Craig S. Mullins

Return to Home Page

August / September 2008

The Resource for Users of IBM zSeries & S/390 Systems

z/JOURNAL

zData Perspectives

by Craig S. Mullins

DB2 9 Data Format "Stuff"

Today's column will focus on some of the data format "stuff" that changes with DB2 9 for z/OS. I know that "stuff" isn't really a very targeted technical term, but it nicely fits what I'll be addressing here.

Reordered Row Format

If you've worked with DB2 for awhile, especially as a DBA, you've probably heard the advice to re-arrange the columns of your tables to optimize logging efficiency. Basically, the more data that DB2 has to log, the more overhead your programs will incur, and performance will degrade. DB2 logs data from the first byte changed to the last byte changed – unless the row is variable, in which case DB2 will log from the first byte changed to the end of the row – unless the change does not cause the length of the variable row to expand, in which case DB2 goes back to logging from the first byte changed to the last byte changed.

So, the general advice goes something like this: put your static columns (those that do not change frequently) at the beginning of the row and your dynamic columns (those that will change more frequently) at the end of the row. And put your variable columns at the end of each.

Well, DB2 9 for z/OS takes this advice to heart (sort of). In New Function Mode (NFM), for new table spaces, DB2 will automatically put the variable columns at the end of the row. This is called reordered row format (RRF); the row format we are all familiar with today is now referred to as basic row format (BRF). This only impacts how the data is stored on disk – it does not mean that your DDL is changed nor does it require changes to anything external or how you access the rows.

To summarize, a row in RRF will store the fixed-length columns first and the variable columns at the end. Pointers within the row will point to the beginning of the variable columns.

So far, so good; but DB2 will also convert our old table spaces to RRF over time. Once we are in DB2 9 NFM, a REORG or a LOAD REPLACE will cause a change from BRF to RRF. So if you run a LOAD REPLACE on a table in NFM, its table space will have the row format changed to RRF. REORG a partition and the row format for that partition changes. And yes, you can have a partitioned table space with some partitions in BRF and some in RRF.

With BRF we can be sure that DB2 is putting our variable columns at the end of the row – where they belong. But it still is not helping us with placing static columns before the dynamic ones. You'll still have to guide DB2 to do that.

Do You Want to Ignore Clustering?

DB2 9 for z/OS also offers a new DDL parameter for your tables: APPEND. If you specify APPEND NO, which is the default, DB2 will operate as you are accustomed to it operating. That is, when rows are inserted or loaded DB2 will attempt to sequence them based on the clustering index key.

If you specify APPEND YES though, DB2 will ignore clustering during inserts and online LOAD processing. Instead, DB2 will just append the rows at the end of the table or partition. You might want to choose this option to speed up the addition of new data. Appending data is faster because DB2 does not have to search for the proper place to maintain clustering. And you can always re-cluster the table by running a REORG.

The APPEND option cannot be specified on LOB tables, XML tables, or tables in work files.

To track the state of the APPEND option there is a new column, APPEND, in the DB2 Catalog in SYSTABLES. Its value will be either 'Y' or 'N'.

Other "Stuff"

These are not the only two features that are format-related. DB2 9 introduces a universal table spaces that combine the attributes of segmented and partitioned table spaces. With partition-by-growth universal table spaces, DB2 automatically adds partitions as needed to support your rapidly-growing data. And remember that you will no longer be able to create simple table spaces in DB2 9 NFM.

Let’s not forget the new data types. DB2 9 delivers DECFLOAT (decimal floating point), BIGINT (8 byte integer), BINARY and VARBINARY types, in addition to pureXML data types. And you can create HIDDEN columns that won’t show up in a SELECT *, which is not exactly a format issue, or is it?

Don’t forget the index format improvements. DB2 9 allows you to compress indexes for the first time; and you can create indexes on expressions, instead of just on columns. Not to even mention clone tables… You see, DB2 9 brings us a lot of good format “stuff!”

From zJournal, August / September 2008
.

Home.