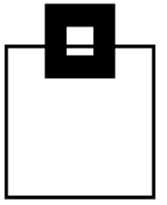
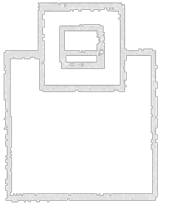
UTS – Past, Present, Future

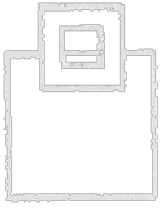
Ulf Heinrich, SEGUS Inc

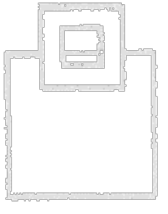
Agenda



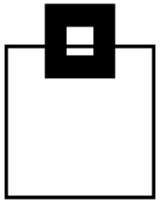
- Tablespaces – A brief history
- Universal Tablespaces – A brief history
- Present UTS situation
- Future UTS situation
- Q&A



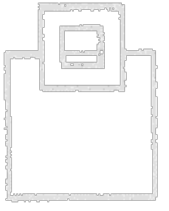


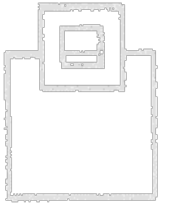


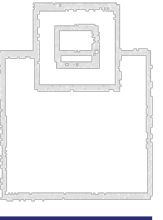
Agenda



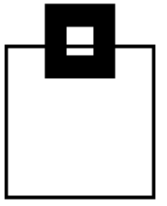
- Tablespaces – A brief history
- Universal Tablespaces – A brief history
- Present UTS situation
- Future UTS situation
- Q&A



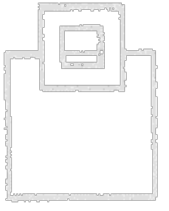




Tablespaces – A brief history

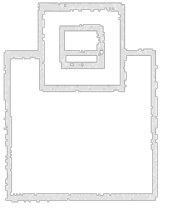


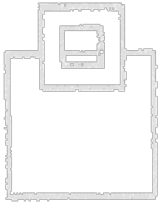
When DB2 was released it came with two different ways of storing data:



1) Non-partitioned
2) Partitioned

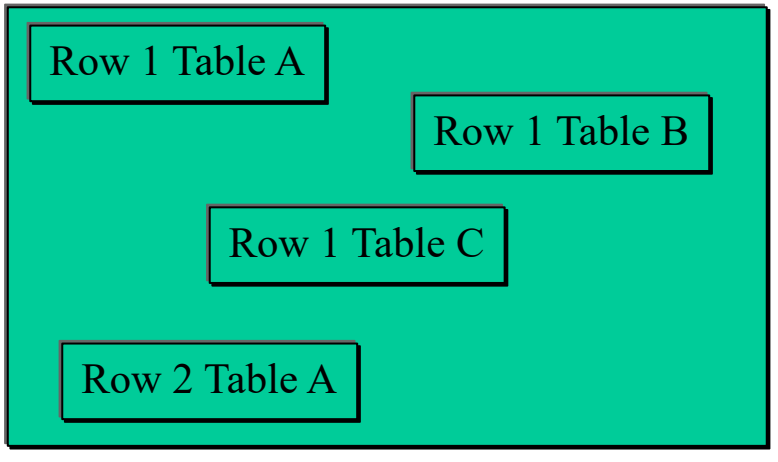
That was it… so simple in those days (excuse the pun…)





Tablespaces – A brief history

Non-partitioned:

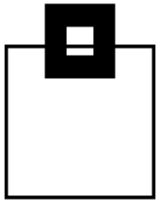


One Page of the tablespace

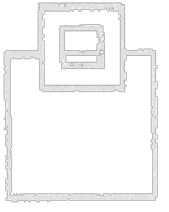
Can contain many tables, all rows are stored anywhere with pages where they fit.

Nice and easy to code but not so good for performance!

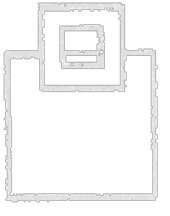
Tablespaces – A brief history

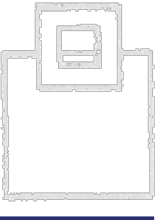


Non-partitioned:

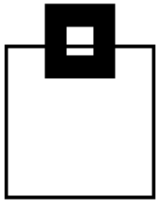


Each underlying VSAM dataset could be 2GB and, whenever needed, DB2 would allocate a new one up to 32 giving a maximum size of 64GB.

