bmc

Thought Leadership White Paper

Embrace Digital Transformation with IBM® DB2®

Maximize DB2 efficiencies with data management best practices

By Craig S. Mullins

Table of Contents

Executive Summary

Organizations today are being realigned by digital transformation, which comprises the many changes associated with the application of digital technology to all aspects of business and society. Digital technology is pervasive in the form of smartphones and tablets. The Internet of Things brings connectedness to all of our devices. This burgeoning technology and increased connectedness impacts all computing platforms, including IBM® DB2® for z/OS®.

Indeed, **DB2 for z/OS is changing rapidly and these changes are altering the way in which DB2 databases and applications are managed, administered, and developed**.

The industry is changing, the way that DBAs work is changing, and DB2 is changing. Today's DB2 capabilities and infrastructure differ significantly from what they were 15, 10, or even 5 years ago—and we all need to come to grips with the fact that the way we worked with DB2 in the past is no longer the way we work with today's modern DB2. Challenges include:

- Universal table spaces
- Large objects for storage
- Modern storage
- Analytics
- New coding techniques

This paper will examine these functional and structural changes to DB2, and how, once appreciated, can benefit your organization.

INDUSTRY AND DBA TRENDS

Recent industry trends are impacting IT and business and thereby contributing to the changes that IBM has been making to DB2 for z/OS. The first, and most obvious trend, is that we are saving more data than ever before. According to a recent study by IDC the digital universe will continue growing at 40 percent a year into the next decade. By 2020, IDC estimates that the digital universe will grow to 44 zetabytes, or 44 trillion gigabytes.[1]

40%

Growth

The digital universe will continue growing at 40 percent a year into the next decade.

By 2020, IDC estimates that the digital universe will grow to 44 zetabytes, or 44 trillion gigabytes.[1]

The amount of data being generated continues to grow for many reasons, including the move to an online business model, the growth of smart device usage, social media, and the Internet of Things, not to mention the desire to derive insight from analytics on large amounts of data. The ability to collect large amounts of data has given rise to the big data industry trend, which is in itself a collection of trends. **The big data movement is about being able to quickly derive meaning and insight from vast quantities of data—both structured and unstructured—in order to improve business decision making**.

Big data initiatives are driving major shifts in IT requirements and technology. The phenomenal growth of data and the speed at which it is being generated offers an opportunity for uncovering information and trends that were heretofore unknown.

As a final note here, IDC estimates that the amount of unstructured data (that is, data that is not strictly numeric, character, or date/time) accounts for 90 percent of all digital information.[2] Moreover, unstructured data increasingly is finding its way into our database systems.

IDC estimates that the amount of unstructured data (that is, data that is not strictly numeric, character, or date/time) accounts for 90 percent of all digital information.[2]

