

Comparing XML database approaches

What are the similarities and differences between pureXML and native XML databases?

Skill Level: Intermediate

[Adriaan de Jonge](#)

Software Professional
Freelance

16 Dec 2008

The increasing use of XML asks for systems that store semi-structured data without forcing it into inadequate data structures. These requirements are met by both native XML databases and relational databases with integrated XML support. The question is: Why should you prefer a native XML database over an XML-aware classic database or the other way around? This article compares the approaches of a number of varying solutions, including eXist, Mark Logic, and IBM® DB2® Express-C. The differences are translated into practical advantages and disadvantages.

In the early years of XML databases, the difference between so-called native XML databases (NXDs) and XML-aware regular relational database management systems (RDBMSs) was large. NXDs were optimized for storing documents described in XML. Older RDBMSs only added some syntactic sugar around regular binary large objects (BLOBs) that happened to contain XML.

Frequently used acronyms

- API: Application programming interface
- HTTP: Hypertext Transfer Protocol
- IT: Information technology
- XML: Extensible Markup Language

Nowadays, NXDs are still NXDs but more advanced. At the same time, the vendors of mature RDBMSs have had time to catch up when it comes to the storage of XML documents. XML fragments are no longer stored in BLOBs. Rather, they are stored in tree-like structures, optimized to contain tree-like documents, the basic nature of a typical XML document.

In the time between early implementations and today's mature solutions, some fundamental developments have added to the standardization of query languages on XML documents—most notably, XQuery 1.0, with support for XML Path Language (XPath) 2.0. The concept of XQuery was under development for years; the end result has similarities with the early versions but is more mature. Like Structured Query Language (SQL), XQuery promotes vendor independence and reuse.

Why do you need XML databases?

Regular databases can store both highly structured data and unstructured documents. Both require data structures that do not change often. However, a weakness of relational databases is in storing documents with a semi-structured nature. Unlike structured data, these documents can have many degrees of freedom in the order of document elements and the way those elements are nested within each other. Unlike unstructured documents, the individual elements can be classified using descriptive labels. These elements tend to be fine-grained.

Can you store semi-structured data in relational structures? Of course you can, but you are likely to end up with a specific data structure that changes frequently, a generalized data structure that loses descriptiveness of the labels or with an abstract model such as those that content management systems use, mixing data with things that should actually be metadata.

XML formats, on the other hand, are good at describing data of a semi-structured nature. In addition, you can easily maintain the data model. Adding new element names does not change the data structure—it always remains a tree structure. It only involves a change in the XML schema describing the way that element names are used and related within the tree structure.

For documents such as job resumes, product descriptions, and customer orders, XML documents might be a suitable format. At the same time, XML is also capable of describing structured and unstructured data.

Then, do you still need relational databases?

When you create a new software solution, the answer might be "No!". If you have a solution that lets you store semi-structured data, you can also use it to store

structured and unstructured data. Having all your data in a single storage solution with the possibility to interconnect and create queries that cover everything in one pass is a lot easier than integrating data from multiple storage sources.

To be reasonable, for an honest "no" to this question, most of your data should be document-like storage of semi-structured nature. If, however, the majority of your data fits better in a highly structured entity-relationship model and is less document-like and more intertwined, then choosing an NXD might not help your situation.

So how do you determine the nature of your data? And what do you do if the amount of structured, semi-structured, and unstructured data is roughly equal? In case of uncertainty, the good news is that classic databases are better suited to contain XML documents or fragments of XML documents these days. The ways to access these XML fragments might differ between databases. One thing the implementations have in common, though, is that they use constructions from XQuery 1.0.

Solutions

Some products on the market implement an XML database one way or another, including Xindice, Tamino, X-Hive, Oracle, and Microsoft® SQL Server. However, I don't discuss these products in this article. A full and lengthy product comparison is not feasible, and it would not be credible given the fact that this article is published on the site of a vendor of an XML database solution. Having an independent author does not sufficiently solve questions of neutrality.