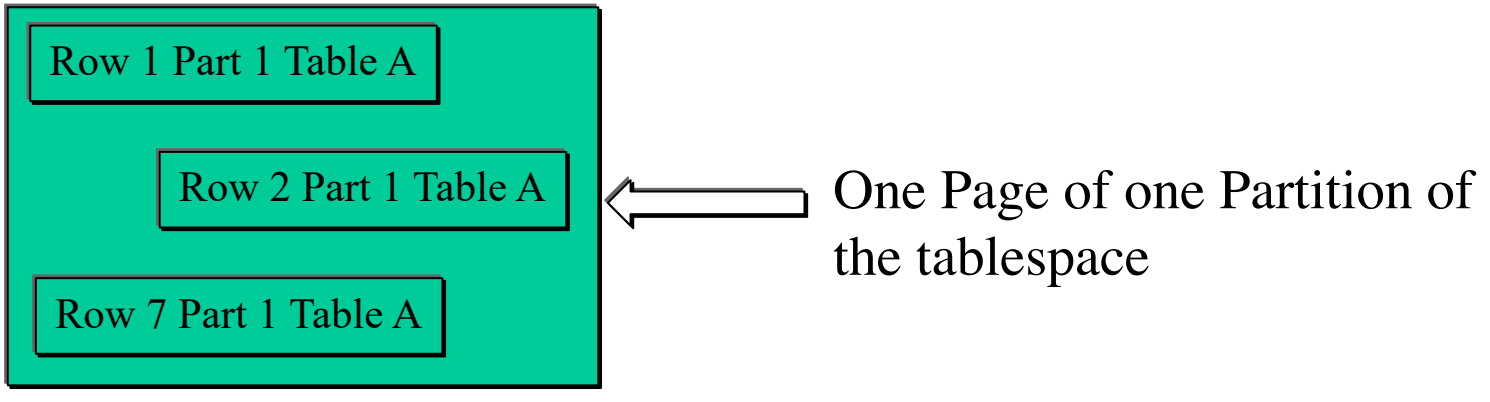


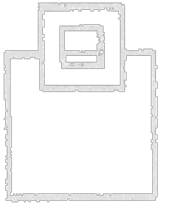


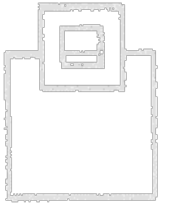
Tablespaces – A brief history



Partitioned:

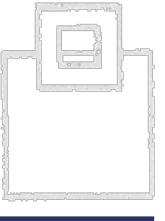






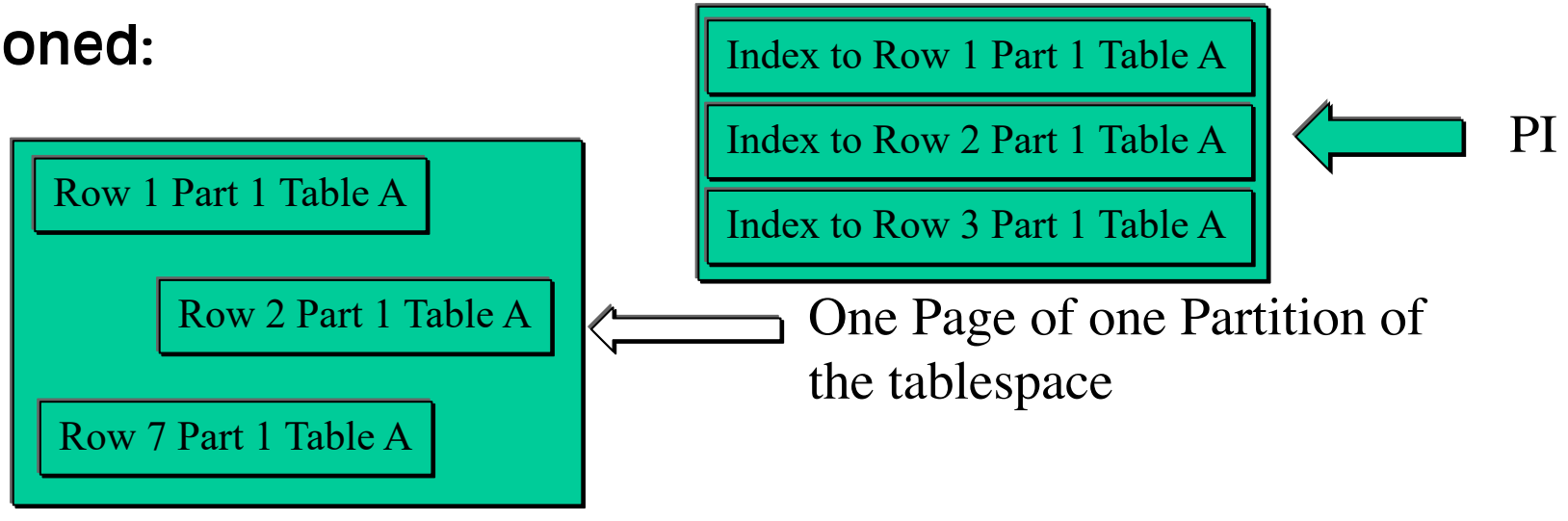
Can only contain one table, all rows are stored anywhere with pages where they fit in the pre-defined partition.