CHANGES TO DATABASE ADMINISTRATION

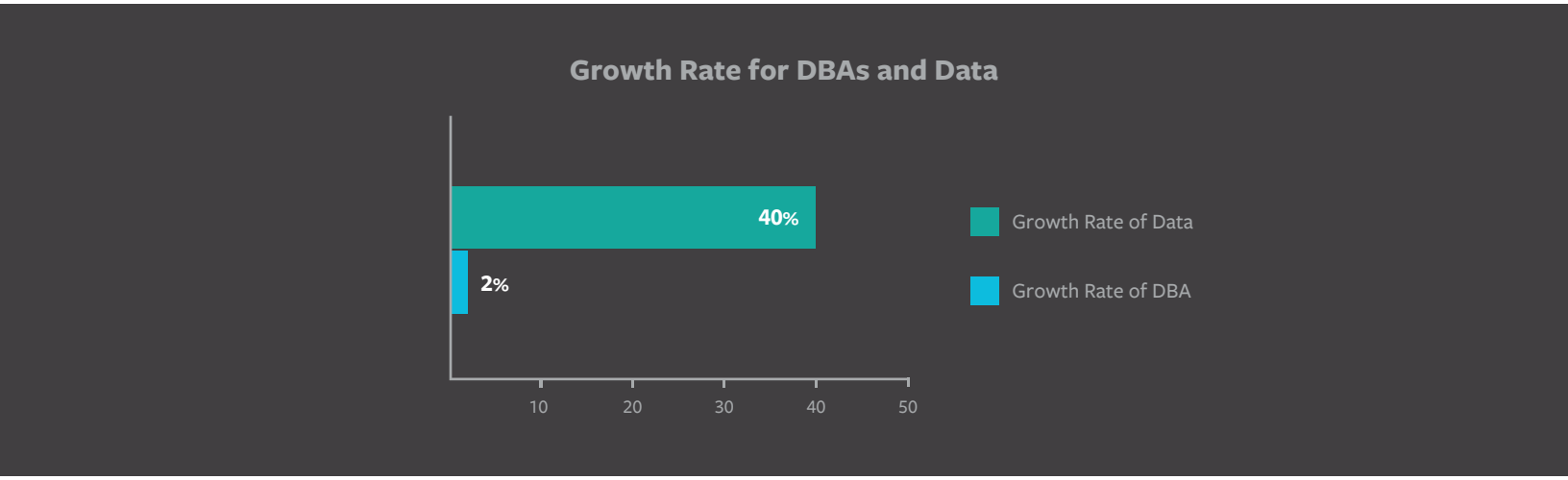
Database administration is another lens through which we can view the current state of data and DB2. DBAs are responsible for ensuring the ongoing operational functionality and efficiency of an organization's databases and the applications that access that data. But modern DBAs are relied upon to do far more than just keeping database systems up and running. Most DBAs have years of IT experience and many work on technologies other than database systems. The DBA's experience often encompasses areas as diverse as application development, middleware implementation, transaction processing, business intelligence, and networking. Therefore, the modern DBA's job responsibilities span more than just the database.

1 The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things, IDC Corporation, April 2014.

2 Bob Violino, Unstructured Data Offers Big Benefits, Information Week, July 23, 2014.

An additional change is that there are far fewer single-DBMS DBAs than there used to be. Twenty (or even ten) years ago, it was common for a DB2 for z/OS DBA to work on only DB2 for z/OS. However, in this day-and-age of heterogeneity, many DBAs increasingly are tasked with managing multiple DBMSes (DB2 and Oracle®; or DB2 z/OS, DB2 for Linux, UNIX, and Windows and IMS™; or even DB2 and a NoSQL database). When focus is diluted in this manner it becomes difficult for DBAs to be the DBMS-specific experts that developers and management expect them to be.

Furthermore, fewer DBAs are being asked to manage more data. More and more data is being stored and accessed, but that is not translating into additional DBAs being hired. Although the Bureau of Labor Statistics[3] projects the growth rate for DBAs as "faster than average" at 11 percent between the years of 2014 and 2024, that is still less than 2 percent per year.[3] And when the amount of data is growing at 40 percent a year into the next decade, it isn't hard to see that the number of people that manage data is not keeping up with the amount of data that must be managed.



CHALLENGE: UNIVERSAL TABLE SPACES

All of these big data and DBA trends factor into the changing world of DB2 for z/OS. But what are the significant DB2 changes that are challenging the DBAs and developers?

One of the biggest recent changes to DB2 for z/OS was the introduction of new types of table spaces, known as **universal table spaces**, or UTS. UTS are quickly becoming the de facto standard type of DB2 table space and at some point, UTS will displace your existing segmented and classic partitioned table spaces.

Universal table spaces combine the best attributes of partitioned and segmented table spaces, offering improved space management because they use space map pages (like segmented table spaces). 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: partition-by-growth (PBG) UTS where new partitions are created as needed as the amount of data grows, and partition-by-range (PBR) UTS, which require a key range for partitioning like classic partitioned table spaces. 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.

As of DB2 11 for z/OS, it is still possible to choose segmented and classic partitioned table spaces, but universal table spaces are preferred over segmented and classic partitioned table spaces. The only time where a different choice still makes sense is for table spaces with more than one table, in which case segmented table spaces are the only option.