What I can do is discuss IBM DB2 Express-C with pureXML features and compare it to the approach of classic NXDs. For this comparison, I selected an open source project called eXist-DB. Both eXist and DB2 Express-C are available for free and offer a lot of user-friendly functionality.

If you want to store very large amounts of data, I advise you to purchase the commercial version of DB2. At first, DB2 might sound like heavyweight stuff, but you can install DB2 Express-C on a desktop or laptop to evaluate pureXML capabilities with relative ease. Although there is no commercial version of eXist, Mark Logic is a good replacement candidate for eXist when performance and scalability requirements go beyond capabilities of eXist. You can evaluate Mark Logic using either a 30-day trial license or a community edition that is limited to 100MB storage.

I can imagine the need for product comparisons between similar products such as DB2 and Oracle. You can find some older debates online about these two products. A much more fundamental discussion is how the pureXML features in DB2 Express-C compare to eXist-DB. Or, similarly, how DB2 compares to Mark Logic.

Native XML database

Like most product category names, the term *native XML database* is a bit misleading. It raises questions such as: What is a database? What is native XML? Is DB2 an NXD?

According to Wikipedia, "a computer database is a structured collection of records or data that is stored in a computer system." *Native XML* is the use of XML-related technologies without mixing them with non-XML technology. This means the ability to use XQuery and XPath without traces of SQL. IBM anticipates the question about whether DB2 is native XML by calling its features pureXML. I'll get back to that definition in the [next section](#).

When comparing NXDs to XML-aware RDBMSs, I think a typical NXD can also be classified as a document repository. The term *document repository*, however, is taken by products such as Alfresco and Magnolia. These products are layered on top of existing databases, and they lack the infrastructure offered by eXist and Mark Logic—most notably the capability to execute XQueries in an efficient way. They focus on higher-level functionality such as workflow and user-friendly interfaces. NXDs leave such things up to the users of their products. They only offer lower-level document repository APIs, such as Web-based Distributed Authoring and Versioning (WebDAV) or custom RESTful connectors.

So a typical NXD stores XML documents in an efficient way. It provides XQuery technology and a thin layer of document repository functionality.

NXD tend to be *resource oriented*, essentially meaning the individual pieces of content stored in the repository can be identified using Uniform Resource Identifiers (URIs). Using HTTP or WebDAV, the same URIs allow access to the data, which makes connectivity a non-issue.

Treating data as single resources also has its weaknesses. The separation of documents makes it more difficult to create relationships between data that is divided over multiple documents. If one document contains the authoritative data that other documents refer to, it is more difficult to maintain referential integrity. Vendors of larger XML databases offer options to put constraints over data in multiple documents. However, this is not as standardized as other XML technologies.

pureXML

IBM chose to avoid the term native XML database but still wanted to convey the pure XML nature of their solution. DB2 Express-C is atypical for an NXD, but it does share some of the characteristics. From one perspective, DB2 seems like a regular

RDBMS with XML-aware columns. If you want to use a relational database without being bothered with XML technologies, you can just ignore the XML-aware columns. However, if you do want XML features, don't mistake the subtleties of the XML-aware columns for a lack of XML capabilities.

The name pureXML is legitimized in two ways:

- XML data is stored in a native tree format, separate from the relational data.
- You are allowed to access all data—both relational and XML—through a single XML interface.

XQuery 1.0 is not limited to querying XML documents. IBM provides XQuery functions that allow you to query relational data and mix and match the results with the XML from the XML-aware columns. The result of the combined data can be returned in XML format.

DB2 is pure database infrastructure. It does not offer the thin layer of document repository functionality that many NXDs do. That does not mean that you cannot use DB2 to implement a document repository—either thin or thick. At the moment, most document repositories such as Alfresco are implemented on top of non-XML RDBMSs rather than NXDs. Using DB2 as basis for these repositories means at least the same result they offer now, but, more likely, with the added benefit of more flexible data models using XML storage capabilities.

