Structure and Format Enhancements in DB2 9 for z/OS

By Willie Favero

DB2 9 for z/OS has plenty of great features; everyone is bound to find something they like in the latest version. This article explores:

- A feature that has generated considerable discussion—a new structure type called universal table space
- An enhancement that you may not hear all that much about—a new storage format referred to as reordered row format.

Both of these will significantly improve your DB2 experience by making a couple aspects of DB2 easier to manage.

You may not have heard a lot of clamoring for universal table space, but it's something many people have wanted for years. As you deploy DB2 9, you'll find this to be one of the more useful features and I believe it will soon become the de facto standard for building DB2 objects.

With DB2 for z/OS V8, you have four types of table spaces to choose from:

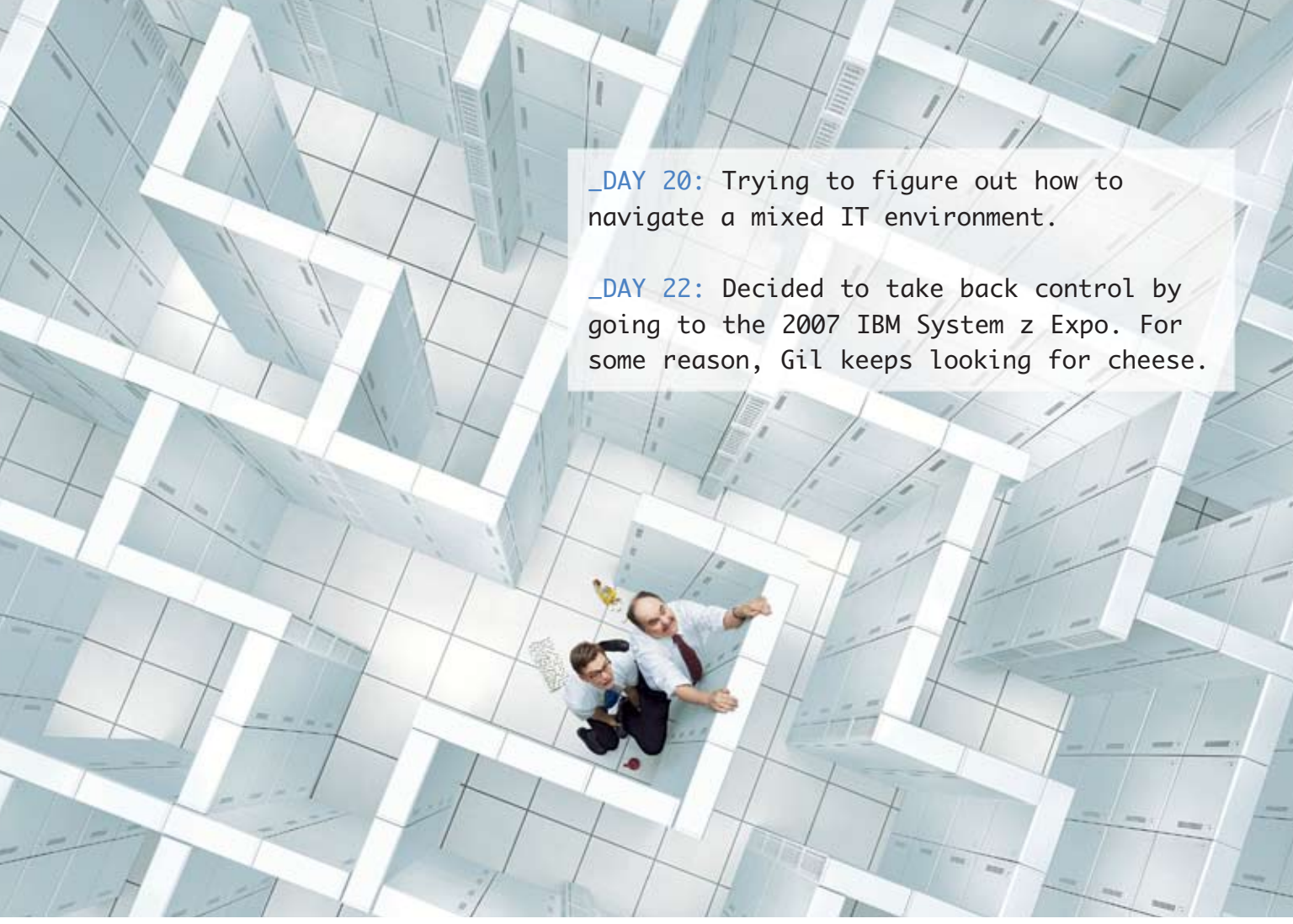
- **Simple:** multiple tables per table space and all rows from all tables can share the same page in that table space. Simple table spaces go back to the dawn of DB2 on the mainframe. A simple table space has a maximum size of 64GB.
- **Segmented:** segmented table spaces, which didn't arrive in DB2 until V2, let you create multiple tables in a single table space. However, each table is contained in its own segment. You get to pick the segment size, something between four pages and 64 pages in multiples of 4 (4, 8, 12, etc.). A segmented table space also has a maximum size of 64GB.
- **Partitioned:** allows multiple partitions; each partition can have a possible maximum of 64GB and you can define from one to 4,096 partitions in current releases of DB2. However, only one table is allowed per partitioned table space.
- **Large Object (LOB):** the last table space type is for LOBs. If an LOB table space is defined, the rows defining the LOB are contained in a separate table space called the base table space.