Using UTS for almost all of your DB2 table spaces makes sense for a number of reasons. The most important reason is that many new features of DB2 can only be used with universal table spaces. Examples of such features include clone tables, hash-organized tables, currently committed locking, pending DDL, inline LOBs, XML multi-versioning and ALTER TABLE with DROP COLUMN.

3 Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, 2016-17 Edition, Database Administrators, http://www.bls.gov/ooh/computer-and-information-technology/database-administrators.htm (visited May 10, 2016).

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.

A good general rule of thumb is to use UTS for all new DB2 table spaces and to start converting 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.

CHALLENGE: LARGE OBJECTS FOR STORAGE

Another impactful change to DB2 for z/OS has been the adoption of Large Objects or **LOBs for storing unstructured data**. Although structured data remains the bedrock of the information infrastructure in most organizations, unstructured data is growing in importance. And there is much more unstructured data "out there" than there is structured. Indeed, analysts at IDC estimate that unstructured data accounts for as much as 90 percent of all digital information.[4]

DB2 for z/OS can store unstructured data using Binary LOB (BLOB), Character LOB (CLOB), and Double-Byte Character LOB (DBCLOB) data types, collectively known as LOBs. Using these data types, relational tables can house unstructured data like images and audio and video data. But DB2 for z/OS users were slow to adopt LOBs in their mainframe databases. The primary cause of this hesitation was that the initial implementation of LOBs in DB2 for z/OS was immature. But today DB2 for z/OS has overcome those early deficiencies and there are now tools that can help organizations to exploit and effectively manage DB2 LOBs.

The other new force driving LOB usage is the "big data" movement. Big data is driving organizations to accumulate and analyze more data, and more varied types of data, to gain business insight. A specific example of big data driving the usage of LOBs in DB2 is the JavaScript Object Notation (JSON) support that has been added to DB2. JSON objects are stored in DB2 as BLOBs.

Thus, more organizations are adopting LOB data in their DB2 databases to support unstructured data, for their big data projects, and to store documents and multimedia data. And LOBs are stored and managed differently than your other DB2 data.

First of all, LOBs are larger than your typical column data. As size increases, so do the management concerns, such as lengthy elapsed times to run utilities, performance of accessing the data, and so on.

Unless it is an inline LOB where the entire LOB is stored in the base table, a LOB will require a LOB table space, auxiliary table, and LOB index. When building auxiliary tables and indexes, you do not specify columns the same way that you do for normal tables and indexes. For the auxiliary table, you specify the LOB column and base table, and then DB2 automatically generates the columns needed. For the auxiliary index, you just specify the auxiliary table, and then DB2 implicitly generates the needed index keys. All of this differs from the way in which traditional DB2 data is managed.

Each LOB instance can be up to two gigabytes (GB) per row! Each LOB table space can have as many as 254 different data sets with a DSSIZE from 2 GB to 64 GB each for a total of about 16 terabytes (TB). This is per partition, so if there are 4096 partitions (which is the maximum), then the total size for a single LOB is over 66,000 TB. Managing such large amounts of data can be quite challenging. For example, are you prepared for running utilities on such large table spaces?

Most importantly, because pointers are used to maintain LOBs, errors can occur when there are inconsistencies between the components of the LOB:

1. The ROWID-Version number in the base table row may not be found in the LOB index.
2. There may be entries in the LOB index that are not referenced by any row in the base table.
3. The LOB data itself may not be where the LOB index points to.
4. There may be LOBs in the LOB table space that are not referenced by the LOB index.

CHECK DATA can be used to find errors 1 and 2 (from the list above); CHECK LOB can be used to find errors 3 and 4. But it is possible that CHECK LOB will convert a type 4 error into a type 2 error, so proceed with caution.

4 Bob Violino, Unstructured Data Offers Big Benefits, Information Week, July 23, 2014.

Then there is the issue of LOB index consistency. If the LOB index is inconsistent with the base table data, the LOB data cannot be accessed. There is no direct access to the LOB table space except through the LOB index. If the LOB index is inconsistent with the LOB table space, DB2 will get errors trying to access the LOB data for that row.