It might be a while before products like Alfresco pick up features such as pureXML. The reason is that such products do not want to be tied to a single database product. Despite XQuery 1.0, the way the XQuery language is used in database products still lacks standardization. For these products, the safer bet is to delay use of the new capabilities of XML databases, wait for more uniform usage of XQuery and connection protocols, and for now, limit themselves to standards with more uniform implementations like WebDAV or relational database connectors.

Where can you use the pureXML capabilities of DB2?

The answer is quite simple. The great majority of IT solutions in the world have invested heavily in relational databases. Even when relational databases lacked the flexibility required for a solution, they were still more mature and better suited for heavy demands than the alternatives.

Companies that want to take the next step and adopt XML technologies are unlikely to throw away the results of many years of development and replace it with an NXD. And, though they might use an NXD in addition to their relational database, this can create new challenges in the integration of data divided over multiple sources. Think

of transaction handling and referential integrity. Although not impossible to solve, a single integrated solution that takes care of these concerns has many advantages.

The combination of relational data and XML data in pureXML helps provide a smooth transition or, more likely, partial transition to XML technology.

Conclusion

Although XML documents can describe structured data, an NXD containing many XML documents might not be the best solution to describe relationships between structured data divided over multiple documents and maintain referential integrity. If it is already possible to manage such relationships (perhaps using XPointer), the usage is not standardized.

The best advice is to use the right tool for the right job. However, coexistence of an NXD with an RDBMS creates integration challenges. The pureXML features of DB2 combine the best of both worlds and allow the storage of both structured and semi-structured data. For the storage of unstructured data, NXDs and RDBMSs are equally suited.

If you are actually looking for a document repository rather than a database, you should ask a different question. Do your requirements focus on the front-end capabilities of the document repository or do they focus on the back-end? If you need a lot of end-user functionality such as an administration interface, workflow, and visualization, then many products on the market, such as Alfresco, offer this kind of support. If you are interested in powerful query mechanisms, such as XQuery and full text search, and efficient storage of semi-structured documents, then NXDs such as eXist or Mark Logic might be of interest to you.

The world of database solutions is dominated by a small number of large vendors. IBM is one of them. NXDs are still a niche market. The questions are whether these smaller players can survive the competition, whether they will be bought by larger corporations (as X-Hive was bought by EMC), or whether they will lose market share to hybrid solutions. In a clear trend, RDBMS vendors, unwilling to give up market share to NXDs, offer their own fully fledged but well-integrated XML solutions to answer the needs of their customers.

Resources

Learn

- [XQuery 1.0](#): Learn about the powerful XML query standard used in every self-respecting XML database.
- [XML technical library](#): See the developerWorks XML Zone for a wide range of technical articles and tips, tutorials, standards, and IBM Redbooks.
- [developerWorks technical events and webcasts](#): Stay current with technology in these sessions.
- [developerWorks podcasts](#): Listen to interesting interviews and discussions for software developers.
- [Technology bookstore](#): Browse for books on these and other technical topics.

Get products and technologies

- [DB2 pureXML](#): Find out more about the capabilities of the full DB2 platform including pureXML capabilities.
- [DB2 Express-C with pureXML](#): Evaluate pureXML technology using the free community edition.
- [eXist-DB](#): Download this open source NXD that is gradually evolving into an NXD competitor for MySQL.
- [Mark Logic](#): When your performance and scalability requirements exceed eXist's capabilities, Mark Logic is a good replacement candidate.

Discuss

- [XML zone discussion forums](#): Participate in any of several XML-related discussions.
- [developerWorks blogs](#): Check out these blogs and get involved in the [developerWorks community](#).

About the author

Adriaan de Jonge

Adriaan de Jonge is a software professional currently working for the Dutch government, juggling a few projects in several roles. Adriaan has written XML-related articles for IBM developerWorks and Amazon. You can reach Adriaan at adriaandejonge@gmail.com.

Trademarks

IBM, the IBM logo, ibm.com, DB2, developerWorks, Lotus, Rational, Tivoli, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. See the current list of [IBM trademarks](#).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.