Use a Native XML Database for Your XML Data

You already know it’s time to switch.

Gregory Burd • Product Manager • gburd@sleepycat.com
Agenda

• Quick Technical Overview
  • Features • API • Performance
• Clear Up Some Misconceptions
• How to Justify Choosing an XML Database
• Use Cases, Commercial and Open Source
• XML Database Markets
• Product Review
What is Berkeley DB XML?

A native XML database (NXD) with XQuery support.

- XML document management
  - storage
  - modification
  - and retrieval
- Scalable
  - millions of documents
  - terabytes of data
- Parse, query, plan, optimize
- ACID Transactions, even XA
- and much more
Berkeley DB XML • Features

a library, linked into your application or server

• reduced overhead
• simple installation
• easy administration
• Apache module, PHP API
• could easily become a server, but what standard? XDBC? XML:DB? SOAP/WSDL? REST?
Berkeley DB XML • Features

inherits everything from DB

• transactions, concurrency, replication, encryption
• scalable, fast, and resource efficient
• portable, field tested, and proven database technology
Berkeley DB XML • Features

documents

• optional validation
• named UTF-8 encoded documents
• associated, searchable, and indexed meta-data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE page-specification ...>

<inventory>
  <item>
    ...
  </item>
</inventory>
```
Berkeley DB XML • Features

containers
• manage groups of documents
• XML stored as nodes or whole documents
• query across multiple containers
Berkeley DB XML • Features

queries
• XQuery and XPath
• eager or lazy evaluation
• pre-compiled queries with variables

query ' for $collection in collection("books.dbxml")/collection return <link class="collectiontitle"
  title="{$collection/@title/ .}" idref="{$collection/@id/ .}" />
',
print
Berkeley DB XML • Features

indexes
• an index is targeted by path, node, key, and syntax type
• human-readable query plans
• add or remove indexes at anytime
• performance at insert(-), update(-) and access(++++)
Berkeley DB XML • Features

**dbxml command line shell**
- experiment and learn
- administer
- integrate into scripts

```
$ dbxml -s test.xquery
```

```
dbxml>
```

```
$ dbxml -s test.xquery
```
Berkeley DB XML • Features

web site scripting support
- quick prototypes
- use any combination of scripting and compiled concurrently
- LAMP using XML rather than rows and columns
// Query for sets of documents within
// namespaces and use variables.

std::cout << "it’s time for some code, let’s begin..."
// First open and configure the Berkeley DB environment

DbEnv env(0);
env.set_cachesize(0, 64 * 1024 * 1024, 1); // 64MB cache
env.open(path2DbEnv.c_str(),
        DB_INIT_MPOOL|DB_CREATE|DB_INIT_LOCK|DB_INIT_LOG|
        DB_INIT_TXN, 0);

// And now setup a Berkeley DB XML manager

XmlManager mgr(&env);
Berkeley DB XML • API

// Open an XML document container inside that db environment

XmlTransaction txn = mgr.createTransaction();

std::string theContainer = "namespaceExampleData.dbxml";

XmlContainer container = mgr.openContainer(txn, theContainer);

txn.commit();
// Create a context and declare the namespaces

XmlQueryContext context = mgr.createQueryContext();

custom.setNamespace("fruits",
    "http://groceryItem.dbxml/fruits");
context.setNamespace("vegetables",
    "http://groceryItem.dbxml/vegetables");
context.setNamespace("desserts",
    "http://groceryItem.dbxml/desserts");
// Perform each of the queries

// Find all the Vendor documents in the database.
// Vendor documents do not use namespaces, so this
// query returns documents.

// ‘doContextQuery()’ is application code, not BDB XML code
// We’ll look inside the method in a second...

doContextQuery(mgr, container.getName(), "/vendor", context);
// Find the product document for "Lemon Grass" using
// the namespace prefix 'fruits'.

doContextQuery(mgr, container.getName(),
    "/fruits:item/product[.="Lemon Grass"]", context);
// Find all the vegetables

doContextQuery(mgr, container.getName(), "/vegetables:item", context);
// Set a variable for use in the next query
context.setVariableValue((std::string)"aDessert", (std::string)"Blueberry Muffin");

// Find the dessert called Blueberry Muffin
doContextQuery(mgr, container.getName(), "/desserts:item/product[.=$aDessert]", context);
// Here is the implementation of the method we've been calling

doContextQuery(XmlManager &mgr,
    const std::string &cname,
    const std::string &query,
    XmlQueryContext &context ) {

    try {

        ...

    }
// Build up the XQuery, then execute it.

std::string fullQuery = "collection('" + cname + "')" + query;

std::cout << "Exercising query '" << fullQuery << "' " << std::endl;

std::cout << "Return to continue: ";
getc(stdin);

XmlResults results(mgr.query(fullQuery, context));
// Here is the loop that gets each match and displays it

XmlValue value;
while(results.next(value))
{
    // Obtain the value as a string and
    // print it to the console
    std::cout << value.asString() << std::endl;
}
// Now output the number of results we found for the query

std::cout << results.size() << std::endl;
std::cout << " objects returned for expression " << std::endl;
std::cout << fullQuery << "\n" << std::endl;
try {
  ...
}

//Catches XmlException
catch(std::exception &e) {
  std::cerr << "Query " << fullQuery << " failed\n";
  std::cerr << e.what() << "\n";
  exit(-1);
}

} // end of the method doContextQuery()
exit(0);

// End of the code, hope you liked it.

This example ships with Berkeley DB XML:

dbxml/examples/cxx/gettingStarted/queryWithContext.cpp
Quote after evaluating Berkeley DB XML for a production medical records system.

“Frankly we’re impressed. We loaded up well over 100,000 documents of moderate size, set up some indices, and then executed some complex XQuery expressions. Most results were around one-tenth of a second.”
We use XBench to performance test

The benchmark statistics are calculated using the XBench benchmarking toolkit, using data sets of size 20k, 1Mb and 10Mb. The time recorded is for the sum of the three queries, same query for each container size.

The data sets that XBench produces comes in 4 flavors. It creates text centric and data centric data sets, and single document and multiple document data sets. We then run these data sets against both document and node storage, using a set of indexes hand picked to work well with the given data and queries.

Test Machine:
Intel P4 3.0GHz, 1GB RAM, Red Hat 9 Linux
Berkeley DB XML • Performance

TC-MD 16 - Return the titles of articles which contain a certain word ("hockey").