When you upgrade to DB2 9, you'll gain two new table space types. However, you end up with only a total of five table space types. It's the new math. In DB2 9, you can no longer create a simple table space. Although you can still use and alter existing simple table spaces, any newly created table space in DB2 9 can't be simple. The five table space types available to you in DB2 9 still include segmented, partitioned, and LOB. The two new table spaces added in DB2 9 are:

- **Universal** (the subject of this article)
- **XML**, which holds the new XML data in DB2 9. An XML table space is a little like an LOB table space. If an XML table space is defined, the rows defining the XML are contained in a separate table space called the base table space.

What's a universal table space? The simplest way to explain a universal table space is to say it's a cross between a partitioned table space and a segmented table space, hopefully giving you many of both of its parent's best features. When using a universal table space, you get the size and growth of partitioning while maintaining the space management, mass delete performance, and insert performance of a segmented table space. It's kind of like having a segmented table space that can grow to a 128TB of data (assuming the right DSSIZE and right number of partitions are specified) that also gives you partition independence.

Another benefit of a universal table



Register today for the 2007 IBM System z Expo

When: September 17 – 21, 2007

Where: San Antonio, Texas – San Antonio Marriott Rivercenter and Marriott Riverwalk Hotels

Cost: $2,095

The power of many, the simplicity of one.

The IBM mainframe has experienced a strong resurgence in the marketplace over recent years, largely as a result of its continued innovation and leadership in key technologies such as multi-dimensional virtualization, security, resiliency, workload management and integration. This event provides expert insight about the IBM System z and how it provides the foundation for helping to integrate and manage your mixed IT environment.

Sessions will cover:

- *IBM System z9™ BC and EC server updates*
- *z/OS® V1.9 improvements*
- *z/VM® V5.3 enhancements*
- *z/VSE™ Version 4 refinements*
- *Linux® on System z™ developments, Linux Utilities*
- *WebSphere® Application Server for z/OS V6 Implementation*
- *Server Time Protocol, GDPS® and Parallel Sysplex®*
- *Service Oriented Architecture (SOA)*
- *WebSphere MQ, DB2®, CICS®, IMS™*
- *Virtualization*
- *IBM System z Security*
- *Professional Development*

To enroll and for more information, visit
ibm.com/training/us/conf/systemz
Or call **1-800-IBM-TEACH** (1-800-426-8322)

Attention Exhibitors: *Information about exhibition and sponsorship opportunities are available at the conference Web site to the right, click on "Be an exhibitor."*

IBM®

space is the ability to use cloned tables; a universal table space is a prerequisite to using CLONE tables in DB2 9. Of course, like the original partition table spaces, a universal table space can contain only one table per table space. A universal table space also must be managed by DB2 (using DB2 storage groups). A universal table space can't be a user-managed table space. Reordered row format, the last subject of this article, is always used for universal table space. Finally, before you get too excited, realize you won't be able to take advantage of universal table space until DB2 9 for z/OS New Function Mode (NFM).

