

Building PHP Applications with Berkeley DBXML

George Schlossnagle

2005-08-04

OSCON

What is DBXML?



— [Embeddable engine

— [Robust storage engine for XML documents

— [Fast, indexed searching

DB4

— [Berkeley DB4 is a high-performance, enterprise-grade embedded database engine

— [Support for:

— Transactions

— Locking

— Online Backups

— Support for XA and Replication

— Data Agnostic

DB4 Concepts: Db4

— [A Berkeley DB4 database.

— [Supports:

— key/value lookup

— cursor support

— multiple storage mechanism

— supports alternative indexes and joins via secondary databases

— platform-independent storage

DB4 Concepts: DbEnv

— [A Database Environment that encapsulates one or more DBs, and provides:

- Nested transactions
- Replication Support
- Write-ahead logging
- Multiple databases per file

DB4 Concepts: Db4Txn

- [Standard transaction semantics

- [Nested transactions for decomposing large transactions into smaller components.

DBXML Concepts

DBXML Concepts: XmlContainer

[A collection of XML documents

[One-to-one correspondence to a db4 database

[Indexes are bound to this container



DBXML Concepts: Indexes

— [Indexes are critical for query performance against collections.

— [Indexes are defined on a container.

— [Indexes can be defined for equality, presence or substring.

— [Indexes can be on node content, attributes, or metadata.

DBXML Concepts: XmlManager

[A factory object that manages XmlContainers.

[Optionally bound to a db4 environment context



DBXML Concepts: XmlDocument

— [An XML Document (whoda thunk?)

— [Lives in a XmlContainer

— [Documents are uniquely named within their container.

DBXML: Under the Hood

- [Xerces (XML parsing and handling)

- [Pathan (XPath parsing and evaluation, built on Xerces)

- [XQuery (XQuery implementation built on Pathan)

- [Limitations for PHP:

- PHP's native XML handling is libxml2 based

- Passing to other XML extensions requires conversion

DBXML Concepts: XQuery

- [A W3C Draft Recommendation.

- [XQuery is an XML query language built as an extension on XPath 2.0.

- [Queries can span:

- A document

- A collection of documents in an XmlContainer

- Multiple collections of documents.

DBXML Concepts: Path Expressions (I)

/	A documents root
/Foo	Foo nodes under the root node
/Foo/Bar	Bar nodes that are a child to Foo
/* /Bar	Bar nodes under any node under the root node
//*	Shorthand for all nodes in the document
.	The current node
..	The current node's parent
/Foo/@baz	The baz attribute of Foo
//Foo	All Foo nodes, anywhere
//myns:Foo	All Foo nodes in the namespace aliased to 'myns'

DBXML Concepts: Path Expressions (II)

```
//Foo[@bar = 'baz']  
/Foo[@number > 100 and @number < 200]  
//Foo[../@bar = 'baz']  
//Foo[count(./Bar) > 1]
```

```
doc("my.dbxml/doc.xml")/Foo  
doc("file:/var/tmp/doc.xml")/Foo  
collection("my.dbxml")/Foo
```

FLWOR Power!

[XQuery Query Structure

— FOR

— LET

— WHERE

— ORDER BY

— RETURN



DBXML Concepts: FLOWR in Action

```
<results>
{
let $con := collection("books.dbxml")
for $book in $con/Item
let $t := $book/ItemAttributes/Title
let $p := $book/ItemAttributes/ListPrice/Amount
let $prettyprice := $book/ItemAttributes/ListPrice/FormattedPrice
where count($book/ItemAttributes/Author) > 1
order by $p ascending
return
  <result>
    <title>{$t}</title>
    <price>{$prettyprice}</price>
  </result>
}
</results>
```

XQuery UDFs

```
declare function local:summary($books as element(Item)*
  as element(results)*
{
  for $d in distinct-values($items/Publisher)
  let $e := $books[Publisher = $d]
  return
    <results>
      <publisher>{$d}</publisher>
      <bookcnt> {count($e)} </bookcnt>
      <pagecount> {sum($e/NumberOfPages)} </pagecount>
    </results>
};
```

```
local:summary(collection("books.dbxml")//Item/[Author = "Herman Melville"])
```

DBXML Concepts: Index Types

Uniqueness = (unique)?

Path Types = (node | edge)?

Node Types = (element | attribute | metadata)

Key = (equality | presence | substring)

Type = (string | boolean | date | ...)

Examples:

node-element-equality-string

node-attribute-equality-floats

node-element-substring-string

metadata-equality-string

DBXML in Practice

Where To Get It

Go to: <http://www.sleepycat.com>

Berkeley DBXML Package contains all you need

PHP Components:

[db-4.3.x/mod_db4](#) - An Apache (1.3) module for coordinating recovery and providing environment caching.

[db-4.3.x/php_db4](#) - A PHP wrapper for DB4

[dbxml/src/php](#) - A PHP wrapper for DBXML

Read the READMEs!

