It’s Not Your Daddy’s DB2! – Part 2

Universal Table Spaces

In Part 1 of this series we took a look at the trends that impact the world of database systems and database administrators in general. With today’s post we will begin to look at the major changes and challenges facing the world of DB2 for z/OS in the digital era.
One of the biggest changes in the last decade or so has been the introduction of new types of table spaces – known as universal table spaces, or UTS. Not only are UTS new to DB2, they are quickly becoming the de facto standard type of table space for DB2 applications, new and old. At some point, universal table spaces will displace your existing segmented and classic partitioned table spaces. We’ll get into why this is so later in the post, but first let’s briefly describe what universal table spaces are.

Two Types of Universal Table Spaces

New as of DB2 9, universal table spaces combine the best attributes of partitioned and segmented table spaces. Universal table spaces offer improved space management for variable length rows because they use space map pages (like segmented table spaces). Also, like segmented table spaces, UTS deliver improved mass delete performance, and you can immediately reuse the table segments after the mass delete. And like partitioned table spaces, universal table spaces can grow large (up to 128TB of data) and consist of multiple partitions.
There are two types of universal table spaces:

1. **Partition-by-growth (PBG):** The PBG UTS creates new partitions as the amount of data grows without the need to specify key ranges. This type of UTS is beneficial for tables that grow over time and need the additional limits afforded by partitioning but can benefit from the performance of segmented.

2. **Partition-by-range (PBR):** The range-partitioned, or PBR UTS requires a key range for partitioning like classic partitioned table spaces. A PBR UTS basically adds segmentation to the existing partitioned table space.

Both types of UTS can contain only a single table, but IBM presentations have indicated that this is likely to change at some point in the future.

A partition-by-range UTS is basically a segmented, partitioned table space. The limit key ranges must be specified in the table DDL. Index partitioning, which was supported for the earliest classic partitioned table spaces, is not supported for a PBR UTS. So before converting your classic partitioned table spaces to PBR UTS, you must first convert from index-controlled partitioning to table-controlled partitioning. Check out this blog post for a trick to quickly convert to table-controlled partitioning.

The second type of UTS is the partition-by-growth Universal table space. As its name implies, a PBG UTS can automatically add a new partition as the data in the table space grows. Over time, as the UTS is used by applications, data gets added to the table. When the PBG UTS reaches its maximum size, a new partition is automatically added to the table space. The new partition uses the same characteristics as the existing partitions, including compression details, free space, and so on.

You control the type of UTS using the DDL keywords: NUMPARTS,
MAXPARTITIONS, and SEGSIZE. To create a PBR UTS you specify both NUMPARTS and SEGSIZE. To get a PBG UTS you must code the MAXPARTITIONS and SEGSIZE parameters. MAXPARTITIONS indicates the limit on the total number of partitions that a PBG UTS can grow to. Be careful, because if you only code the NUMPARTS parameter without SEGSIZE, then you will create a traditional partitioned table space. If you only code the SEGSIZE parameter (without either NUMPARTS or MAXPARTITIONS) you will create a traditional segmented table space.

Why Are Universal Table Spaces the Future of DB2?

As of today (DB2 11 for z/OS), there are basically four types of table spaces from which to choose:

1. Segmented table spaces
2. Universal Partition-By-Growth (PBG) table spaces
3. Universal Partition-By-Range (PBR) table spaces
4. Classic partitioned table space

Of course, for new databases, it is best to remove the classic partitioned table space from consideration because the PBR UTS is more efficient (and classic partitioning will likely be deprecated at some point in the future). Technically, there are two other types of table spaces (LOB and XML table spaces), but they can be used only in specific situations (and we will talk about LOBs in a future post in this series).

So why do I advise that you favor Universal table spaces over segmented whenever you can? Well, for a number of reasons. First of all, because Universal table spaces are newer and all you really need for most every DB2 implementation. Secondly, because many new features of DB2 can only be used with Universal table spaces. Newer features that only work with UTS include:

- Clone tables
- Hash-organized tables
- Currently committed locking
- Pending DDL
- Inline LOBs
- XML multi-versioning
- ALTER TABLE with DROP COLUMN

And this trend is likely to continue. As IBM introduces new versions of DB2 with new features that only work with UTS, it will become increasingly difficult for DBAs to keep track of which table spaces are not UTS so that they can make sure they are not using any new features that will not work with their old types of table spaces.

What this means is that you really should only be using segmented table spaces today when you absolutely must have a multi-table table space. A best practice that I recommend is to make all new table spaces Universal (except for multi-table table spaces which can be segmented).

So what is the future of the segmented table space? For the immediate future, segmented table spaces will continue to be supported. My guess is that, at some point, IBM will deliver a multi-table UTS capability, and then at some point deprecate segmented table spaces. But this is only my guess. As of the date I am writing this, IBM has not committed to a multi-table UTS and the segmented table space is still the recommended (and only) method doing of putting multiple tables into a single table space. My recommendation is that you avoid multi-table table spaces unless you have many very small tables and are close to reaching the open data set limit (200K as of DB2 V11).

My general recommendation for table spaces is to slowly work on a conversion project to migrate your classic partitioned table spaces to PBR UTS and your segmented table spaces to PBG UTS. Doing so will bring you to the latest and greatest DB2 table space technology and position you to be able to use all new functionality in current and future versions of DB2 whenever – and wherever – you see fit.
Summary

To make sure that you systems are up-to-date and ready for new functionality it makes sense to adopt Universal table spaces for all of your DB2 tables. The only exception is for multi-table segmented table spaces, and you shouldn’t have too many of them.

Good luck universalizing your DB2 databases!