Craig S. Mullins

Return to Home Page

August / September 2007

zData Perspectives

by Craig S. Mullins

DB2 9 for z/OS Improves LOB Support.

Although there are a lot of nice, new features in DB2 9 for z/OS, many will be happy that IBM has improved DB2's ability to store and manage LOB data. As anyone who has tried to use LOBs in a previous version of DB2 knows, the usability limitations were troublesome. But with Version 9, IBM chips away at some of the more annoying LOB limitations.

FETCHing LOBs

Prior to V9, there were two methods you could deploy in your programs to fetch LOB data:

- Fetching data into a pre-allocated buffer
- Using a LOB locator to retrieve a handle on the data.

Both methods could be troublesome. Fetching data into a preallocated buffer can cause virtual storage constraint problems, especially for larger LOBs. Using LOB locators while committing infrequently or failing to explicitly free the locators can use considerable amounts of DB2 resources.

V9 introduces a new clause, WITH CONTINUE, for use on your FETCH statements. By coding your program to use WITH CONTINUE you can retrieve LOB columns in multiple pieces without using a LOB locator, and continue a FETCH operation to retrieve the remaining LOB data when truncation occurs. (Note: this method can be used with XML data, too.) You will have to manage the buffers and reassemble the pieces of data in your application program.

So, by specifying WITH CONTINUE on your FETCH statement you tell DB2 to allow subsequent FETCH CURRENT CONTINUE operations. These will allow you to access the remaining truncated LOB (or XML) column after the initial FETCH. If truncation occurs, DB2 will remember the truncation position and will not discard the remaining data. DB2 will return the total length that would have been required to hold all of the LOB data. This will either be in the first four bytes of the LOB host variable structure or in the 4 byte area that is pointed to by the SQLDATALEN pointer in the SQLVAR entry of the SQLDA for that host variable.

File Reference Variables

DB2 V9 also adds support for a LOB file reference variable. A file reference variable is a host variable that is defined in your host language code to contain the file name that directs file input and output for a large object (LOB).

Using file reference variables, large LOB values can be inserted from a file or selected into a file rather than a host variable. This means that your application program does not need to acquire storage to contain the LOB value. File reference variables also enable you to move LOB values from the DBMS to a client application or from a client application to a database server without going through the working storage of the client application.

LOBs and Utilities

The manner in which DB2 handles LOBs in utility processing has also been improved in DB2 V9.

Loading and unloading LOBs has been improved. For LOAD, an input field value can contain the name of the file that contains a LOB column value. The LOB column value will then be loaded from that file. For UNLOAD, you can store the value of a LOB column in a file and record the name of the file in the unloaded record in the base table.

What about REORG? Well, prior to V9 you could not access LOB data during a REORG. And a REORG did not reclaim physical space from the LOB data set because LOBs were moved within the existing LOB table space. V9 fixes

these problems. During a REORG (in V9), the original LOB table space is drained of writers. All LOBs are then extracted from the original data set and inserted into a shadow data set. When this operation is complete, all access to the LOB table space is stopped (the readers are drained) while the original data set is switched with the shadow data set. At this point, full access to the new data set is enabled, and an inline copy is taken to ensure recoverability of data.

Additionally, both CHECK LOB and CHECK DATA have SHRLEVEL REFERENCE and SHRLEVEL CHANGE options. Using these you can minimize downtime. For LOBs, CHECK DATA checks for consistency between a base table space and the corresponding LOB (or XML) table spaces. The CHECK LOB utility identifies structural defects in the LOB table space and any invalid LOB values. Running CHECK (DATA or LOB) with SHRLEVEL REFERENCE indicates that applications can read from but cannot write to the index, table space, or partition that is to be checked. SHRLEVEL CHANGE means that applications can read from and write to the index, table space, or partition that is to be checked.

Performance

Finally, DB2 V9 provides several performance improvements related to LOBs. Using file reference variables or WITH CONTINUE to “chunk” read LOBs can improve performance over using locators.

And as we all know, removing locks can improve performance. DB2 V9 eliminates LOB locking for space allocation, as well as for insert, delete, update, and select operations. Additionally, a LOB lock is no longer required

to serialize the consistency between the value of the LOB and the column of the base row for an uncommitted read operation.

From zJournal, Aug / Sept 2007

.

Home.