Universal table spaces come in two flavors: partition-by-growth and range-partitioned (a.k.a. partition by range). Partition-by-growth universal table spaces have no partitioning columns and grow "on demand." As data is inserted into a partition-by-growth universal table space, a new partition is automatically created when the current partition becomes full. A range-partitioned universal table space is similar to the partition table spaces we've all grown up with, with the exception that it's segmented with all the advantages of a segmented table space.

Let's take a closer look at partition-by-growth universal table spaces. When a partition of a partition-by-growth universal table space reaches its maximum size from the addition of new rows, a new partition is automatically added to the table space. The new partition takes on the characteristics of the previously filled partitions, including using the same dictionary if compression is turned on. In addition to copying over the dictionary, the new partition has the same FREESPACE, DEFINE, logging, and TRACKMOD attributes.

Partition size, and how much data a single partition of a partition-by-growth universal table space can hold is determined by the DSSIZE keyword, or DSSIZE default if the keyword isn't specified. If DSSIZE is 2GB, a new partition is added when that 2GB partition is full. If a partition-by-growth universal table space is growing (adding partitions) on its own, what prevents a runaway application from taking over all your available disk space while building the maximum number of partitions?

DB2 9 for z/OS has a new keyword on the CREATE/ALTER SQL statement: MAXPARTITIONS. As the parameter implies, MAXPARTITIONS tells DB2 how many partitions a partition-by-growth universal table space can use before reaching its maximum number of partitions, therefore, stopping its growth. If you don't specify the MAXPARTITIONS keyword, it defaults to 256 partitions. You also should specify the segment size for the table space. If you don't specify SEGSIZE, it defaults to four pages. A four-page segment size typically isn't optimal for whatever use you have planned for the table space.

You can specify several combinations of NUMPARTS, MAXPARTITIONS, and SEGSIZE. If NUMPARTS and SEGSIZE are specified, you get a range-partitioned (partition by range) universal table space. If only the NUMPARTS keyword is used (no SEGSIZE specified), the result is a traditional partition table space that won't take advantage of segmentation. If, however, only SEGSIZE is used (no NUMPARTS or MAXPARTITIONS keywords specified), you end up with a traditional segmented table space, with no partitioning. Remember, there's no longer any combination of keywords that will let you create a simple table space.

DB2 9 for z/OS also lets you implicitly create a partition-by-growth universal table space using the CREATE TABLE statement. The new phrase PARTITION BY SIZE EVERY xxx G (where xxx is an integer less than or equal to 64) tells DB2 to create this table in a partition-by-growth universal table space. If a table space name is specified on the CREATE TABLE's IN clause, it must point to a partition-by-growth table space. If a table space is specified, the EVERY xxx G must specify the same value as the table space's DSSIZE. If the table space is implicitly created, it's created with LOCKMAX set to SYSTEM, MAXPARTITIONS equal to 256, SEGSIZE equal to 4, and DSSIZE equal to 4G. Choose carefully before letting DB2 implicitly create a partition-by-growth table space. The only way to change the SEGSIZE or DSSIZE of an implicitly created partition-by-growth table space is to drop and re-create the table space. There's no ALTER TABLESPACE option for DSSIZE and SEGSIZE.

The DB2 catalog also has had a few changes in support of universal table space. The column TYPE in the catalog table SYSIBM.SYSTABLESPACE has some new values:

- K for a range-partitioned table space
- G for a partition-by-growth table space
- P for an implicit table space created for XML columns.

There's also a new column in SYSIBM.SYSTABLESPACE. MAXPARTITIONS contains (yes, you guessed it) the maximum number of partitions for a partition-by-growth table space. The existing PARTITIONS column in SYSIBM.SYSTABLESPACE will contain the number of physical partitions that currently exist in a partition-by-growth table space.

The other significant change in DB2 9 for z/OS is how the order of a table's columns is maintained when the underlying data is stored on disk. An ongoing challenge for DB2 professionals has been the placement of the fixed length (CHAR) columns vs. the variable length (VARCHAR) columns when defining a row for a newly created table. There have been many different opinions about what's correct, efficient, and best. Most agreed that the VARCHAR columns should go at the end of the row. However, no matter in what order you place the column, once DB2 hits a VARCHAR column in a row, it has to scan the row to find the rest of the columns. Until DB2 reads the length on the VARCHAR column (the two-byte prefix on every VARCAHR column), it has no idea where the subsequent columns in that row begin.

