IBM Information Management

DB2 Automatic Storage

The Future of Storage Management In DB2 for LUW

Aamer Sachedina, Matt Huras and Kelly Schlamb

e business on demand.

Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

Agenda

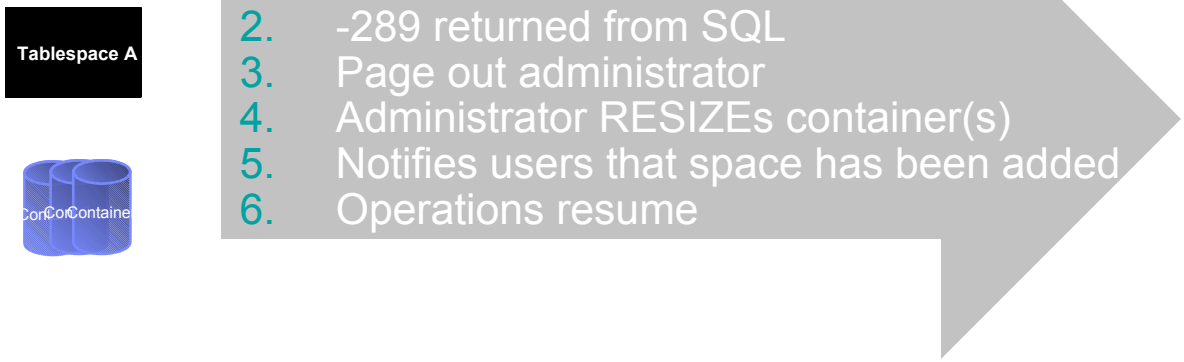
- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

What are Auto-Resize Tablespaces ?

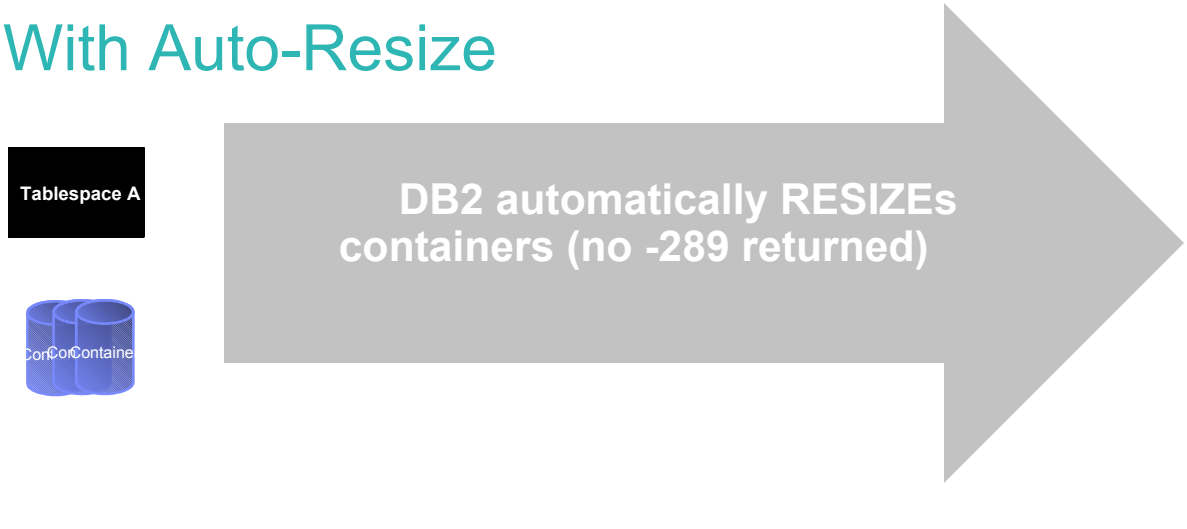
- DMS tablespaces with file containers that automatically extend
  - Introduced in DB2 V8.2.2 (V8 FP9)
  - Applies to DMS file tablespaces only; tablespaces with raw device containers cannot be enabled for auto-resize
- Instead of getting an “unable to allocate new pages” error, DB2 will automatically grow the last range of containers in the tablespace
  - Of course, if the underlying file system is full, auto-growth cannot occur and SQL0289N – unable to allocate new pages – will be returned

Auto-Resize Tablespaces

Without Auto-Resize



With Auto-Resize



Aside : Why DMS File ?

- At a high-level, tradeoffs involved:
  - Outstanding performance of DMS Raw - vs -
  - Very good performance and flexibility of DMS File - vs-
  - Outstanding manageability of SMS
- Technology Trend : significant advances are being made in file systems
  - Eg. Direct/concurrent I/O capabilities of many file systems
  - Performance gap between file systems and raw devices is narrowing
  - Over time, use of raw devices is likely to diminish in favor of file systems
    - Result: Ease of file system management but with the performance of raw
- Idea : get best of both worlds via DMS File
  - Near-raw performance with DIO/CIO
  - Flexibility of DMS functionality
  - Manageability of file systems
  - On-demand growth as in SMS

New CREATE/ALTER TABLESPACE Syntax

- New syntax introduced for CREATE and ALTER TABLESPACE:

```
CREATE TABLESPACE <tsName> MANAGED BY DATABASE
  USING (<listOfFileContainers>)
  [AUTORESIZE {NO|YES}] [INCREASESIZE integer {PERCENT|K|M|G}]
  [MAXSIZE {NONE | integer {K|M|G}}]

ALTER TABLESPACE <tsName>
  [AUTORESIZE {NO|YES}] [INCREASESIZE integer {PERCENT|K|M|G}]
  [MAXSIZE {NONE | integer {K|M|G}}]
```

- Existing syntax is still supported
  - Default is auto-resize disabled
- In multi-partition databases (DPF), values provided are per-partition

New CREATE/ALTER TABLESPACE Syntax (cont.)

