IDUG
The Worldwide DB2 User Community
SOLUTIONS JOURNAL
THE OFFICIAL JOURNAL OF IDUG

DB2 and Storage Management

By Craig S. Mullins

As an IT professional who uses DB2, you know that all database management systems rely on some form persistent storage to maintain data. That means that the DBMS interoperates with operating system files, or data sets. As such, storage management should be a key part of the database operations required of a DBA. Typically database storage means disk devices or subsystems, but it can also mean solid state disk, removable storage, or even trusty "old" tape.

DB2 Storage Basics

At the most basic level it is important to know that DB2 stores its data in VSAM Linear Data Sets (LDS). Each table space and index space you create requires at least one, possibly more, underlying VSAM data sets. DB2 uses VSAM Media Manager for its I/O operations. For every I/O, VSAM Media Manager builds a channel program and sends a request to the I/O supervisor.

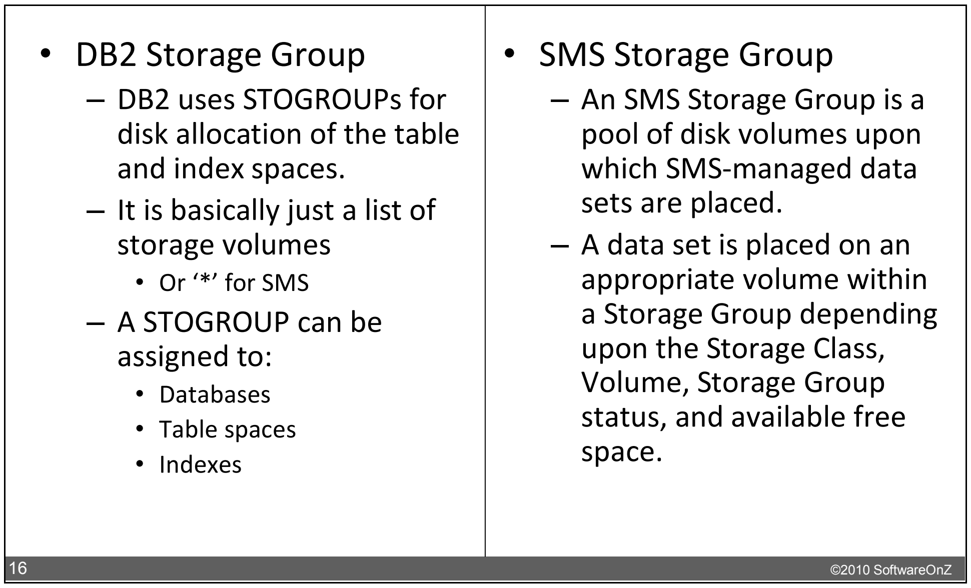
But let's back up a moment. The following items are the core of the storage-related objects and items you will need to know about for DB2 for z/OS:

- DB2 Storage Groups: list of disk volumes
  - DB2 can also work with SMS so you will need to differentiate between DB2 storage groups and SMS storage groups (see Figure 1).
- Table Spaces: stored on disk as at least one VSAM LDS data set
- Indexes: stored on disk (in an index space) as at least one VSAM LDS data set
- System data sets
  - Active Log: stored on disk
  - Archive Logs: stored on disk or tape
  - BSDS: stored on disk
- Image Copy Backups: stored on disk or tape
- Other "stuff"
  - DB2 library data sets
  - Temporary data sets (used by utilities)

So you can see that there are multiple areas within DB2 that require storage and need to be managed. This article will touch upon most of these areas. But back to data set basics for the time being.

You may have noticed that I said that *multiple* data sets may be required… so when does DB2 utilize multiple VSAM data sets for a table space or index? There are three situations where this will arise:

1. When the object is partitioned, each partition will reside in a separate data set.
2. When a data set in a segmented or simple table space reaches its maximum size of 2 GB, DB2 can automatically create a new data set.
3. When the table space is cloned; each clone has its own underlying data set(s).



