What is XML, and can it live up to its hype?

XML (eXtensible Markup Language) is much like its counterpart HTML (HyperText Markup Language). It is a set of instructions that tells a program what to do with objects or text. Think of a markup language as the programmers highlighter. For example, in HTML if I wanted something in bold I would say <B>I'm in bold!</B> The <B> called a tag, tells the web browser to start bolding the object and the </B> tag tells the browser to stop bolding the object. XML marks up data, for example if I want you to know my dogs name is spot, in XML I would say <dogs_name>Spot</dogs_name>.

XML has its own set of instructions maintained by the World Wide Web Consortium (WC3) and users can make their own tags. XML's greatest contribution to the EMS community will be the ability to tag data. Tagging data with XML allows users to exchange data in a format that not only contains the data, but also contains the tag that tells us what the data are and how they are formatted.

For Example:

I have a table that stores information about EMS agencies. My table has the name of the agency, the county that they serve, a date of licensure, and a licensure expiration date. If I exported the data into a normal text document it may look something like:

Dixie County EMS, Dixie, 2/1/2002, 2/2/2004
Bland Volunteer Fire Dept, Dixie, 12/1/2000, 12/1/2002
Gold Marbles EMS, Morgan, 04/1/2002, 04/1/2004

I get all the data, but I may not know what the fieldnames are or anything else about the data. If I did not receive a document explaining the data, it would be very easy to confuse the data of licensure with the licensure expiration date

If I represent this table in XML it could look like this:

```xml
<?xml version="1.0"?>
<EMS_Agencies_Table>
   <Agency>
      <Name>Dixie County EMS</Name>
      <County>Dixie</County>
      <DateOfLicensure>2/1/2002</DateOfLicensure>
      <DateOfExpiration>2/1/2004</DateOfExpiration>
   </Agency>
</EMS_Agencies_Table>
```
Here there is a user readable fieldname for every record in the table. I would not need any additional documentation regarding fields to understand what data are contained in the set. Further information regarding field types, lengths, and keys can also be represented in XML. If you have Internet Explorer 5 or higher or Netscape 6 or higher, see the sample here. Information about the data is called metadata - or data about your data. Embedded metadata can be extracted from the dataset and converted into data definition language (DDL) and this can be used to create tables, or complete databases from the dataset. In essence the entire database along with the corresponding data could be sent in XML format and extracted onto a receiving database system without any additional documentation about the data.

XML can also represent hierarchical data. Hierarchical data is best described as items having a parent-child relationship. For example, each EMS agency has employees. The agency is the parent and the employees are the child. Using XML, the information regarding the employee can be distributed along with the information about the employee's EMS Agency.
In this example, there is an implicit parent-child relationship between the Employees and the EMS_Agency_Table because the Employee tags are within the Agency tags. And again, with XML you can see who the employees work for and what other fields are available for the employees by following the tags.

XML also allows simple creation of dynamic HTML tables from a dataset. By creating an XML style sheet (XSLT), data can be parsed into different rows and columns based on the tags that are used. This is demonstrated in our cube browser for the CODES project. Creating dynamic HTML tables before XML meant looping through the data until all pieces were put together, using XML has greatly simplified this process.

**XML Databases**

Many vendors are offering "XML databases". These databases allow users to extract XML data without using special formatting statements. These databases are growing in popularity with many developers, however most database administrators, argue that the data are still stored in a relational manner and therefore labeling the systems as an XML database is a misnomer. However these systems are labeled, XML databases offer the advantage of easily creating XML datasets without writing formatting statements. As with most computer systems, you usually lose something when you add something else. With XML databases the loss comes in the transacting logging and overall scalability of the system. Most XML databases lack the mature transaction logging of larger more scalable systems (such as Oracle, SQL Server and DB2) and therefore become slow as the number of records grows.

**Some Disadvantages:**

There are some disadvantages to XML. To start with, the extra tags can make file sizes huge, typically two to three times the normal size of the data alone. So you should never store the data along with the tags in your database.

Next, XML data will need an XML parser or translator to import or export the data. Most database products, even the newest and most expensive, do not have simple ways of importing XML data. To import XML data you need to create a translate document or something that tells your system what fields exist and, if this is a relational database, what other tables exist and how these tables are related. These translators may be in the form of a style sheet (XSLT) or an XML schema definition. Many software products are available to create and manage XML schemas, but they can be costly and they can also be proprietary, such that you cannot
Another problem can be getting XML data out of databases. Newer database systems (Oracle 9i, SQL Server 2000, DB2 version 7.2 ...) are including XML parsers that allow users to easily create XML datasets, but older database systems lack this capability. To pull XML data out of older systems requires third party software. These packages can range from free to over $10,000 depending on functionality, ease of use, and the database you are trying to pull data from.

Another issue, although relatively minor, is parsing reserved words. This issue recently appeared when formatting an XML document for the web. The document contained a field with a "">" symbol (age >80 years) and the web browser was unable to parse the document correctly because it expected an XML tag when it hit the "">". So if your dataset contains reserved characters, they may need to be reformatted to an ASCII equivalent. For example, the "">" symbol in ASCII format is: "&gt;".

Lastly because users can create their own XML tags, they may not all tag their data in exactly the same way, so you may end up with a lot of XML data that still needs lots of work to aggregate.

**In Conclusion:**

XML can be a very useful tool for exchanging and formatting data, but it won't solve all your data issues, and don't underestimate its complexity or potential cost. However, XML's use will continue to grow and new products may help overcome some of XML's shortcomings.