```xml
for $a in collection() / article
where some $p in $a//p satisfies contains($p, "hockey")
return
$a/prolog/title
```

<table>
<thead>
<tr>
<th>Category</th>
<th>Percent of Total Time</th>
<th>Time Taken/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Storage</td>
<td>39.55 %</td>
<td>0.718638</td>
</tr>
<tr>
<td>Node Storage</td>
<td>5.66 %</td>
<td>0.102871</td>
</tr>
<tr>
<td>Node+ Storage</td>
<td>9.65 %</td>
<td>0.175285</td>
</tr>
</tbody>
</table>
Return the names of publishers who publish books between a period of time (from 1990-01-01 to 1991-01-01) but do not have FAX number.
List the orders (order id, order date and ship type), with total amount larger than a certain number (11000.0), ordered alphabetically by ship type.

```
for $a in collection()//order
where $a/total > xs:decimal("11000.0")
order by $a/ship_type
return
  <Output>
    {${a/@id}}
    {${a/order_date}}
    {${a/ship_type}}
  </Output>
```
#1 “XML is a file format. Repeat after me. A text file format.” (source Slashdot March/2005)

- XML (Extensible Markup Language) is a W3C initiative that allows information and services to be encoded with meaningful structure and semantics that computers and humans can understand.

- XQuery is the query language for XML. … Just as SQL is a query language that queries relational tables to create new relational tables, XQuery queries XML documents to create new XML documents.
#2 “The thing the XML databases are nice for is if folks can't really lock down the schema.”
(source Slashdot March/2005)

- DTD, XSchema and document validation aren’t good enough for you? With Berkeley DB XML you can validate on each document differently, or not at all and still query everything at once.

- Sometimes it can be advantageous to not require a schema.
Berkeley DB XML • Myths

#3 “Lack of mature database features.” (source Slashdot March/2005)

• Most XML solutions are file based, so it is common to confuse implementation with appropriateness.

"Ignorance of the decades of scientific research and engineering experience in the field of relational database management systems, relational algebra, set theory and predicate calculus; lack of real atomicity of transactions, lack of guaranteed consistency of data, lack of isolated operations, lack of real durability in the ACID sense, and in short, the lack of relational model; scalability, portability, SQL standard, access to your data after two years and after twenty years; to name just a few."

(source Slashdot March/2005)
http://developers.slashdot.org/article.pl?sid=05/03/10/2043202&tid=221
Berkeley DB XML • Justify

- Do you have thousands of XML files?
- Is your XML data larger than 200MB?
- Are you trying to build a hierarchy into tables?
- Could your data change over time?
- Have you spent more that $100 on books explaining SQLXML?
- Are you waiting for the next release of some RDBMS to fix your performance problems?
Searching for a nice place to stay?

If you’re searching Starwood, you’re using Berkeley DB XML.

Starwood Hotels
The Juniper NetScreen Security Manager has to determine what is a threat, the threat reports live in Berkeley DB XML.
Berkeley DB XML • Use Case

Mass amounts of data for financial models models in XiMoL

XML objects reside in Berkeley DB XML.
Berkeley DB XML • Use Case

Publish and subscribe integration with XML messages,

where the data moves through Berkeley DB XML.
Your prescription drug history may be stored as XML...

Ask a Pharmacist

in Berkeley DB XML.

Berkeley Medical
Berkeley DB XML • Use Case

Zero programming next generation XML based content management...

and the content resides in Berkeley DB XML.

Feedstream
Berkeley DB XML • Open Source

An open source XML content publication system written in Python,

Syncato

built on top of Berkeley DB XML.

Syncato

A content management system for the little things
Berkeley DB XML • Markets

- Research
- Notification
- Integration
- Bioinformatics/Genomics
- Security
- Content Management
- The ‘X’ in LAMP
Berkeley DB XML • Review

By Rick Grehan

Highest rating ever given ->

(source InfoWorld May/2005)
http://www.infoworld.com/article/05/05/23/21TCxmlmdb_2.html
Berkeley DB XML • Release

Now available, Berkeley DB XML 2.2

- Node level indexes
- Optimized query planning and intelligent indexing
- Faster path expression evaluation and predicate evaluation
- Optimized resource utilization
- Efficient type casting
- New index lookup functions
- Supports the April 2005 drafts of XQuery 1.0 and XPath 2.0

And 500% or better improvement in average query speed!
Berkeley DB XML

Choose the right database for your XML data.

Gregory Burd • Product Manager • gburd@sleepycat.com