It is also important to understand that LOB data can be distributed over many different pages of a LOB table space. DB2 uses a structure of map pages to point to data pages. At the top is the first map page, and it is this page number that is stored in the LOB index. This first map page contains a list of pages, which can be other map pages and data pages. It also contains the total size of the LOB data. If all the data pages are not referenced by map pages or if the map pages are not properly referenced by a higher level map page, LOB data will be lost.

With all of these pointers and structures to maintain, there are a variety of things that can go wrong. To verify that your LOBs are structurally sound you must run a series of DB2 utilities, in the following order:

1. Run CHECK DATA to verify that the ID fields specified in the base table are also found in the LOB index.
2. Run CHECK INDEX to verify that the LOB index is valid.
3. Run CHECK LOB to verify that the internal structure of the LOB table space is sound.

Of course, there are easier ways. Using a modern tool that understands the nuances of LOBs can make a lot of sense, and can enable you to manage them accordingly and appropriately.

CHALLENGE: MODERN STORAGE

There are many other important structural changes that have been made to DB2 for z/OS over the course of the past few releases. One of these changes is the migration to **modern storage**, typically RAID (Redundant Array of Independent Disks) devices.

An array is the combination of two or more physical disk storage devices in a single logical device or multiple logical devices. The basic idea behind RAID is to combine multiple disk devices into an array that is perceived by the system as a single disk drive. RAID offers high availability and rapid failover because the drives are hot-swappable, meaning that a drive can be replaced while the array is up and running.

RAID offers high availability and rapid failover because the drives are hot-swappable, meaning that a drive can be replaced while the array is up and running.

Thus, when you are doing a read or write you are probably doing that to multiple physical disks in the disk array under the covers, instead of doing a read or write to a single 3390 device. The new disk architectures and devices, with concepts like log structured files and cache in the gigabyte sizes, have a noticeable impact on database physical design considerations. Conventional database design rules based on data set placement are becoming less important and can be ignored in most cases.

SMS-managed storage, represents another important storage-related change. With SMS, or storage management subsystems, the system determines data set placement, thereby minimizing DBA work. SMS is a requirement for DB2 10 for z/OS because the DB2 Catalog must be SMS-managed as of DB2 10. Furthermore, any table spaces or indexes with data sets larger than 4 GB require Data Facility Storage Management Subsystems (DFSMS)-managed data sets.

Without DFSMS, the user is responsible for distributing DB2 data sets among disks. This process needs to be reviewed periodically, either when the workload changes, or when the storage server configuration changes. Furthermore, with RAID devices there is no guarantee that your data set placement attempts will actually achieve the separation you request because the RAID device spreads the data across multiple disks behind the scenes in a way that you cannot control.

To achieve a successful implementation, an agreement between the storage administrator and the DB2 administrator is required so that they can together establish an environment that satisfies both their objectives. But they need to let go of the mindset of trying to place your data to separate access types.

Another storage related DB2 issue is compression, and compression is underutilized in most shops. With today's DB2-provided hardware assisted compression, you can achieve high compression rates with excellent performance. As of DB2 9 for z/OS, you can compress indexes, too. Many of the same reasons that would cause you to compress a table space can compel you to consider compressing indexes. Index compression differs significantly from table space compression, even though the end result (a reduction in storage) is the same.

However, more DB2 data generally is being compressed than ever before, and that can impact how we administer and manage table spaces and index spaces.

CHALLENGE: ANALYTICS

With the rise of big data and analytics, more organizations are performing analytics on large data sets. IBM has addressed this need with the **IBM DB2 Analytics Accelerator for z/OS, or IDAA**. IDAA is an appliance that integrates with your IBM® z Systems™ mainframe and DB2 for z/OS. It provides high-speed analytical processing, at times delivering orders of magnitude performance improvement. When installed, the DB2 Optimizer becomes aware of the IDAA and can build access plans to shuttle appropriate workload to IDAA. That means you do not have to change your programs to take advantage of the power of IDAA.