DBXML Defined Classes

```
class XmlManager {}  
class XmlContainer {}  
class XmlData {}  
class XmlDocument {}  
class XmlIndexSpecification {}  
class XmlModify {}  
class XmlQueryContext {}  
class XmlQueryExpression {}  
class XmlResults {}  
class XmlValue {}  
class XmlUpdateContext {}  
class XmlTransaction {}  
class XmlInputStream {}  
class XmlStatistics {}
```

Creating a Container and Adding Some Data

```
$book_name = 'book1';  
$book_content = '<book><title>Knowledge Discovery in Databases.</title></book>';  
  
$mgr = new XmlManager(null);  
if(file_exists("test.dbxml")) {  
    $mgr->openContainer("test.dbxml");  
} else {  
    $con = $mgr->createContainer("test.dbxml");  
}  
$con->putDocument($book_name, $book_content);
```

Extracting Data From a Container

```
$mgr = new XmlManager(null);  
$con = $mgr->openContainer("test.dbxml");  
$results = $mgr->query("collection('test.dbxml')/book");
```

```
$results->reset();  
while($results->hasNext())  
{  
    $val = $results->next();  
    $doc = $val->asDocument();  
    print $doc->getName()." = ".$val->asString()."\n";  
}
```


Querying with Namespaces

```
$mgr = new XmlManager(null);
$con = $mgr->openContainer("test.dbxml");
$qc = $mgr->createQueryContext();
$qc->setNamespace("books2", "http://foo.bar.com/books.dtd");

$query = $mgr->query("collection('test.dbxml')/*[books2:title='Knowledge Discovery in Databases.']", $qc);
$results->reset();
while($results->hasNext())
{
    $val = $results->next();
    $doc = $val->asDocument();
    print $doc->getName()." = ".$val->asString()."\n";
}
```

Adding Indexes to a Container

```
$mgr = new XmlManager(null);  
$ns = "";  
$node = 'title';  
$con = $mgr->createContainer("test.dbxml");  
$con->addIndex($ns, $node, "node-element-equality-string");
```

Reflecting Indexes

```
$mgr = new XmlManager();  
$con = $mgr->openContainer("test.dbxml");  
$con->addIndex("", "title", "node-element-equality-string, edge-element-equality-string");  
$indexes = $con->getIndexSpecification();  
foreach($indexes->getIndexes() as $a) {  
    var_dump($a);  
}
```

Using Prepared Queries

```
$qc = $mgr->createQueryContext();
$qc->setVariableValue("title", "");
$query = $mgr->prepare("collection('test.dbxml')//*[title=\$title]", $qc);
foreach($titles as $title) {
    $qc->setVariableValue("title", "Knowledge Discovery in Databases.");
    $results = $query->execute($qc);
    // ...
}
```

Query Local Documents

```
$mgr = new XmlManager(null);  
$results = $mgr->query("doc('file:./14.xml')/book");
```

```
$results->reset();  
while($results->hasNext())  
{  
    $val = $results->next();  
    $doc = $val->asDocument();  
    print $doc->getName()." = ".$val->asString()."\n";  
}
```

Performing DML on Documents (I)

```
$mgr = new XmlManager();  
$qc = $mgr->createQueryContext();  
$uc = $mgr->createUpdateContext();  
$mod = $mgr->createModify();  
  
$select = $mgr->prepare("/book/title", $qc);  
$mod->addAppendStep($select, XmlModify_Element, "foo", "bar");  
  
$retdoc = $mgr->query("collection('test.dbxml')/book");  
$mod->execute($retdoc, $qc, $uc);
```

Performing DML on Documents (II)

```
$mgr = new XmlManager();  
$qc = $mgr->createQueryContext();  
$uc = $mgr->createUpdateContext();  
$mod = $mgr->createModify();  
  
$select = $mgr->prepare("/book/title", $qc);  
$mod->addUpdateStep($select, "bar");  
$retdoc = $mgr->query("collection('test.dbxml')/book");  
$mod->execute($retdoc, $qc, $uc);
```

Managing Containers

```
$mgr = new XmlManager();  
$con = $mgr->createContainer("test1.dbxml");  
$mgr->renameContainer("test1.dbxml", "test2.dbxml");
```

```
$con = $mgr->openContainer("test2.dbxml");  
$mgr->removeContainer("test2.dbxml");
```


Using DbEnvs

```
$env = new Db4Env();  
$env->open();  
$mgr = new XmlManager($env);  
$txn = $mgr->createTransaction();  
$con = $mgr->createContainer($txn, "test.dbxml", DBXML_TRANSACTIONAL);  
$doc = $mgr->createDocument();  
$doc->setContent($book_content);  
$doc->setName($book_name);  
$con->putDocument($txn, $doc);  
$txn->commit();
```

Transactions Spanning Multiple Containers

```
$env = new Db4Env();  
$env->open();  
$mgr = new XmlManager($env);  
$txn = $mgr->createTransaction();  
$con1 = $mgr->openContainer($txn, "test.dbxml");  
$con2 = $mgr->openCContainer($txn, "test2.dbxml");  
$con1->putDocument($txn, $book_name, $book_content);  
$con2->putDocument($txn, $book_name, $book_content);  
$txn->commit();
```

Join Collections in a Query

```
$query = '<html><body>
<ul>
{
for $component in collection("components.dbxml")/component
return
<li>
<b>Component number: {$component/@number/text()}</b><br/>
{
for $part-ref in $component/uses-part
return
for $part in collection("parts.dbxml")/part[@number = $part-ref cast as xs:decimal]
return
<p>{$part/description/text()}</p>
}
}
</li>
}
</ul>
</body></html>';
```

Future Bits

— [Better PHP5 Support (iterators)

— [High-Level wrappers (hide the gory details)

— [DBXML 2.2 (Q4)

— [DB-4.4 (Q4?)

— [Apache 2 Support for mod_db4?

Thanks!
Questions?