- DB2 Storage Group
  - DB2 uses STOGROUPs for disk allocation of the table and index spaces.
  - It is basically just a list of storage volumes
    - Or '*' for SMS
  - A STOGROUP can be assigned to:
    - Databases
    - Table spaces
    - Indexes
- SMS Storage Group
  - An SMS Storage Group is a pool of disk volumes upon which SMS-managed data sets are placed.
  - A data set is placed on an appropriate volume within a Storage Group depending upon the Storage Class, Volume, Storage Group status, and available free space.

16 ©2010 SoftwareOnZ

Figure 1. ***DB2 Storage Groups vs. SMS Storage Groups***

To understand how DB2 accommodates these situations we will need to take a look at how DB2 data sets are named. Figure 2 shows the naming convention for DB2 data sets. Although many of you reading this article may be familiar with this naming convention, a quick review is still a good idea. The database name and the page set name (table space or index space name) is part of the data set name. This is one of the reasons that you cannot have more than one table space or index space of the same name in the same database. The "interesting" part of the naming convention (if a naming convention can be interesting at all) comes at the end. We have an instance qualifier and a data set number.

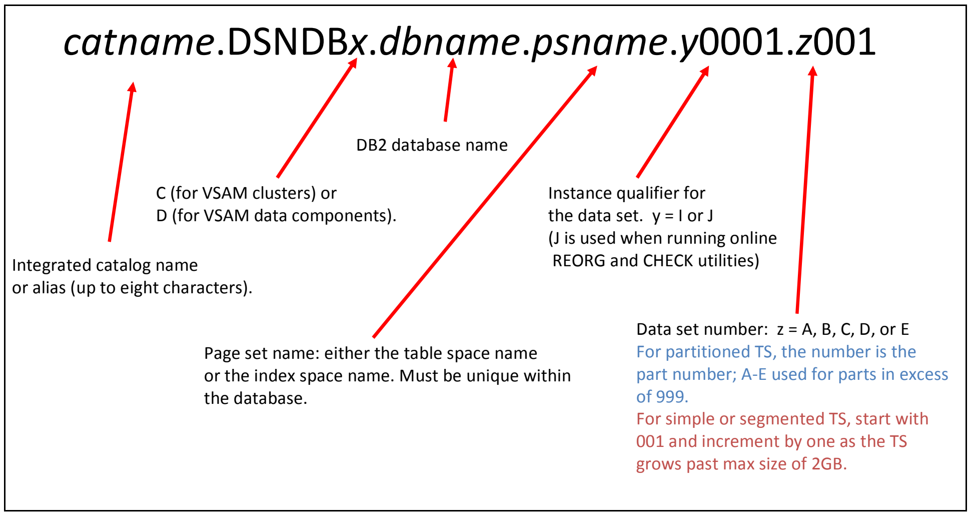

catname.DSNDB*x*.*dbname*.*psname*.*y*0001.*z*001
DB2 database name
C (for VSAM clusters) or D (for VSAM data components).
Instance qualifier for the data set. y = I or J (J is used when running online REORG and CHECK utilities)
Integrated catalog name or alias (up to eight characters).
Data set number: z = A, B, C, D, or E
For partitioned TS, the number is the part number; A-E used for parts in excess of 999.
For simple or segmented TS, start with 001 and increment by one as the TS grows past max size of 2GB.
Page set name: either the table space name or the index space name. Must be unique within the database.

Figure 2. ***DB2 Data Set Naming Convention***

The *instance qualifier* is used when running online REORG and CHECK utilities. For an online utility DB2 uses a shadow data set and will switch from the current to the shadow after running the utility. So DB2 will switch the instance qualifier between I and J when you run online

REORG and CHECK utilities. The numbers after the instance qualifier can change if you use clones. Although this is not the place for a comprehensive discussion of cloning let's skim the surface to understand what happens to this portion of the data set name. Basically, cloning creates a table with the exact same attributes as a table that already exists, except that it has no data. The close is created using the ALTER TABLE SQL statement with the ADD CLONE parameter and the clone table is created in the same table space as the existing table… except in a different VSAM data set. The base table starts with **I0001** in the data set name; the clone will be **I0002**. This can change because the SQL EXCHANGE statement flips the VSAM data sets.