DB2 9 has a strong solution to this quandary. Once you get to DB2 9 for z/OS NFM, all VARCHAR columns on newly created table spaces will be placed at the end of the row. DB2 9 can make this type of decision because the row format in DB2 9 has changed. DB2 9 introduces "reordered row format," which is a straightforward, simple concept. A DB2 9 NFM row will have all the fixed length columns first, followed by pointers to the beginning of each VARCHAR row. The pointers are followed by the actual variable length data. Rather than scan what could be some lengthy columns just to find the beginning of the column you're looking for, DB2 needs to scan only the list of pointers to find the location for the beginning of the column you want.

As stated earlier, get reordered row format for any table space created in DB2 9 NFM. You'll also have your table spaces, or table space partitions, converted to reordered row format when a REORG or LOAD REPLACE is run against a table space or table space partition. Watch out for the partitioning piece. If you REORG only selected partitions of a table space, then you'll end up with some partitions—the ones that have been reorganized in the new refor-

matted row format. The remaining partitions that haven't yet been reorganized will stay in basic row format. Basic row format is the phrase used to describe the current row format, the row format prior to DB2 9 for z/OS.

There's a potential issue that might arise from using reordered row format. It concerns those of you who have just moved to DB2 9 for z/OS NFM and also use table space compression. If you're satisfied with your compression's dictionary, when you run REORG or LOAD REPLACE, you should probably specify the keyword KEEPDICTIONARY. KEEPDICTIONARY avoids a dictionary rebuild, a task that could add a bit of time to your REORG process. With DB2 9, the first access to a table space by REORG or LOAD REPLACE changes the row format from basic row format to the new reordered row format, assuming you're in DB2 9 NFM. It's anticipated that rebuilding your compression dictionary after converting to reordered row format will yield a more efficient compression dictionary. The possible problem is that many shops have KEEPDICTIONARY specified in their existing REORG and LOAD job streams and the dictionary won't get rebuilt. APAR PK41156 was introduced to avoid forcing everyone to change all their jobs for just one execution to get the dictionary rebuilt. APAR PK41156 modifies REORG and LOAD REPLACE so they ignore KEEPDICTIONARY for that one-time run when the rows are reordered; this allows for a rebuild of the dictionary regardless of the KEEPDICTIONARY setting.

However, what happens if you really don't want to do a rebuild of the compression dictionary right now? How do you get around this APAR's change? Well, the APAR also introduces a new keyword for REORG and LOAD REPLACE that gives you a workaround and still doesn't require you to change your jobs if you simply want DB2 to rebuild the dictionary. The new keyword is HONOR_KEEPDICTIONARY and it defaults to NO. So, if you don't specify this keyword, your dictionary will be rebuilt. However, if you do want to "honor" your current dictionary, you can add this keyword to your REORG or LOAD REPLACE job and things will behave as they did previously. This pertains only to table space compression; index compression doesn't use a dictionary.

SOUND OFF !
Comment on this article by visiting http://community.zjournal.com/DB2, and clicking on the thread titled "Structure and Format Enhancements in DB2."

The plan is to also convert a table space to reordered row format if an ALTER ADD PART or an ALTER ROTATE PART is performed on the table space and you're in NFM and none of the exclusions mentioned in the previous paragraph exist on the table.

However, there are exceptions to the aforementioned process automatically occurring. The DB2 catalog and directory table spaces and LOB table spaces won't be converted to the new reordered row format. In addition, any table space containing *(Continued on page 86)*

personal information typically include selective encryption of stored data, separation of duties, and centralized, independent audit functions.

Auditing your network is one of the first steps toward achieving compliance with whatever regulations impact your industry. But after you perform that assessment, it may become alarmingly apparent that complete compliance doesn't dovetail neatly with existing business procedures. Often, changes in practices and policies are required, in addition to technical solutions, to achieve full compliance.

It's easy to feel overwhelmed by compliance requirements, but these regulations aren't something you can afford to ignore. Apart from businesses' implicit responsibility to protect consumer data, fines for non-compliance are steep—and costs for publicly reportable data breaches are punishing. All it takes is one successful attack to wipe out years of "savings" on not implementing security. With security experts agreeing that online crime has become more sophisticated, more frequent, and better organized over the past several years, do you really want to risk your bottom line as well as customer and investors' confidence on the hopeful idea that it just won't happen to you?

