SQL Server XML Performance Tips

When using the FOR XML clause in your Transact-SQL applications, don't include the XMLDATA option. The **XMLDATA option** returns additional XML schema data that generally is not needed. Because of this, using this option adds extra overhead to your server and network connection, reducing performance. [2000] *Updated 1-10-2006*

*****

The **OPENXML function** in SQL Server 2000 can be useful for processing multiple table inserts within a single database call, reducing overhead and boosting performance. The ability to map an XML document to a rowset representation of a specified portion of the XML document within a stored procedure can maximize the efficiency with which repetitive type inserts are accomplished. [2000] See related article *Updated 1-10-2006*

*****

As you may know, **it can be a performance drag on SQL Server and your web server if you SELECT the same data over and over from SQL Server in order to dynamically create Web pages**. For example, say that you need to display some text on your Web page that is stored in SQL Server, but that it only changes every couple of days. Currently, you are running a SELECT statement from a stored procedure to retrieve the text each time the page is dynamically created and displayed. This can significantly hurt performance, especially if you are getting many page views a second.

One way to help avoid this problem, and to boost SQL Server and your Web server's performance, is to use SQL Server and XML to periodically re-create static content (such as once an hour, once a day, etc.) instead of dynamically pulling the content each time it is needed from SQL Server, and storing this static content on the Web server. This assumes, of course, that the data doesn't change often, as in our example. This method still allows the page to be created dynamically, but when the text is needed to be inserted into the page, it is retrieved locally off the Web server, not from SQL Server each time it is needed, reducing overhead and boosting performance. Click here to see an article on how to do this. [2000] *Updated 1-10-2006*

*****

SQL Server 2000 offers three types of FOR XML queries. They include RAW, AUTO, and EXPLICIT. Each have performance pros and cons.

The RAW type offers the best overall performance, especially if you will be moving a lot of data. The disadvantage of this is that not all XML-based applications are able to use the format the RAW type returns.

The AUTO type offers the next best overall performance, and many more XML-based applications are able to use the format returned, unless of course your application requires XML data in a predefined format.

If your application has to accept data in a predefined format, then you have to use the EXPLICIT type, which is generally the slowest performing option. [2000] *Updated 1-10-2006*

*****

If you need to use the EXPLICIT type of FOR XML query, in some cases you can boost performance if you replace the EXPLICIT FOR XML query with an XPath query instead. XPath queries are faster, and in most, but not all cases, can replace the functionality of a EXPLICIT FOR XML query. [2000] *Updated 1-10-2005*

*****

If you have installed SQLXML 2.0 on your SQL Server 2000 server, you can take advantage of a new feature that can significantly affect SQL Server's performance. SQLXML 2.0 supports what is called the Client Side for XML. What this feature does is to move the conversion from the SQL rowsets to XML at the client, or middle-tier, instead of on the SQL Server. This can significantly reduce the load on SQL Server, helping to boost its performance. Of course, the conversion still has to be done somewhere, and resources on the client or middle-tier must be used to do this. This performance-enhancing feature is especially useful to those whose SQL Server is already operating at full capacity and the workload need to be off-loaded to other servers. [2000] *Added 5-9-2002*

*****

To take full advantage of SQL Server 2000's XML capabilities, you will need to download the SQLXML 2.0

(XML for SQL Server 2000) from Microsoft's website and install in on your current SQL Server 2000 server(s). This is because SQL Server 2000's native XML support is limited. By installing SQLXML 2.0, you get these new XML features:

- Updategrams
- XML bulk load functionality
- XSD mapping schemas
- Client-side XML functionality
- Updated SQLXML OLEDB provider
- The ability to wrap stored procedures with the Transact-SQL FOR XML clause
- The ability to use client-side XML functionality to move XML processing to the middle tier
- A tool to convert XRD mapping schemas to XSD
- Much better performance
- And many more features, not listed here

[2000] *Added 5-9-2002*

*****

SQLXML 2.0 offers three options to use XML documents to modify a SQL Server database. They include OPENXML, Updategrams, and Bulk Load. Each of these have their own performance-related pros and cons.

As you might guess, the best performance is offered by using Bulk Loads, assuming that it will work for you. If you need to insert very large numbers of XML documents as defined by a mapping schema, then Bulk Loads can be very efficient. But in many instances you can't use the Bulk Load option because it is only limited to INSERTs, not other types of data modifications.

The next most efficient way to use XML documents to modify a SQL Server database is to use the Updategrams option. Updategrams are more efficient that OPENXML because all XML parsing and SQL statement generation is not performed by SQL Server, but in the client or middle-tier. Like Bulk Loads, Updategrams can't be used in all instances. Updategrams usually require a mapping schema and also require that XML input be in a special format, which may or may not be practical.

The least efficient, but most flexible way to modify SQL Server data using XML documents, is to use OPENXML. The performance problem is a result that an OPENXML stored procedure is parsed into a DOM by a parser that runs in the SQL Server process, and uses SQL Server memory until it is released. While OPENXML works OK for low-volume applications, it does not scale well. [2000] *Added 5-9-2002*

*****

Here are some ways you can help to **maximize the performance of Updategrams**:

- If you use the sql:key-fields annotation, be sure you select a primary key for each table. If you don't, all columns in the before element are included in the WHERE clause, and can cause the search to take much longer than necessary.
- If you are going to update or delete data a record, and you don't care what the before value is, you can help boost performance of Updategrams by not including all the columns included in the before element in the WHERE clause.
- If an Updategram is used to make unrelated updates, the updates should be split into separate synch blocks. This reduces the transaction size, and helps to boost performance.

[2000] *Added 5-9-2002*

*****

The "Advanced" tab of the "Configure SQLXML Support in IIS" configuration dialog box has a check box to turn "caching" on and off. When on (the default option), allows Mapping Schemas to be cached and used over and over, helping to boost overall performance. Don't turn this option off, unless you are in a development or testing environment. [2000] *Added 5-9-2002*