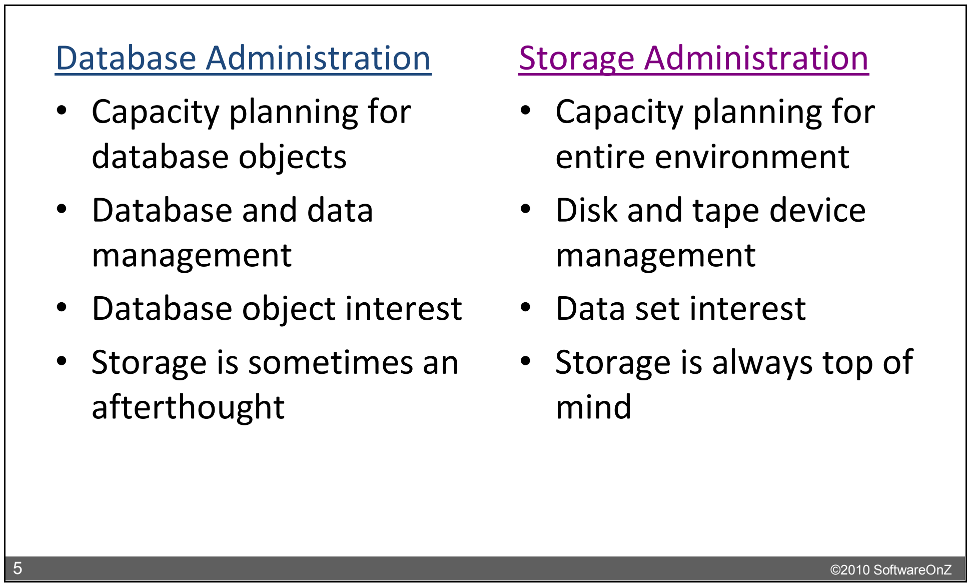
Next we have the *data set number*, which appropriately enough, is used when a page set requires multiple data sets. The z is usually an "A", but it can be A, B, C, D, or E. For partitioned table spaces, the number is the partition number and A-E is used for partitions in excess of 999. So partition 1 would be A001 and partition 1,000 would be B001, and so on. For simple or segmented table spaces, the data set number starts at 001 and is incremented by one as the page set grows past the maximum size of 2GB.

These are the most basic storage "things" that you will need to know as you manage DB2 for z/OS storage.

Important DB2 for z/OS Storage Issues

Although storage management can be an afterthought for the DBA it really shouldn't be. According to Gartner, Inc. the cost of managing storage is 4 to 10 times the initial cost of storage acquisition. And the growth rate for disk storage was 37% for the years 1996 through 2007. So storage issues are vitally important and unless it is managed appropriately it can be very costly. And unmanaged DB2 storage can result in system outages, which is the last thing any DBA wants to have happen, isn't it?

Even so, it is common for storage-related issues to be relegated to the backburner by DBAs. Let's face it, most robust mainframe organizations have an entire unit dedicated to storage management and administration. And the DBA has enough to contend with without adding storage tasks to the list. But DBAs and storage administrators are concerned about different things – and that is the way it should be.



Database Administration	Storage Administration
• Capacity planning for database objects	• Capacity planning for entire environment
• Database and data management	• Disk and tape device management
• Database object interest	• Data set interest
• Storage is sometimes an afterthought	• Storage is always top of mind

5 ©2010 SoftwareOnZ

Figure 3. ***DBA versus Storage Administration***

Refer to Figure 3. The DBA has a database-centric focus, whereas the storage administrator will focus on storage issues for the entire shop, focusing on devices and data sets. But the storage folks are not DB2 experts, nor should they be. Likewise, most DBAs are not storage experts. Making matters worse is that these two groups rarely communicate well.

So there is a gap between Database Administration, Storage Administration and Capacity Planning. Information is available to DBAs from various sources including RUNSTATS, STOSPACE, real-time statistics (RTS), DB2 Catalog, and so on, but the details are scattered all over the place and it can be difficult to gain a complete, accurate, and up-to-date picture. And any historical view into DB2 storage usage has to be managed manually.