- **INCREASESIZE** specifies the amount of space to automatically add to the tablespace when it becomes full
  - Specified as an explicit size (e.g. 64 M) or as a percentage of the size of the tablespace when the growth occurs (e.g. 10 PERCENT)
- **MAXSIZE** specifies the maximum size that the tablespace is allowed to grow to
  - NONE means “unlimited growth” (practically, there are still DB2 limits on tablespace size and space available to the file system(s))
  - Also limits how much space can be added via user-initiated container operations
- If **AUTORESIZE** is enabled (YES) but no values are specified, the defaults are:
  - INCREASESIZE = 32 M , MAXSIZE = NONE (“unlimited”)

Examples

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'TS1' 1000) AUTORESIZE YES
```

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE
  USING (FILE '/dir/TS2/C0' 100 M, FILE 'dir/TS2/C1' 100 M)
  AUTORESIZE YES    INCREASESIZE 50 M    MAXSIZE 1 G
```

```
CREATE TABLESPACE TS3 MANAGED BY DATABASE
  USING (FILE 'D:\TS3' 2000)

ALTER TABLESPACE TS3
  AUTORESIZE YES    INCREASESIZE 50 PERCENT    MAXSIZE NONE
```

```
CREATE TABLESPACE TS4 MANAGED BY DATABASE
  USING (FILE 'TS4' 200 M)    AUTORESIZE YES

ALTER TABLESPACE TS4    AUTORESIZE NO
```

How Growth Occurs

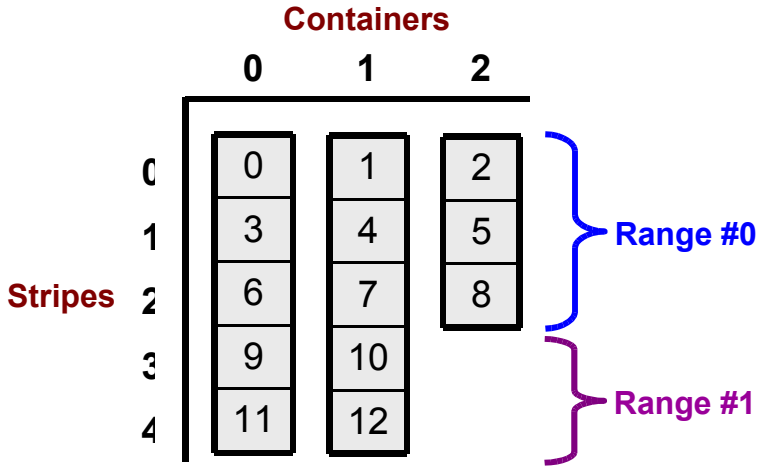
- Table space will auto-extend when it is full and more space is needed
  - Can use fast storage allocation where available (eg. AIX JFS2, Windows)
- Only those containers that are part of the last range in the tablespace map will grow (see next slide for an explanation of ***ranges***)
  - Ensures that a rebalance will never take place as part of an auto-resize
- Auto-growth will stop when any of the following happen:
  - The value specified for MAXSIZE is reached
  - One of the containers in the last range cannot grow any further
    - To preserve the level of striping, DB2 will not automatically extend the other containers in the last range
    - To continue growth, can add space to file system, or a new stripe set

A Quick Primer On Ranges

- Every DMS tablespace has a “map” that describes the logical and physical layout of the tablespace
- A **range** is a region of this map where the striping involves a unique set of containers

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'cont0' 60, FILE 'cont1' 60, FILE 'cont2' 40)
  EXTENTSIZE 10
```

(remember: 1 extent per container is used to hold a container ‘tag’)

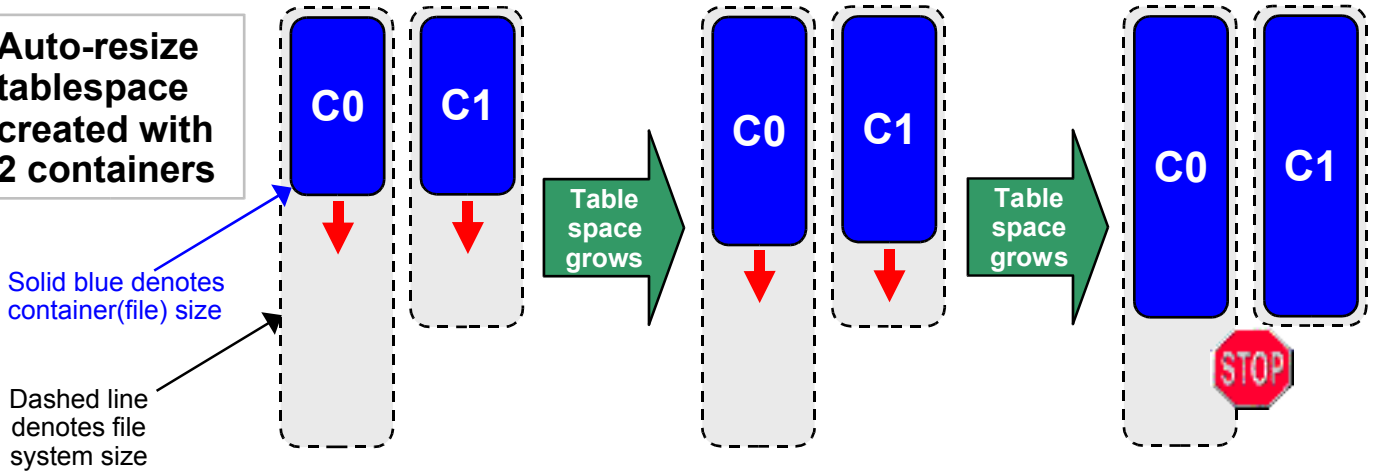


Range Number	Stripe Set	Max Extent	Start Stripe	End Stripe	Containers
0	0	8	0	2	3 (0, 1, 2)
1	0	12	3	4	2 (0, 1)

* Using containers that are of differing sizes is not recommended – this is simply shown as an example

Example of Auto-Growth Stopping

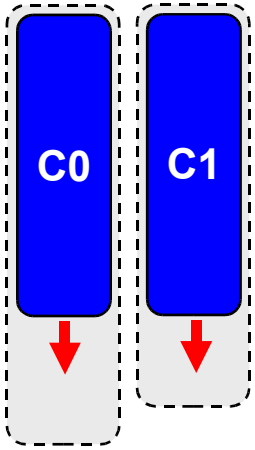
Auto-resize tablespace created with 2 containers



Options to kick start auto-growth again?

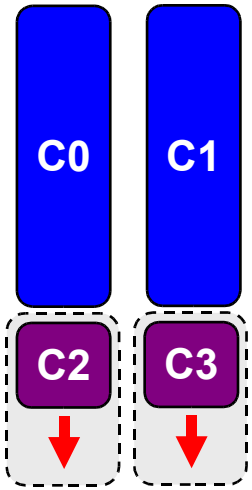
2. Make more room available on the file system holding C1
3. Add a new stripe set (recommended if #1 not possible)
4. Explicitly extend C0 by some amount (reduces striping)

1)



-or-

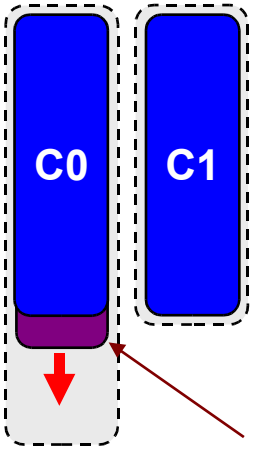
2)



Adding the new stripe set here results in a new range being created in the tables space map. Hence, auto-resize will only extend these new containers from here on.

-or-

3)



Extending C0 results in a new range being created that holds only that one container. Since all of the containers in the last range can grow, auto-resize will resume (on only that container).

Explicit ALTER TABLESPACE …

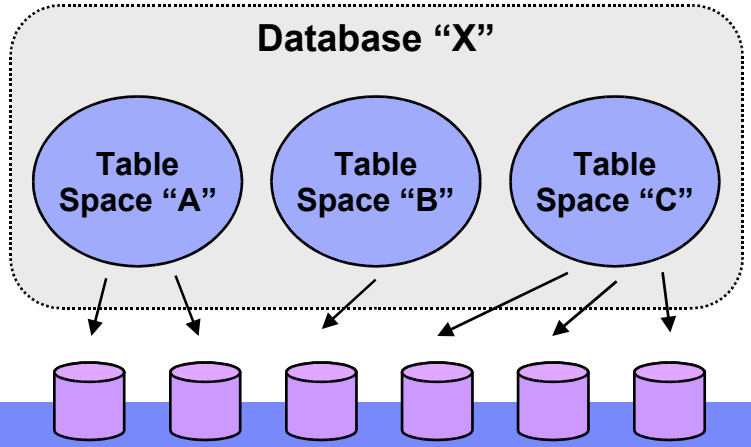
Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

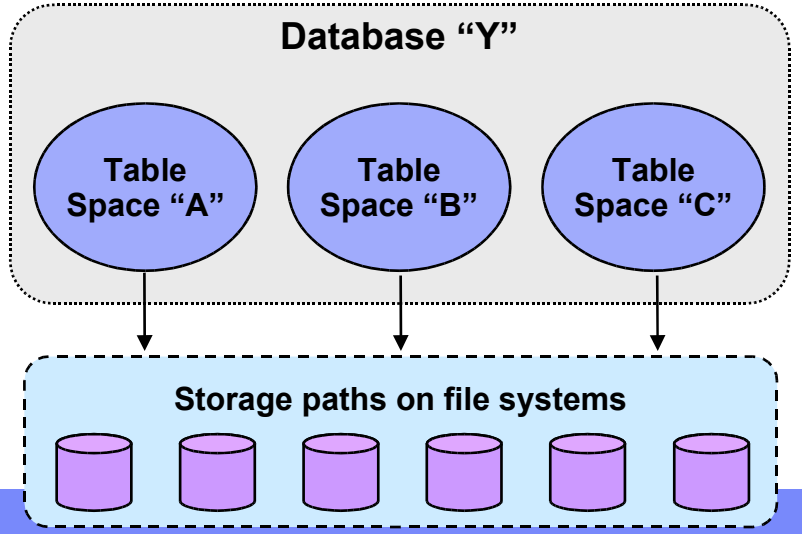
What is Automatic Storage ?

- **New storage management technique where storage for multiple tablespaces is automatically managed at the database level**
  - Multiple tablespaces automatically draw increments of storage from a “database storage pool” on demand
  - Removes need to watch out for disk shortages in each individual tablespace
  - Removes need to manually enlarge containers or add stripe sets
  - Uses DMS infrastructure internally : combines performance benefits of DMS infrastructure with manageability benefits of SMS
- **Tablespaces retain their other properties which can be useful for logical grouping of tables and objects, eg:**
  - Combining logically related tables in the same tablespace so that they can be recovered together
  - Separating logically unrelated tables in different tablespaces so they can be recovered independently
  - Placing the objects of a table (data, index, long) in separate bufferpools

Non-Automatic Storage



Automatic Storage



Automatic Storage Functionality

- Automatic Storage allows you to …
  - Create a database and associate a set of storage paths with it
  - Add storage paths to the database after creation
  - Create AUTOMATIC STORAGE tablespaces
    - No explicit container definitions are provided
    - Containers automatically created across the database storage paths
    - Growth of existing containers and addition of new ones managed by DB2
  - Redefine database storage paths during a database restore

New CREATE DATABASE Syntax

