Spring, 2007

LTER
Long Term Ecological Research

01001100 01010100 01000101 01010010
LTER DataBits
Information Management Newsletter of
The Long Term Ecological Research Network
01001100 01010100 01000101 01010010

Database Storage Model Considerations: XML and Relational Database Approaches

- James Conners (PAL & CCE)

When writing a computer software application, be it for standalone purposes, web applications, etc. there is almost always the issue of persistent data storage models to consider. This may involve a range of factors such as account settings, application configuration parameters, and scientific dataset characteristics. The question of whether to store this data in XML files or a relational database often arises. The decision requires an assessment of the requirements for each application, taking into consideration the programmatic aspects of development, maintenance, documentation, as well as the inherent characteristics of the data. Searching through developer forums will certainly bring up a variety of discussions on the more technical aspects of the comparison of these approaches by technologists with detailed experience; this article focuses simply on a few higher-level considerations concerning portability, data access, and inherent data characteristics.

In consideration of portability, the desired degree of this attribute will often dictate which is the best storage technology to implement. The greater the focus is on an application that can be used by a variety of users and on different platforms, the more data storage options tend to point towards portable documents. XML documents, simply being text files, are at the very high end of portability. Any technology that can parse text files can implement a storage model that relies on XML. Implementing relational database technology requires, on the other hand, an installation of a particular DBMS (Database management system) such as MySQL, Oracle, etc. on either local machines or a dedicated server that can be accessed by remote instances of the program on client machines. Of course, the last DBMS approach removes a degree of portability with the requirement for connectivity to a remote system. Another aspect of portability deals with the transport of data. XML is perhaps best known for its advantages in transport, that is, in the sharing of data. The XML standards established by the WWWC (World Wide Web Consortium: http://www.w3.org/TR/2000/REC-xml-20001006#dt-doctype) provide for a means of validating XML documents against established DTD's (Document-type Declarations). The standardized schema definition language XML Schema (W3C) builds on DTD's and provides for a more robust document definition. Either method provides a mechanism for maintaining data integrity during transportation.

Data access is another issue to consider when deciding between XML and a relational database. RDB solutions are known for their advantages in terms of data retrieval. If the stored data must be queried by varying parameters, and the dataset is large, the relational database system has advantages. Relational theory is a well-established field of study and as a result the current DBMS's are highly optimized for searching through large amounts of data based on parameter specifications. The majority of modern, higher-level programming languages provide interfaces to the well-established DBMS's thus enabling applications access to the efficiency of SQL and relational structures. With the standardization of the query language like XML Query (http://www.w3.org/XML/Query/) the capability of searching the relational structure of data stored in XML format is being improved. Yet, as the size of the files grows, XML’s serial hierarchical structure and text-based format means that searching through XML documents can significantly tax the performance of an application.

Another deciding factor in choosing between XML and a relational database is whether the data structure at hand is consistent or not. Relational databases are a natural fit for tabular data with a regular, well-defined structure. When the structural properties of the data vary, however, XML might be a better fit. "Because XML syntax records only what is present, not everything that might be present" it can be better suited to records that can vary drastically with regard to missing values or even completely different field sets 1. Ronald Bourret gives an example during a discussion on storing XML files in which he points to medical records where different patients may have completely disparate records with mutually exclusive properties. Accommodating all the possible parameters in a relational database would lead to sparse tables; using XML’s explicit structure of available data appears to be a more suitable solution.

In exploring solutions, it is possible to consider arrangements that combine the best features of both XML and relational databases. A common perception of the two technologies is that XML is best used as a document-sharing format, that is, a transport mechanism. On the other hand, a relational database is recognized for its ease of implementing the many aspects of data maintenance and search optimization. Thus, many DBMS’s either provide native capabilities for manipulating XML-formatted documents or at minimum provide plug-in functionality for this purpose. The combination of using a database for data storage together with XML documents for transporting of these data thus represents a method well supported by programming languages. One example of combining these technologies is with Ajax web-applications. Storing persistent data in a database backend makes easy work of updating, deleting, and inserting data, and using XML to transport the data utilizes one of HTTP's (Hypertext Transport Protocol) supported mime types and allows the document to be manipulated as an object within local javascript code. With this arrangement, each technology is used where it best supports application development.

A specific local project provides a case study of this approach. The project consisted of building a glossary application for web access to commonly used acronyms. Beginning with an XML document to store these simple term/definition pairs seemed appropriate because the storage requirement was so small. After an editing feature was implemented, however, the need to allow multiple updating transactions prompted the use of a RDBMS. No aspects of the relational storage model are used in the single-table database. Rather, the choice was guided by the concurrent transaction support a RDBMS provides, allowing multiple users to edit the glossary through the web interface.

The decision of whether to implement an XML or relational database storage model is typically case specific. For the storage of persistent data, the particular application of that data or the data itself will offer clues to the best approach for storage. The three issues/factors discussed above–portability, data access, and data characteristics–are far from comprehensive but do represent some important broad assessment categories. Personal preferences and experience will always be strongly influential, along with frequently used conventions. Within some application arenas, there may be an existing, generally more acceptable, approach to data storage. With both popular conventions and traditions used in our local informatics environment comes the benefit of standardization and developer support. And so, as expected, there is no simple or single answer to the question of data storage models.

[1] C. M. Sperberg-Mcqueen, World Wide Web Consortium XML and Semi-structured Data(http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=339&page=1)