Think about your environment for a moment and then reflect on whether or not you can easily answer the following questions:

- Do all of my databases have sufficient allocation to satisfy business requirements?
- Why is DB2 storage growing when our business is not?
- Am I wasting any storage?
- When will more storage be required?
- How much additional storage is needed?
- What needs to be done to align the additional storage with the DBMS?

Without a tool to capture, integrate, and manage information about your DB2 storage infrastructure answering these questions can be quite difficult.

A Little Bit About Modern Disk Arrays

Mainframe disk, or DASD, is usually equated to a 3380 or 3390. In other words, people think of physical hardware devices with a one-to-one relationship between a disk drive and a volume. The logical view is broken down as:

- Track size, or the number of bytes per track.
- Capacity, or the size of the device, in terms of number of tracks or gigabytes.
- Device address, which is a thread onto which I/O operations are serialized by the operating system

But the physical world is not the same as the logical anymore. Today's modern storage architecture uses disk arrays, or RAID. RAID stands for Redundant Array of Independent Disk; an *array* is the combination of two or more physical disk storage devices in a single logical device or multiple logical devices. The array is perceived by the system to be a single disk device. RAID can offer big benefits in terms of availability because the disks are typically hot-swappable, meaning that a drive can be replaced while the array is up & running. There are many levels of RAID technology, each delivering different levels of fault-tolerance and performance. A nice educational overview of the various levels of RAID is offered by Advanced Computer & Network Corporation at http://www.acnc.com/04_00.html. But what about mainframe disk arrays?

The RAMAC Virtual Array (RVA) came first for the mainframe and it was based on *virtual* disks that emulated 3380s and 3390s. The RVA dynamically maps functional volumes to physical drives. This mapping structure is contained in a series of tables stored in the RVA control unit. RVA was OEM'ed from Storage Technology Corp. (STK) which is now part of Oracle.

The ESS (Enterprise Storage System), also known as Shark, followed when the STK OEM agreement expired. The ESS is scalable from 420GB to 55.9 TB. It offers improved performance over RAMAC, especially for prefetch and analytical queries.

And the latest and greatest IBM mainframe disk technology is the DS8000, which employs *virtualized disk*. It adds functionality for storage pool striping, thin provisioning, and quick initialization, among other innovations. Its capacity scales linearly from 1.1 TB up to 192 TB (up to 320 TB with turbo models).

Of course, IBM is not the only game in town for mainframe storage. EMC, Hewlett Packard, Hitachi, and Sun also offer modern disk arrays for the mainframe.

Cache Versus Buffer

Modern disk arrays cache data, but so does DB2. In the old days we used to disable disk cache for DB2, but no longer. There are two DSNZPARMs you should be aware of that can be setup to properly control disk caching for DB2 data sets.

First up is the SEQCACH parameter. The original meaning of this parameter, for 3390 DASD, was whether DB2 I/O should bypass the disk cache, but the meaning is different now because you do not want to bypass the cache on modern storage arrays. There are two options:

- BYPASS - the disk will perform Sequential Detection
- SEQ - creates an explicit Prefetch request; the recommendation is to use this setting for improved performance.

The second DSNZPARM is SEQPRES, which is similar to SEQCACH, but for DB2 LOAD and REORG utilities. If set to YES the Cache is more likely to retain pages for subsequent update, particularly when processing NPIs. The general recommendation is to set to SEQCACH to YES.

Keep in mind, too, that your storage administrators should be aware that DB2 buffering may cause DB2 data to use the disk cache differently than other non-database data sets. But that doesn't mean that DB2 is not benefiting from disk caching.

A Little Bit About DFSMS

DFSMS is IBM's **D**ata **F**acility **S**torage **M**anagement **S**ystem. It offers data management, backup and HSM software from IBM mainframes, combining backup, copy, HSM and device driver routines into a single package. So DFSMS is actually multiple products; it is a suite of data and storage management offerings:

- DFSMSdfp: Data Facility Product - provides the logical and physical input and output for z/OS storage, it keeps track of all data and programs managed within z/OS, and it provides data access both for native z/OS applications and other platforms.
- DFSMSdss is a priced optional feature. It is a DASD data and space management tool for moving and copying data.
- DFSMShsm: Hierarchical Storage Manager - a priced optional feature for managing low-activity and inactive data. It provides backup, recovery, migration, and space management functions.
- DFSMSrmm: Removable Media Manager - a priced optional feature for managing removable media resources (e.g. IBM's Virtual Tape Server).
- DFSMStvs: Transactional VSAM Services – is another priced optional feature that enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently.