```
CREATE DATABASE database-name
[AT DBPARTITIONNUM | [AUTOMATIC STORAGE {NO | YES}]
[ON path[{,path}...][DBPATH ON path]]
. . .
[CATALOG TABLESPACE tblspace-defn] [USER TABLESPACE tblspace-defn]
[TEMPORARY TABLESPACE tblspace-defn] [WITH "comment-string"]]
[AUTOCONFIGURE [USING config-keyword value [{,config-keyword
value}...]]
[APPLY {DB ONLY | DB AND DBM | NONE}]]

tblspace-defn:
  MANAGED BY { SYSTEM USING ('string' [ {,'string'} ... ] ) |
  DATABASE USING ({FILE | DEVICE} 'string' number-of-pages
  [ {,{FILE | DEVICE} 'string' number-of-pages} ... ]) | AUTOMATIC
       STORAGE}
  [EXTENTSIZE number-of-pages] [PREFETCHSIZE number-of-pages]
  [OVERHEAD number-of-milliseconds] [TRANSFERRATE number-of-
milliseconds]
  [NO FILE SYSTEM CACHING | FILE SYSTEM CACHING]
  [AUTORESIZE {NO | YES}] [INITIALSIZE integer {K|M|G}]
  [INCREASESIZE integer {PERCENT|K|M|G}] [MAXSIZE {NONE | integer
       {K|M|G}}]
```

(Existing (non-automatic storage) behavior is the default if the new syntax is not explicitly used)

New CREATE DATABASE Syntax (cont.)

- Automatic storage can be enabled explicitly using the AUTOMATIC STORAGE YES option:

```
CREATE DATABASE TESTDB1 AUTOMATIC STORAGE YES
CREATE DATABASE TESTDB2 AUTOMATIC STORAGE YES ON /db2data
```

- Or it can be implicitly enabled:
  - By specifying more than one path with the ON option
  - By specifying the DBPATH ON option

```
CREATE DATABASE TESTDB3 ON C:,D:
CREATE DATABASE TESTDB4 /fs1,/fs2,/fs3 DBPATH ON /fs0
```

New CREATE DATABASE Syntax (cont.)

- The paths listed with the ON option are the database's **storage paths**.
  - If ON is not specified then the database has one storage path, determined by database manager configuration parameter *dftdbpath*
- The **database path** is determined in the following order:
  - Path specified with the DBPATH ON option
  - If that is not specified, the first path listed with the ON option
  - If that is not specified, the database manager configuration parameter *dftdbpath*
- By default, SYSCATSPACE, TEMPSPACE1, USERSPACE1 and any subsequently created tablespaces are created as *automatic storage tablespaces* (more on these later), in an automatic storage database
  - Note: can override this; can create any mix of automatic storage and "normal" SMS or DMS

CREATE DATABASE (cont.)

Examples:

```
CREATE DATABASE TESTDB1
```

-Automatic storage enabled: **No**

-Database path: ***dftdbpath***

```
CREATE DATABASE TESTDB2
 ON /testdb2
```

-Automatic storage enabled: **No**

-Database path: **/testdb2**

```
CREATE DATABASE TESTDB3
 AUTOMATIC STORAGE YES
```

-Automatic storage enabled: **Yes**

-Database path: ***dftdbpath***

-Storage path: ***dftdbpath***

```
CREATE DATABASE TESTDB4
 AUTOMATIC STORAGE YES ON /dbdir
```

-Automatic storage enabled: **Yes**

-Database path: **/dbdir**

-Storage path: **/dbdir**

```
CREATE DATABASE TESTDB5
 ON /db2/dir1,/db2/dir2,/db2/dir3
```

-Automatic storage enabled: **Yes**

-Database path: **/db2/dir1**

-Storage paths: **/db2/dir1, /db2/dir2, /db2/dir3**

```
CREATE DATABASE TESTDB6
 ON d:\db2_as1,e:\db2_as2
 DBPATH ON c:
```

-Automatic storage enabled: **Yes**

-Database path: **c:**

-Storage paths: **d:\db2_as1, e:\db2_as2**

New ALTER DATABASE SQL Statement

- Adds storage paths to an existing automatic storage-enabled database
  - Note that these paths may not be used until the existing paths are consumed
- Transactional in nature
  - Log record is written and eligible for replay during a roll forward command
- Syntax:

```
ALTER DATABASE [<dbName>] ADD STORAGE ON '<path1>' [,'<path2>',...]
```

- Examples:

```
ALTER DATABASE ADD STORAGE ON '/dbpath3'
ALTER DATABASE ADD STORAGE ON 'D:\NewPath1', 'E:\NewPath2'
```

Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

What is an Automatic Storage Tablespace ?

- A tablespace that will automatically manage it's storage by drawing storage from the database storage paths
  - Automatically assigns containers to itself upon creation
  - Automatically grows these containers
  - Automatically adds new containers (as new stripe sets) when appropriate
- Selected via new MANAGED BY AUTOMATIC STORAGE clause (or by not specifying a MANAGED BY clause at all)
  - Not really a new tablespace type, more like a new *"storage management style"*
  - Still uses the DMS infrastructure (SMS for temps – more on this later)
- Requires no containers to be specified when the tablespace is created
  - Containers will be defined and allocated by DB2
  - Explicit container operations cannot be performed against the tablespace
- Have all of the other tablespace attributes (extent size, prefetch size, etc.)
- Can only be created in Automatic Storage databases (ie. databases that have storage paths defined)

New CREATE TABLESPACE Syntax

- New syntax introduced for CREATE TABLESPACE:

```
CREATE TABLESPACE <tsName> [MANAGED BY AUTOMATIC STORAGE]
  [INITIALSIZE integer {K|M|G}]
  [AUTORESIZE {NO|YES}] [INCREASESIZE integer {PERCENT|K|M|G}]
  [MAXSIZE {NONE | integer {K|M|G}}]
```

Same as what was shown for auto-resize earlier

- Default initial size is 32 MB and auto-resize is enabled by default
- Examples:

```
CREATE TABLESPACE user1

CREATE TEMPORARY TABLESPACE tempts

CREATE TABLESPACE myts INITIALSIZE 100 M MAXSIZE 1 G

CREATE LARGE TABLESPACE lrgts INITIALSIZE 512 M AUTORESIZE NO

CREATE REGULAR TABLESPACE user2 INITIALSIZE 50 M

CREATE TABLESPACE user4 MANAGED BY DATABASE USING …
```

Container Name Format

<storage path>/<instance>/NODE####/<dbname>/T#######/C#######.<EXT>

<storage path>	A storage path associated with the database
<instance>	The instance under which the database was created
NODE####	The database partition number (always NODE0000)
<dbname>	The name of the database
T#######	The tablespace ID
C#######	The container ID
<EXT>	A three-letter extension based on the type of data being stored: CAT - System catalog tablespace TMP - System temporary tablespace UTM - User temporary tablespace USR - User or regular tablespace LRG - Large tablespace

Examples:

/db2data/path1/kschlamb/NODE0000/TESTDB/T0000002/C0000000.USR
E:\Storage\db2\NODE0000\PRODDB\T0000000\C0000000.CAT

Regular/Large Automatic Storage Tablespaces : Details

- Regular and Large table spaces are created using DMS as the underlying tablespace type
  - With file containers
- Differences between automatic storage and non-automatic storage tablespaces:

Regular/Large Non-auto Storage DMS Tablespaces	Regular/Large Automatic Storage Tablespaces
Containers must be explicitly provided when the tablespace is created.	Containers cannot be provided when the tablespace is created; they will be assigned and allocated automatically by DB2.
Automatic resizing of tablespaces is off (AUTORESIZE NO) by default.	Automatic resizing of tablespaces is on (AUTORESIZE YES) by default.
Container operations can be performed using the ALTER TABLESPACE statement (ADD, DROP, BEGIN NEW STRIPE SET, and so on).	DB2 automatically creates new stripe sets when appropriate. Explicit Container operations cannot be performed because DB2 is in control of space management.
The initial size for the tablespace is implied by the size of the list of containers it is created with.	The initial size for the tablespace can be specified using the INITIALSIZE clause.
A redirected restore operation can be used to redefine the containers associated with the tablespace.	The storage paths for the database can be redefined via database restore and/or relocate DB (more on these later), since storage management occurs at the database level.

Regular/Large Tablespaces Mechanics 1/4

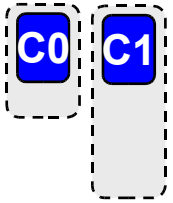
- How does DB2 automatically assign containers to tablespaces ?
  - DB2 will choose to create 0 or 1 container per database storage path
  - Attempts are made to create containers with equal sizes (where possible)
  - Storage paths with a relatively small amount of space will be avoided (where possible)
    - Allows a more consistent set of striping as the tablespace grows (i.e. avoids hitting disk full right away, necessitating a new stripe set)
    - E.g. Three storage paths with 5 MB, 500 MB, and 600 MB of free space respectively. Table space is created with an initial size of 800 MB
      - The tablespace will be created with one container on each of the last two paths (each 400 MB in size)

Regular/Large Tablespaces Mechanics 2/4

- Recently added storage paths are not used until a new stripe set is needed
- As described earlier in the auto-resize section, growth occurs by extending all of the containers in the last range of the map
  - When one of the containers is no longer able to grow, a new stripe set of containers is added
    - This is different than the basic auto-resize case discussed earlier (which would fail at this point with an SQL0289N error)
  - It is at this time that recently added storage paths can be considered for new containers

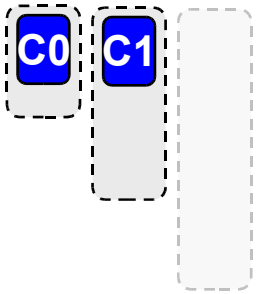
Regular/Large Tablespaces Mechanics 3/4

Two storage paths and a tablespace has a container on each



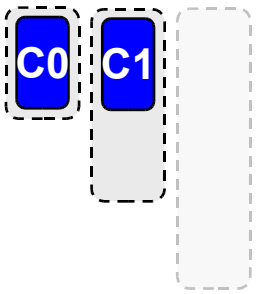
3rd storage path is added

The third storage path is not used by the tablespace yet



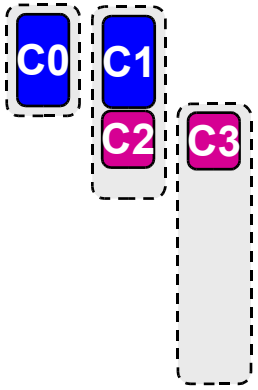
TS grows until C0 can't grow

To continue growing, the tablespace must add a new stripe set



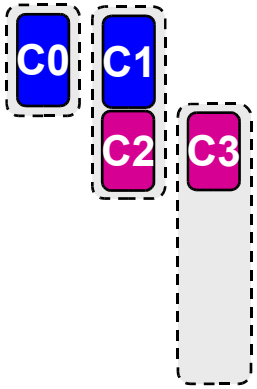
New stripe set added automatically

Now is the recently added storage path utilized



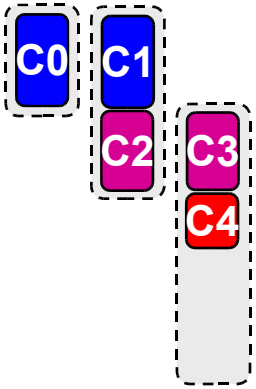
TS grows until C0 can't grow

To continue growing, the tablespace must add a new stripe set



New stripe set added automatically

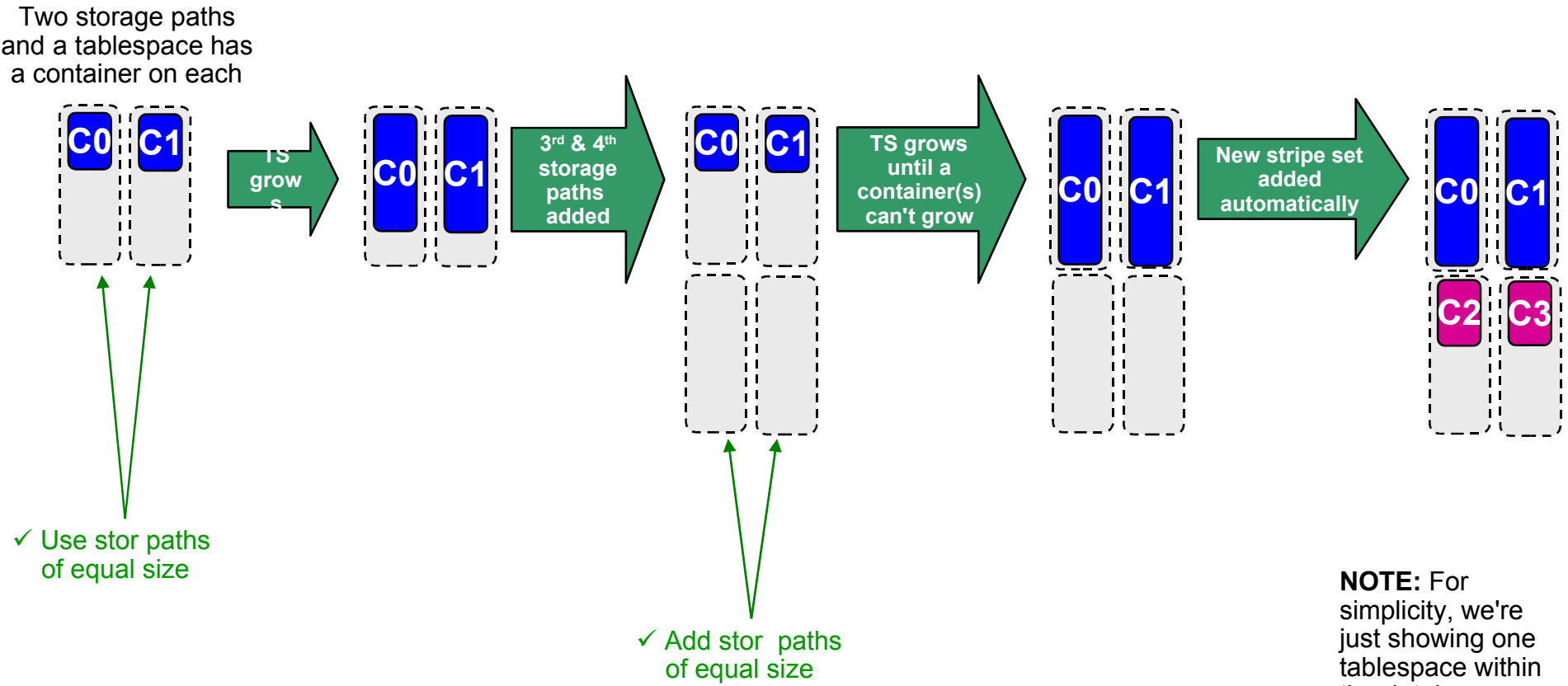
C4 will grow as the tablespace grows from here on



NOTE: For simplicity, we're just showing one tablespace within the database

Regular/Large Tablespaces Mechanics 4/4

A More Typical/Recommended Scenario



NOTE: For simplicity, we're just showing one tablespace within the database

Temporary Automatic Storage Tablespaces : Details

- Temporary automatic tablespaces use SMS as the underlying tablespace type
  - The auto-resize options have no meaning and cannot be specified
    - Remember that SMS is already an auto-extend type of infrastructure (where objects grow by a page or extent at a time)
- Differences between automatic storage and non-automatic storage tablespaces:

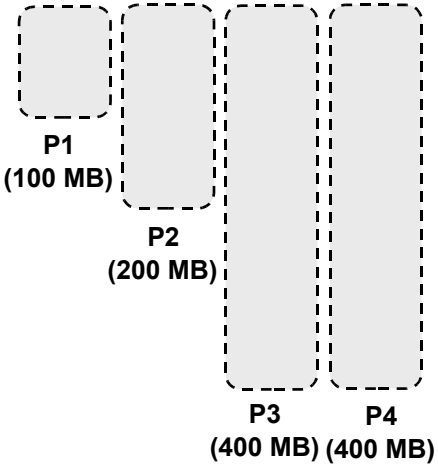
SMS Temporary Non-automatic Storage	Temporary Automatic Storage
Containers must be explicitly provided when the tablespace is created.	Containers cannot be provided when the tablespace is created, they will be assigned and allocated automatically by DB2.
Containers cannot be added after the tablespace has been created.	DB2 will redefine the containers across the storage paths at database startup.
A redirected restore operation can be used to redefine the containers associated with the tablespace.	The storage paths for the database can be redefined via database restore and/or relocate DB (more on these later), since storage management occurs at the database level.

Temporary Tablespaces Mechanics 1/2

- The choice of which storage paths to create the tablespace on is based on the most effective use of space
  - With SMS tablespaces, remember that the smallest container is the limiting factor
    - E.g. two containers with 10 MB and 100 MB – tablespace will be full after using 20 MB (10 MB in each)

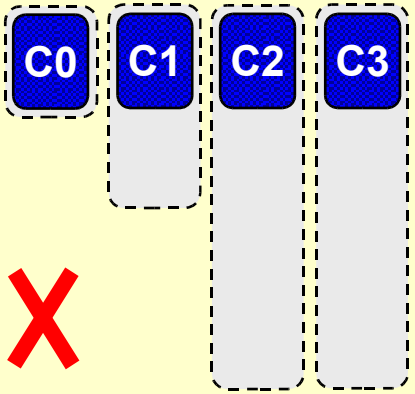
Temporary Tablespaces Mechanics 2/2

Given the following four storage paths, what is the best choice for the temporary tablespace?

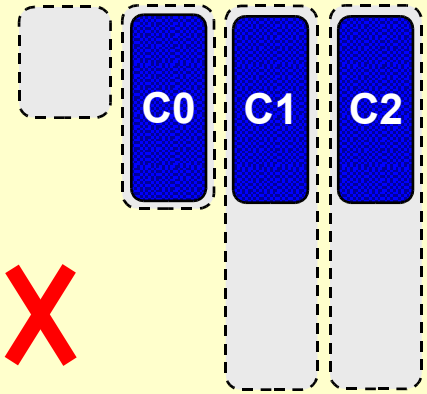


NOTE: This isn't saying that the tablespace will use this much space, just that it possibly could

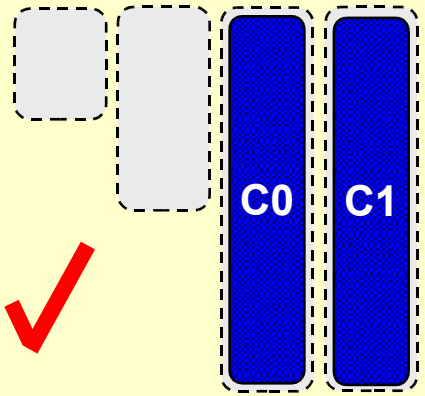
Choice #1: Use P1, P2, P3, P4
Result: Effective size is **400 MB**



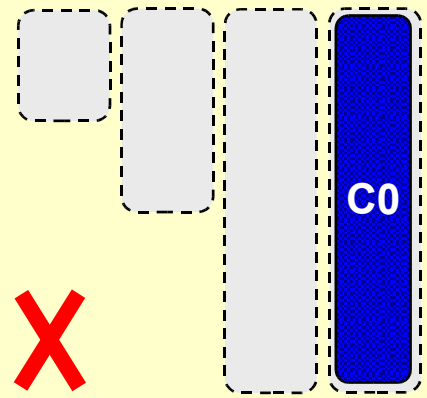
Choice #2: Use P2, P3, P4
Result: Effective size is **600 MB**



Choice #3: Use P3, P4
Result: Effective size is **800 MB**



Choice #4: Use P4
Result: Effective size is **400 MB**



Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

Displaying Automatic Storage Paths

- Storage paths are displayed as part of a DATABASE snapshot
- For databases not enabled for automatic storage:

```
Number of automatic storage paths          = 0
```

- For databases enabled for automatic storage:

```
Number of automatic storage paths          = ##
Automatic storage path                     = <1st path>
Automatic storage path                     = <2nd path>
...
```

- Can access this information through snapshot UDFs as well:

```
select num_db_storage_paths from
       table(snap_get_db('DBNAME', -1)) as dbinfo

select db_storage_path from
       table(snap_get_sto_paths('DBNAME', -1)) as stgpaths
```

Tablespace Snapshot Output

```
Tablespace name                            = TS1
  Tablespace ID                            = 3
  Tablespace Type                          = Database managed space
  Tablespace Content Type                  = Any data
  Tablespace Page size (bytes)             = 4096
  Tablespace Extent size (pages)           = 32
  Automatic Prefetch size enabled          = Yes
  Buffer pool ID currently in use          = 1
  Buffer pool ID next startup              = 1
  Using automatic storage                  = Yes
  Auto-resize enabled                      = Yes
  File system caching                      = Yes
  Tablespace State                         = 0x'00000000'
  Tablespace Prefetch size (pages)         = 32
  Total number of pages                    = 4608
  Number of usable pages                   = 4576
  Number of used pages                     = 4512
  Number of pending free pages             = 0
  Number of free pages                     = 64
  High water mark (pages)                  = 4512
  Initial tablespace size (bytes)          = 1048576
  Current tablespace size (bytes)          = 18874368
  Maximum tablespace size (bytes)          = 104857600
  Increase size (bytes)                    = 1048576
  Time of last successful resize           = 02/14/2005 15:32:17.355772
  Last resize attempt failed               = No
  Rebalancer Mode                          = No Rebalancing
  Minimum Recovery Time                    = 02/14/2005 15:31:57.030410
  Number of quiescers                      = 0
  Number of containers                     = 1
  …
```

New fields that are applicable to auto-resize and/or automatic storage tablespaces.

Note: these are also accessible via SQL via the snap_get_tbsp() and snap_get_tbsp_part() UDFs.

Tablespace Snapshot Output (cont.)

- New elements are accessible through the snapshot UDFs as well:

```
select tbsp_name,
       tbsp_id,
       tbsp_using_auto_storage,
       tbsp_auto_resize_enabled
  from table(snap_get_tbsp('TESTDB', -1)) as tbsinfo

select tbsp_name,
       tbsp_id,
       dbpartitionnum,
       tbsp_initial_size,
       tbsp_current_size,
       tbsp_max_size,
       tbsp_increase_size,
       tbsp_increase_size_percent,
       tbsp_last_resize_time,
       tbsp_last_resize_failed
  from table(snap_get_tbsp_part('TESTDB', -1)) as tbspartinfo
```

New Automatic Storage Health Indicators

Automatic Storage

- Database automatic storage utilization (db.db_auto_storage_util)
  - Tracks free space in the database storage path(s)
  - Calculation: (db.auto_storage_used / db.auto_storage_total) * 100

Auto-Resize DMS File Tablespaces

- Table space automatic resize status (ts.ts_auto_resize_status)
  - Tracks whether or not a tablespace has failed to resize automatically
  - Values: Normal, Resize failed
- Automatic resize tablespace utilization (ts.ts_util_auto_resize)
  - Tracks the consumption of storage and free space for each auto-resize tablespace on which a maximum size has been defined

Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

Backup & Restore

- The syntax and behavior of the BACKUP command has not changed
- The syntax of the RESTORE command has changed to allow the redefinition of storage paths:

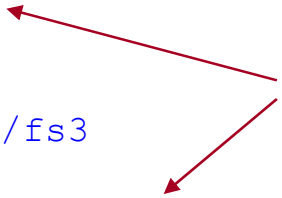
```
RESTORE DATABASE source-database-alias { restore-options | CONTINUE | ABORT }

restore-options:
  [USER username [USING password]] [TABLESPACE [ONLINE] |
  TABLESPACE (tblspace-name [ {,tblspace-name} ... ]) [ONLINE] |
  HISTORY FILE [ONLINE] | LOGS [ONLINE] | COMPRESSION LIBRARY [ONLINE]]
  [INCREMENTAL [AUTOMATIC | ABORT]] [USE {TSM | XBSA} [OPEN num-sess
SESSIONS]
  [OPTIONS {options-string | options-filename}] |
  FROM dir/dev [{,dir/dev} ... ] | LOAD shared-lib [OPEN num-sess SESSIONS]
  [OPTIONS {options-string | options-filename}]] [TAKEN AT date-time]
  [[TO target-directory] | [ON path[{,path}...][DBPATH ON path]]
  [INTO target-database-alias] [LOGTARGET directory]
  [NEWLOGPATH directory] [WITH num-buff BUFFERS] [BUFFER buffer-size]
  [DLREPORT file-name] [REPLACE HISTORY FILE] [REPLACE EXISTING] [REDIRECT]
  [PARALLELISM n] [COMPRLIB lib-name] [COMPROPTS options-string]
  [WITHOUT ROLLING FORWARD] [WITHOUT DATALINK] [WITHOUT PROMPTING]
```

Restore : Rules for Determining DBPATH and Storage Paths

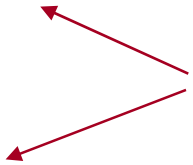
- RESTORE will determine the database path and storage paths as follows
  - The **database path** (which is where DB2 stores various control files for the database)
    - If the TO clause or the DBPATH ON clause is specified: it indicates the database path
    - Else, if the ON clause is used, then the first path listed in the ON clause is used as the database path
    - If none of the TO, ON, or DBPATH ON clauses are specified, the dftdbpath database manager configuration parameter determines the database path
    - *However, if a database with the same name exists on disk, the above is all ignored and the database is restored into the same database path as the existing database* .
  - The **storage paths** (where DB2 creates automatic storage tablespace containers)
    - If the ON clause **is** specified, all of the paths listed are considered storage paths, and these paths are used instead of the ones stored within the backup image.
    - If the ON clause **is not** specified, no change is made to the storage paths (the storage paths stored within the backup image are maintained).

Restore (cont.)

```
CREATE DATABASE TESTDB ON /fs1, /fs2, /fs3 DBPATH ON /fs4
{Database is then backed up and eventually dropped}

RESTORE DATABASE TESTDB
 Database path: dftdbpath
 Storage paths: /fs1, /fs2, /fs3

RESTORE DATABASE TESTDB TO /newfs1
 Database path: /newfs1
 Storage paths: /fs1, /fs2, /fs3

RESTORE DATABASE TESTDB ON /newfs1, /newfs2
 Database path: /newfs1
 Storage paths: /newfs1, /newfs2

RESTORE DATABASE TESTDB ON /newfs1, /newfs2 DBPATH ON /newfs3
 Database path: /newfs3
 Storage paths: /newfs1, newfs2
```



Storage paths are not redefined



Storage paths are redefined

If the database wasn't dropped then the database path would remain on /fs4, regardless of what was specified on the RESTORE command

Rollforward

- If the storage paths associated with a database are changed during a restore:
  - Log records associated with ADD STORAGE are **not** replayed during a subsequent rollforward

    (The assumption is that you are choosing a final storage configuration for the database)

- If the storage paths associated with a database are **not** changed during a restore:
  - Log records associated with ADD STORAGE are replayed

    (The assumption is that after you restore and rollforward you wish the database to be in the same state it was in prior to the restore)

db2relocatedb

- Can be used to change the database name, database path, and the location of tablespace containers
  - Can also be done using backup/restore
  - Benefit of the tool is speed (can use OS/file system tools to do the moving)
- Changes for automatic storage databases:
  - CONT_PATH cannot be specified for an automatic storage tablespace
  - STORAGE_PATH can be used to change a storage path