Finally, remember the importance of creating a culture of security in the corporate environment, a culture where every employee—from the newest sales associate to the CEO—understands the importance of data privacy and protection. When companies have an embedded culture, everything people in that company do reflects that culture. Simply following compliance guidelines to the letter ensures that your organization may technically be in compliance, but that's not enough. Security measures not understood nor fully embraced across the enterprise can and will be circumvented. We can't count on software to completely protect our systems. Smart policies, procedures, and people are just as important as choosing the right security solution. Z

About the Author
Ulf T. Mattsson is CTO at Protegrity and creator of the company's database security technology. His extensive IT and security industry experience includes 20 years with IBM as a manager of software development and a consulting resource to IBM's research and development organization in IT architecture and IT security.
Email: ulf.mattsson@protegrity.com

a table that uses an EDITPROC or VALIDPROC also won't be converted.

As you move to reordered row format, you should be aware that the reordered row format only affects how your data is stored on disk; it has no effect on the logical order of columns. Your columns will still be returned in the order specified when the table was created and new columns (ALTER ADD) will still logically go to the end of the table description. Reordered row format also will add a few new columns to the DB2 9 catalog and make changes to the format of some log records.

You also need to be cautious when using DSN1COPY with the OBID translation option (OBIDXLAT) to move data between DB2 table spaces. If one table space uses reordered row format and the other table space uses basic row format and you want to move their data, you can't. The row formats of the table spaces being copied must match. If they don't, you can get unpredictable results. The good news is that the catalog table SYSIBM.SYSTABLEPART has a new column called FORMAT that identifies what record format the table space is using. If it's the original, pre-DB2 9 basic row format, the column FORMAT will be blank. If the table space has been converted to the new DB2 9 reordered row format, the column FORMAT will contain an "R" (without the quotes, of course). Take the time to check before doing a copy.

Conclusion

This article is only an introduction to universal table space and reordered row format. There's still more to the story and more to learn. If you'd like to read more about these new features and the rest of the enhancements in DB2 9 for z/OS, check out the latest DB2 Redbook from IBM, *DB2 9 for z/OS Technical Overview* (SG24-7330). Z

About the Author
In the past 29 years, Willie Favero has been a customer, worked for IBM, worked for a vendor, and is now an IBM employee again. He has always worked with databases and has more than 20 years of DB2 experience. A well-known, frequent speaker at conferences and author of numerous articles on DB2, he's currently with North American Lab Services for DB2 for z/OS, part of IBM's Software Group.
Email: wfavero@attlgobal.net
Website: www.ibm.com/software/data/db2/support/db2zos/

associated with their mainframes—while they find it difficult to achieve similar visibility with other server platforms. But it's still clear that the cost of hosting most comparable workloads on mainframes is generally much lower than on alternative platforms.

IBM's latest specialty engines, in particular, offer an exceptional opportunity to lower processing costs. They're designed to run certain types of workloads at one-fourth or less of the cost of running them on a general-purpose engine. The system ensures these workloads are managed in a manner consistent with those dispatched on general-purpose engines while retaining independent accountability for performance management and charge-back purposes. Workloads that can be redirected from or isolated to specialty engines include:

- All work hosted by Linux system images (IFL)
- Any pure Java code work that executes within a Java Virtual Machine (JVM) hosted by the z/OS operating system (zAAP)
- All XML system services such as the non-validating parsing service (zAAP) system-oriented work such as data encryption, communications, and transaction monitoring (zIIP).

The mainframe continues to deliver high business value. It remains the gold standard by which all other platforms are measured in terms of reliability, security, and scalability. That's why phrases such as "mainframe-like" are often used as descriptive terms by providers of alternative computing platforms.

Clearly, the mainframe is experiencing a renaissance of innovation that continues to drive down TCO. For IT organizations under constant pressure to do more with less, this could mean falling in love with the mainframe—again. Z

About the Authors
Ron Higgin is vice president and principal architect in CA's Office of the CTO.
Email: ron.higgin@ca.com

Marcel den Hartog is director of marketing, EMEA, at CA.
Email: marcel.denhartog@ca.com