But we are not going to delve into all of these various components. What we are most interested in is how DB2 can benefir from DFSMS. Using DFSMS, a DB2 DBA can simplify the interaction of DB2 database creation and storage specification. It can deliver:

- Simplified data allocation
- Improved allocation control

- Improved performance management
- Automated disk space management
- Improved data availability management
- Simplified data movement

DFSMS has the necessary flexibility to support everything DB2 DBAs may want to accomplish in terms of data set placement and design. In this day and age there is no reason to not take advantage of DFSMS for DB2 data sets. However, 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.

SMS Data Classes, Storage Classes, Management Classes, and Storage Groups can be used along with SMS ACS (Automatic Class Selection) routines to set up and automate DB2 data set allocation and placement. SMS Storage Groups contains volumes that satisfy the service requirements of the data sets allocated to them. They can handle more than one type of data. Separate Storage Groups should be defined for production table spaces, active logs, other production data, and non-production data. ACS routines assign data sets to SMS storage classes. For example, indexes can be assigned to to one SMS storage class and table spaces to a different SMS storage class. For DB2 Storage Groups using SMS the volume list is set to '*'.
As of DB2 9, DATACLAS, MGMTCLAS, and STORCLAS can be specified in DB2 Storage Groups, and if so, then the VOLUMES clause can be omitted. If the VOLUMES clause is omitted, the volume selection is controlled by SMS. Keep in mind, though, that when processing the VOLUMES, DATACLAS, MGMTCLAS, or STORCLAS clauses, DB2 does not check the existence of the volumes or classes or determine the types of devices that are identified or if SMS is active. Later, when the storage group allocates data sets, the list of volumes is passed in the specified order to Data Facilities (DFSMSdfp).

Basically, using SMS with DB2 is the sane thing to do these days because the new disk architectures, with concepts like log structured files and with cache in the gigabyte sizes, render conventional database design rules based on data set placement less important. In most cases, placement is not an issue, and when it is, SMS classes and ACS routines can be setup to take care of things.

What About Extents?

Some folks think "With RAID/modern storage devices and new DB2 & z/OS features, extents are no longer anything to worry about." But this is not *exactly* true. For one thing, the latest extent management features only work with SMS-managed data sets, so if you are still using user-managed data sets then all of the old rules apply!

For SMS-managed data set you can have up to 123 extents on each of 59 volumes. So as of z/OS 1.7, the limit is 7,257 extents for a data set instead of the 255 we've been used to for some time. Again though, to enable this requires DFSMS (modify the DFSMS Data Class to set the Extent Constraint Removal to YES).

Extent consolidation also requires SMS-managed STOGROUPs. If a new extent is adjacent to old, they will be merged together automatically. This can result in some extents being larger than the PRIQTY or SECQTY specification(s). Note that this feature was introduced in z/OS 1.5.

OK, so what if everything is SMS-controlled? Then extents don't matter, right? Well, not really. Even then it is possible for extents to impact performance. Each extent on a disk file has different control blocks controlling access. This means that elapsed time can increase if there is heavy insert activity. For other types of processing (read and update) the number of extents really does not impact on performance.

At any rate, things are not like the olden days where you had to regularly monitor extents and clean them up all the time by reorganizing your table spaces and index spaces. Oh, we want to clean those extents up periodically, but storage administrators have other methods of reducing extents that perhaps can be quicker and/or easier, for example.

