")/bead/text[@lang=”ch”]'

Query      - Starting eager query execution
Query      - beads.dbxml - U : [1111] 256 512 768 1024 257 513 769 1025 2 258
514 770 1026 3 259 515 771 1027 4 260 ...
Query      - Finished eager query execution, time taken = 245.167ms
0 objects returned for eager expression '
collection("beads.dbxml")/text[@lang="ch"]
'

dbxml> addIndex "" lang "node-attribute-equality-string"
Adding index type: node-attribute-equality-string to node: {}:lang

dbxml> query '
collection("beads.dbxml")/text[@lang="ch"]
'
Query      - Starting eager query execution
Query      - beads.dbxml - V(@lang,=,'ch') : NONE
Query      - Finished eager query execution, time taken = 0.976ms
```

Indices

- Created by node, per container
- Updated automatically with container changes
- Can require uniqueness
- Can index nodes by name or by edge
- Can index elements, attributes, or metadata
- Can key on presence, equality, or substrings
- Can enforce a syntax type
- Can be added with a string specification, or via enumerated arguments (programmatically)

Indices

```
[unique]-{path type}-{node type}-{key type}-{syntax type}

path type: node, edge

node type: element, attribute, metadata

key type: presence, equality, string

syntax type: none, string, double, time ...
```

“node-element-presence-none”

“unique-node-metadata-equality-string”

“edge-attribute-equality-float”

Adding Indices (Perl)

```
my $container = $mgr->openContainer("lexicon.dbxml");
$container->addIndex("","Word","node-element-equality-string");
```

O'REILLY

Query Plans

```
dbxml> qplan collection("beads.dbxml")/text/@lang

<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection">
      <OQPlan>P(node-attribute-equality-string,prefix,@lang)</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="text" nodeType="element"/>
    <Step axis="attribute" name="lang" nodeType="attribute"/>
  </Navigation>
</XQuery>
```

Index Notes

- More indices means means slower writes
- Doesn’t follow references to external entities or DTDs
- References are removed from character data
- Internal entity references are replaced with text
- Character data mixed with child data is concatenated
- Expands CDATA sections
- Building indexes can take a long time with many documents, especially substring indexes

Index Notes

```
471016K  xml-src/
196132K  lexicon.dbxml
204320K  lexicon.dbxml (Word index)
```

Multiple Data Sources

```
<bead id="1048">
  ...
  <mappings>
    <wikipedia>D._H._Lawrence</wikipedia>
    <dmoz/>
  </mappings>
  <pointers>
    <isa type="proper" db="wn2">author#n#1</isa>
  </pointers>
</bead>
```

```
collection("lexicon.dbxml")/Synset[Word = "author" and @pos = "n" and
    Word/@lexId = "0"]

collection("lexicon.dbxml")/Synset[Word = "author" and @pos = "n"]/Word
[text() = "author" and @lexId = "0"]
```

O'REILLY

Multiple Data Sources

```
dbxml> preload beads.dbxml

dbxml> preload lexicon.dbxml

dbxml> query '
for $i in collection("lexicon.dbxml")/Synset[Id="54132"]/Word/text()
for $j in collection("beads.dbxml")/bead/[text/@lang="en" and text/
    name=$i]
return $j/@id/text()
```

Reshaping Results

```
dbxml> query '<html><body>
  <ul>
    {
      for $bead in collection("beads.dbxml")/bead[version/@editor="dbrian"]
      return
        <li>
          <b>Bead ID: {$bead/@id/text()}</b><br/>
          {
            for $name in $bead/text/name
            return
                <p>{$name}</p>
          }
        </li>
  </ul>
</body></html>'
```

Sorting Results

```
dbxml> query '
  for $bead in
     (collection("beads.dbxml")/bead[pointers/isa = "11231"])
      order by xs:decimal($bead/@fam) descending
  return $bead/text[@lang="en"]/name[0]/text()
'
```

Managing Databases

- db_stat
- db_archive
- db_recover
- db_dump
- db_load

Conclusion & More Info

BDB XML is a powerful embedded library for indexing, querying, and managing collections of XML.

See Sleepycat's site for (good) documentation.

The 7th Annual

O'REILLY® OPEN SOURCE CONVENTION

AUGUST 1-5 2005