```
CREATE DATABASE TESTDB ON
  /DIR1,/DIR2,/DIR3

{move /DIR1->/NEWDIR1}
{move /DIR2->/NEWDIR2}
{move /DIR3->/NEWDIR3}

db2relocatedb -f config.txt
```

config.txt

```
DB_NAME=TESTDB
INSTANCE=db2inst
DB_PATH=/DIR1,/NEWDIR1
STORAGE_PATH=/DIR1,/NEWDIR1
STORAGE_PATH=/DIR2,/NEWDIR2
STORAGE_PATH=/DIR3,/NEWDIR3
```

Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- Hints, Tips, Best Practices
  - Monitoring
  - Recovery
  - Migration
  - Misc Hints & Tips

Using New Functionality On Existing Databases

- Auto-resize can be enabled for existing DMS FILE tablespaces, regardless of when they were created
  - ALTER TABLESPACE <tsName> AUTORESIZE YES
  - Can only be enabled if all of the containers are file (no raw devices)
  - For tablespaces with raw device containers you can do the following:
    - Backup the database (or individual tablespaces)
    - Do a redirected restore, specifying a list of file containers (instead of raw device containers)
    - Once the database is online, use ALTER TABLESPACE to enable it
- Currently, Automatic Storage can only be invoked on new databases

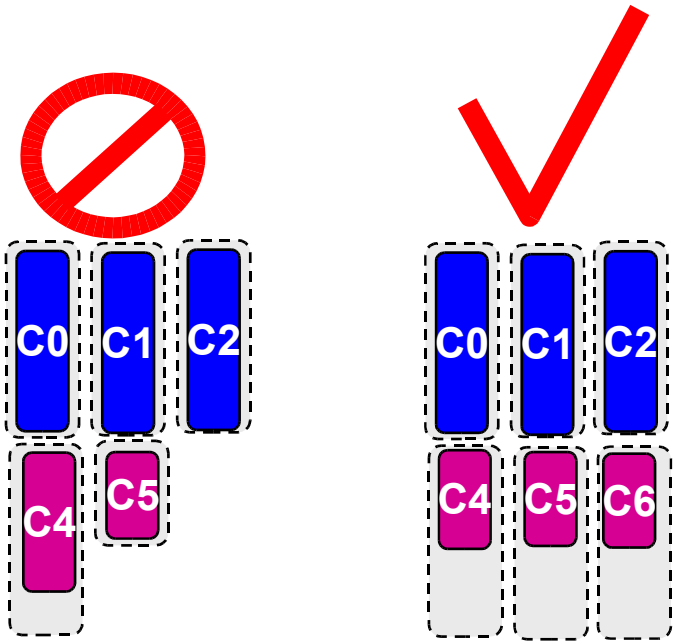
Moving Back To A Pre-V8.2.2 Level of DB2

- Auto-resize tablespaces can be used on older levels of DB2 provided that auto-resize is disabled first
  - ALTER TABLESPACE <tsName> AUTORESIZE NO
  - If not disabled then the database will be unusable
- Automatic storage databases are not usable on pre-V8.2.2. levels of DB2
  - Automatic storage databases have a significant amount of meta-data stored within them that is not recognized on older levels of DB2
  - Includes moving an instance back and connecting, or trying to restore a database backup (both will fail with errors)

Agenda

- New Concepts
  - Auto-Resize Tablespaces
  - Automatic Storage
    - Automatic Storage Tablespaces
- **Hints, Tips, Best Practices**
  - Monitoring
  - Recovery
  - Migration
  - Misc Best Practices, Hints & Tips

Best Practices

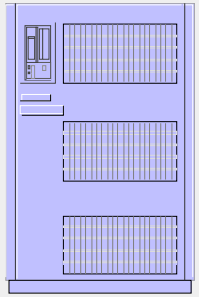


- When adding new containers to an auto-resize DMS tablespace
  - Add them via a new stripe set
  - Add containers that are the same size
- When defining/adding database storage paths, try to ensure the file systems associated with the paths are approximately equal in capacity
- Use the DBPATH clause to keep the database directory control data on different storage than the storage paths, and avoid using the file systems associated with the database storage paths for anything but DB2 tablespace data
  - Depending on the non-DB2 activity that is occurring, it can cause strange growth patterns for the automatic storage tablespaces
  - When determining free space on a storage path, DB2 will not use the last nn MB of space if it shares a file system with the following:
    - The database path (32 MB)
    - The system directory (Windows only – 64 MB)

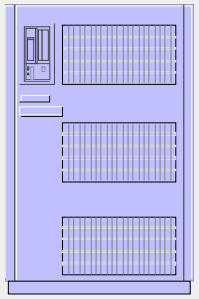
Automatic Storage on DPF

- Storage paths are, by default, homogeneous across database partitions :

2 node DPF instance "inst"



NODE0000



NODE0001

```
CREATE DATABASE mydb
 ON \path1,\path2,DBPATH ON \db1
```

-Automatic storage enabled: **Yes**

-On Database Partition 1 :

Storage paths: **/path1/inst/NODE0000/mydb**
/path2/inst/NODE0000/mydb

Database path: **/db1/inst/NODE0000/SQL00000**

-On Database Partition 2 :

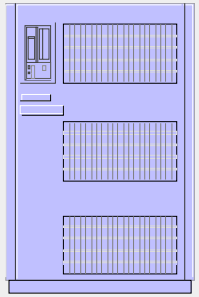
Storage paths: **/path1/inst/NODE0001/mydb**
/path2/inst/NODE0001/mydb

Database path: **/db1/inst/NODE0001/SQL00000**

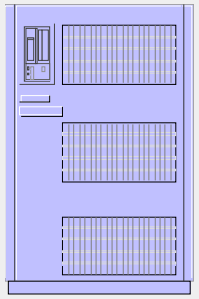
Automatic Storage on DPF

- If desired, non-homogeneous setups are possible:

2 node DPF instance "inst"



NODE0000



NODE0001

```
CREATE DATABASE mydb
 ON \p$N,\p$N,DBPATH ON \db1
```

-Automatic storage enabled: **Yes**

-On Database Partition 0 :

Storage paths: **/p0/inst/NODE0000/mydb**

/p0/inst/NODE0000/mydb

Database path: **/db1/inst/NODE0000/SQL00000**

-On Database Partition 1 :

Storage paths: **/p1/inst/NODE0001/mydb**

/p1/inst/NODE0001/mydb

Database path: **/db1/inst/NODE0001/SQL00000**

Automatic Storage by Default

Examples:

```
CREATE DATABASE TESTDB1
```

-Automatic storage enabled: ~~No~~ YES

-Database path: ***dftdbpath***

-Storage path: ***dftdbpath***

```
CREATE DATABASE TESTDB3
 AUTOMATIC STORAGE NO
```

-Automatic storage enabled: **No**

-Database path: ***dftdbpath***

```
CREATE DATABASE TESTDB6
 ON D:\DB2_AS1,E:\DB2_AS2
 DBPATH ON C:
```

-Automatic storage enabled: **Yes**

-Database path: **C:**

-Storage paths: **D:\DB2_AS1, E:\DB2_AS2**

```
CREATE DATABASE TESTDB1
```

-Automatic storage enabled: ~~No~~ YES

-Database path: ***dftdbpath***

-Storage path: ***dftdbpath***

```
CREATE TABLESPACE MYTS
```

-Automatic storage tablespace automatically drawing storage from dftdbpath

```
CREATE TABLESPACE MYTS2
MANAGED BY DATABASE USING
(FILE 'a' 40000)
```

- Of course, any scripts or procedures that explicitly create non-AS tablespaces will still work and will still create non-AS tablespaces

And Finally … Sorting our your Tablespace Options

Tablespace Option	Single Point of Storage Mgt for Multiple Table-spaces ?	Auto Container Growth ?	Auto Stripe Set Addition ?	File System Buffering ?	Allows Different Tablespaces & Bufferpools for a Table's Data, Index, & Long Data ?	Underlying Infrastructure
SMS	No	Yes	No	Selectable (DIO/CIO)	No	SMS
DMS File	No	No	No	Selectable (DIO/CIO)	Yes	DMS
DMS RAW	No	No	No	No	Yes	DMS
DMS File Auto-Resize	No	Yes	No	Selectable (DIO/CIO)	Yes	DMS
Automatic Storage	Yes	Yes	Yes	Selectable (DIO/CIO)	Yes	DMS (perm) SMS (temp)