- DFSMShsm MIGRATE and RECALL functions
- DFSMSdss COPY or DUMP and RESTORE functions
- DEFRAG with the CONSOLIDATE keyword
- Other products: e.g. Real Time Defrag

As of V8, DB2 can allocate sliding scale secondary extents. This is enabled by setting MGEXTSZ DSNZPARM to YES. Note that the default is NO for V8, but changed to YES automatically when you upgrade to DB2 9. With sliding scale extents the extent sizes allocated gradually increase. DB2 uses a sliding scale for secondary extent allocations of table spaces and indexes when:

- You do not specify a value for the SECQTY option of a CREATE TABLESPACE or CREATE INDEX statement
- You specify a value of -1 for the SECQTY option of an ALTER TABLESPACE or ALTER INDEX statement.

Otherwise, DB2 uses the SECQTY value for secondary extent allocations, if one is explicitly specified (and the SECQTY value is larger than the value that is derived from the sliding scale algorithm). If the table space or index space has a SECQTY greater than 0, the primary space allocation of each subsequent data set is the larger of the SECQTY setting and the value that is derived from a sliding scale algorithm.

Without going into all of the gory details, sliding scale extent allocation can help to reduce the number of extents for your Db2 objects as they grow in size over time. And it can help when you do not have a firm understanding of how your data will grow over time.

Another Thing to Think About

Every now and then I hear from a DB2 DBA complaining about index growth. They'll say something like this: "My index keeps growing and growing and taking additional extents, even after deleting data from the base table. What is going on?"

Well, deleted index keys are not physically deleted, but marked as pseudo-deleted. This can cause an index to grow even as the table data remains at relatively the same level. And it can result in a high CPU cost for index scans because DB2 scans all of the entries, even the pseudo-deleted ones. For this reason, you should keep an eye on tables with a lot of delete activity and periodically reorganize their indexes. You can track pseudo-deleted index keys in SYSINDEXPART if using RUSNTATS or SYSINDEXSPACESTATS if using real time statistics. Consider reorganizing these indexes when pct of pseudo-deleted entries is:

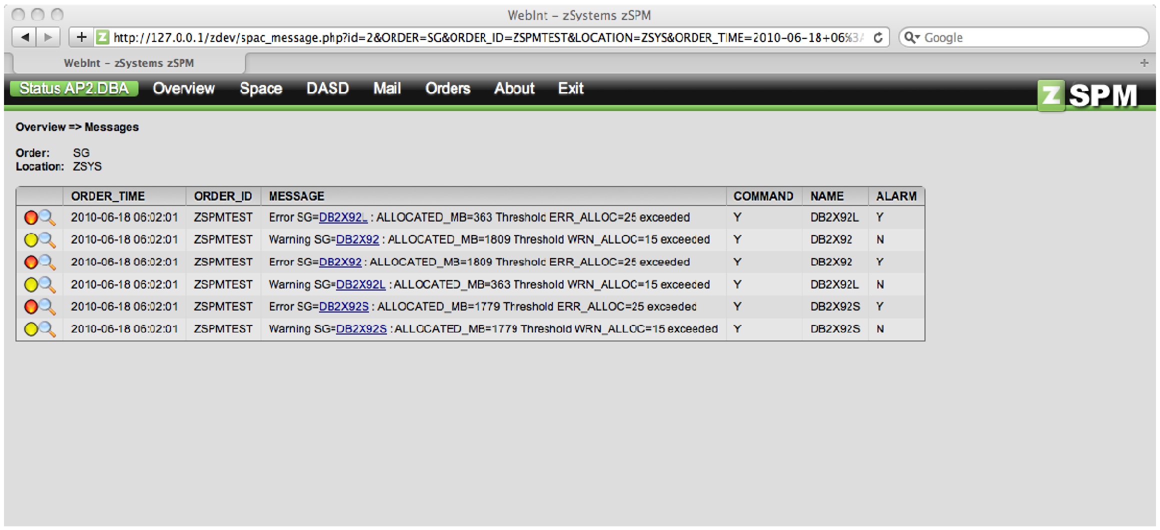
- Greater than 10% for non Data Sharing
- Greater than 5% for Data Sharing.

Best Practices

In general, you should adopt best practices for managing your DB2-related storage. Keep up-to-date on DB2/storage functionality and adopt new practices in the latest releases of DB2.