With the rise of big data and analytics, more organizations are performing analytics on large data sets.

But it does mean that there are additional administration tasks, such as configuring the IDAA and setting up the tasks to synchronize data between DB2 for z/OS and the IDAA appliance.

DB2 10 for z/OS introduced **hash-organized tables**. A hash, or hash function, is an algorithm that converts a defined set of data elements into a small number, usually a single integer that can serve as an index to an array or a storage location on disk. Hashes are particularly efficient for single value lookups, such as a primary key lookup, but are inefficient for sequential access.

Temporal data support was added in DB2 10 for z/OS. **With temporal tables, a time period is attached to the data to indicate when it was valid or changed in the database**. A traditional database stores data implied to be valid at the current point-in-time; it does not track the past or future states of the data. Temporal support makes it possible to store different database states and to query the data as of those different states. That means different DDL to support temporal data, as well as different SQL syntax to query it (time travel query).

And DB2 11 for z/OS adds **transparent archiving**, somewhat similar to system time temporal tables. **Transparent archiving makes it easier for DBAs to separate operational data that is actively being used from inactive data that can be archived**. This improves performance by reducing the size of the table space and helps to protect inactive data from inadvertent access.

There have also been new data types introduced over the past few releases, including binary, pureXML, DECFLOAT, TIMESTAMP with varying precision and TIMESTAMP WITH TIME ZONE. Not to mention that Synonyms were deprecated for DB2 11 for z/OS. That means they are still there and functional now, but will soon be removed. And that means you will soon have to find all of the places where Synonyms are used and replace them with Aliases or direct table access.

CHALLENGE: NEW CODING TECHNIQUES

It is not just the DB2 structures that are changing, so are **coding techniques** and SQL. Mainframe application development has changed dramatically during DB2's lifetime. In the beginning, and for a long time thereafter, most DB2 programs were written in COBOL and coded to use static SQL. This meant that the SQL was bound before it was executed and the access paths were etched in stone. And it also meant that the access paths for any SQL in almost any program were readily available to DBAs in PLAN_TABLEs.

Fast forward to the present, and static SQL in COBOL is the exception rather than the rule. Those COBOL programs with static SQL are still there, running in IBM® CICS® or in batch, but new development is not done this way anymore. Today, programmers use IDEs to build code that accesses mainframe DB2 data using distributed data access over the web or from a GUI. Most modern, distributed, application development projects typically rely upon application development frameworks. The two most commonly used frameworks are Microsoft® .NET and Java/J2EE. And these frameworks use dynamic SQL, not static.

An additional contributing force is the adoption of commercial off-the-shelf applications for ERP and CRM like SAP, PeopleSoft, and Siebel. These applications are not tied to a specific DBMS, but are supported by multiple different DBMSs, one of which is DB2 for z/OS. These applications use dynamic SQL because that makes it easier to accomplish multi-vendor DBMS support.

THE BOTTOM LINE

DB2's infrastructure and the structures used to store data in DB2 are changing rapidly. Organizations and their DBAs must be ready to embrace the change to ensure that effective, efficient, and modern DB2 databases and applications are being built. Failure to do so can negatively impact the business and cause you to react when the older ways of doing things are no longer supported. These organizations must similarly embrace new technology that enables them to better manage this new DB2 to ensure the business can capitalize on the big data movement.

FOR MORE INFORMATION

To learn more about harnessing the power of today's DB2, visit **bmc.com/mainframe.**

ABOUT THE AUTHOR

Craig S. Mullins is president and principal consultant with Mullins Consulting, Inc. He has more than three decades of experience in all facets of database systems development and has worked with mainframe DB2 since V1. Mullins is the author of two books: DB2 Developer's Guide and Database Administration: The Complete Guide to DBA Practices & Procedures. Mullins is also an IBM Gold Consultant and IBM Champion for Information Management. www.mullinsconsulting.com