It is a good idea to perform regular and proactive monitoring. Examples of things you should be tracking include:

1. Space display and monitoring for your entire **DB2** system.
2. Space display and monitoring of individual DB2 **databases**.
3. Space display and monitoring of all of your **table spaces** and **indexes**.
4. Display and monitoring of the **Storage Groups** and the associated volumes of a DB2 system. (Data, Workfile, Image Copies, Logs, Archives, Sort/Work etc.)
5. The ability to display all **VSAM** data sets for all table spaces and indexes (Used, Allocated, Primary and Secondary Quantity, Volumes) and monitoring of the extents for each.
6. Display of the **Page Sets** of table spaces and indexes that reach their maximum size and maximum number of data sets.
7. Tracking of **image copy backup data sets,** including the intelligent HSM migration of same. You should be able to reduce backups by track which are not used for local recovery (to CURRENT), as well as data sets older than the last full image copy (including dual and remote backups).
8. Managing the deletion of **Image copy backup** datasets that are no longer needed because of DROP, DROP/CREATE or MODIFY TABLESPACE ('orphaned', not listed in SYSIBM.SYSCOPY).


WebInt – zSystems zSPM
http://127.0.0.1/zdev/spac_message.php?id=2&ORDER=SG&ORDER_ID=ZSPMTEST&LOCATION=ZSYS&ORDER_TIME=2010-06-18+06%3
Google
WebInt – zSystems zSPM
Status AP2.DBA
Overview
Space
DASD
Mail
Orders
About
Exit
Z SPM

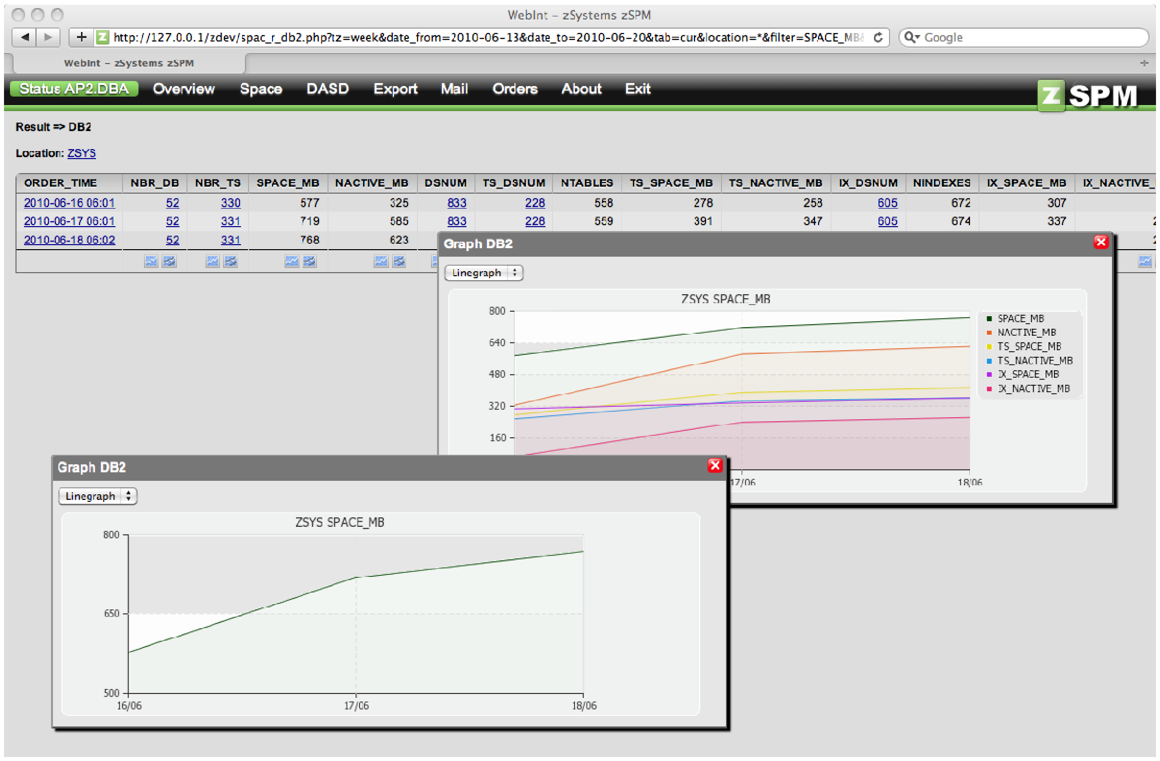
Overview => Messages

Order: SG
Location: ZSYS

	ORDER_TIME	ORDER_ID	MESSAGE	COMMAND	NAME	ALARM
🔴	2010-06-18 06:02:01	ZSPMTEST	Error SG=DB2X92L : ALLOCATED_MB=363 Threshold ERR_ALLOC=25 exceeded	Y	DB2X92L	Y
🟡	2010-06-18 06:02:01	ZSPMTEST	Warning SG=DB2X92 : ALLOCATED_MB=1809 Threshold WRN_ALLOC=15 exceeded	Y	DB2X92	N
🔴	2010-06-18 06:02:01	ZSPMTEST	Error SG=DB2X92 : ALLOCATED_MB=1809 Threshold ERR_ALLOC=25 exceeded	Y	DB2X92	Y
🟡	2010-06-18 06:02:01	ZSPMTEST	Warning SG=DB2X92L : ALLOCATED_MB=363 Threshold WRN_ALLOC=15 exceeded	Y	DB2X92L	N
🔴	2010-06-18 06:02:01	ZSPMTEST	Error SG=DB2X92S : ALLOCATED_MB=1779 Threshold ERR_ALLOC=25 exceeded	Y	DB2X92S	Y
🟡	2010-06-18 06:02:01	ZSPMTEST	Warning SG=DB2X92S : ALLOCATED_MB=1779 Threshold WRN_ALLOC=15 exceeded	Y	DB2X92S	N

Figure 4. ***Automated Storage Alerts***

If possible, build alerts to inform you of problems, shortages, and potential errors. When possible, automate remediation tactics so that the alert tells you what happened, as well as what was done to correct the issue. Tools may be able to assist in automating reaction to shortages, potential errors, superfluous data sets, etc. For example, refer to Figures 4 and 5.



Result => DB2

Location: ZSYS

ORDER_TIME	NBR_DB	NBR_TS	SPACE_MB	NACTIVE_MB	DSNUM	TS_DSNUM	NTABLES	TS_SPACE_MB	TS_NACTIVE_MB	IX_DSNUM	NINDEXES	IX_SPACE_MB	IX_NACTIVE_
2010-06-16 06:01	52	330	577	325	833	228	558	278	253	605	672	307	
2010-06-17 06:01	52	331	719	585	833	228	559	391	347	605	674	337	
2010-06-18 06:02	52	331	768	623									

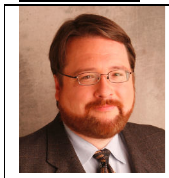
Figure 5. *Tracking the Growth of DB2 Storage Space*

Summary

Finally, there is a people issue that needs to be addressed. The DBA and the storage administrator will need to cooperate in order to facilitate proper database storage. Remember, other types of data that are not stored in the DBMS will need to be saved on disk, too. Databases use storage differently than non-database data. Indexing, partitioning, clustering, and separation of data will cause the database to require more storage, across more drives, and using cache differently than most storage administrators may anticipate.

DBAs need to work storage administrators to make sure that each understands the other domain. And to make sure that s/he has the storage information needed to properly administer DB2. The better the DBA communicates, and the better the relationship is between these two IT professionals, the better your database applications will perform. And that is what it is all really about, right?

Author Bio



Craig S. Mullins is a data management strategist, researcher, and consultant. He is president and principal consultant of Mullins Consulting, Inc., a principal with SoftwareOnZ (a mainframe software distributor), and the publisher/editor for TheDatabaseSite.com. Craig has nearly three decades of experience in all facets of database systems development and has worked with mainframe DB2 since V1.

You may know Craig from his popular books: "DB2 Developer's Guide" (with over 1500 pages of in-depth technical information on DB2 for z/OS) and "Database Administration: The Complete Guide to Practices and Procedures" (the industry's only comprehensive guide to heterogeneous database administration). More information available at www.craigsmullins.com.