Enabled extremely large tables (at least back in the day!) with up to 16 partitions each of which could be 4GB or 64 partitions each of which could be 1GB – Total of 64GB in all cases.



Tablespaces – A brief history

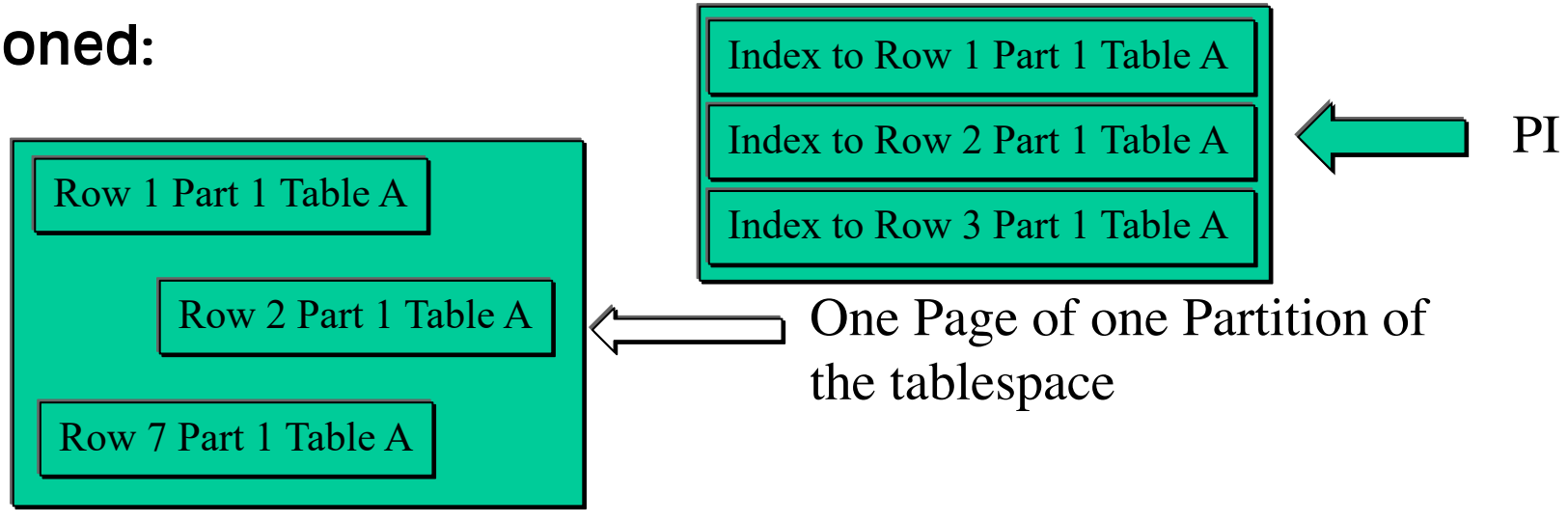
Partitioned:



Now to actually get a Partitioned Tablespace you had to create a Partitioned Index (PI) as well. DB2 then used this to actually control the access. This was naturally an added difficulty especially when your business wishes changed and you required a new partition or a new partitioning definition.

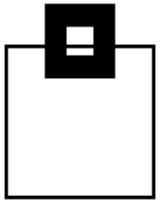
Tablespaces – A brief history

Partitioned:

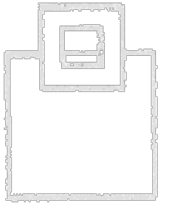


The size of the PI was also a problem as it was dependent on the tablespace and indexspace definition. This was not so good! The inherent problem of tying these things together would not be solved for many years…

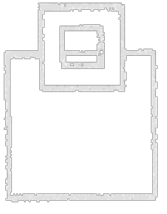
Tablespaces – A brief history

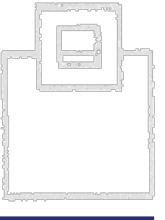


Indexes came in two types: Unique and non-Unique – For this presentation, Indexes are not really important until much later.



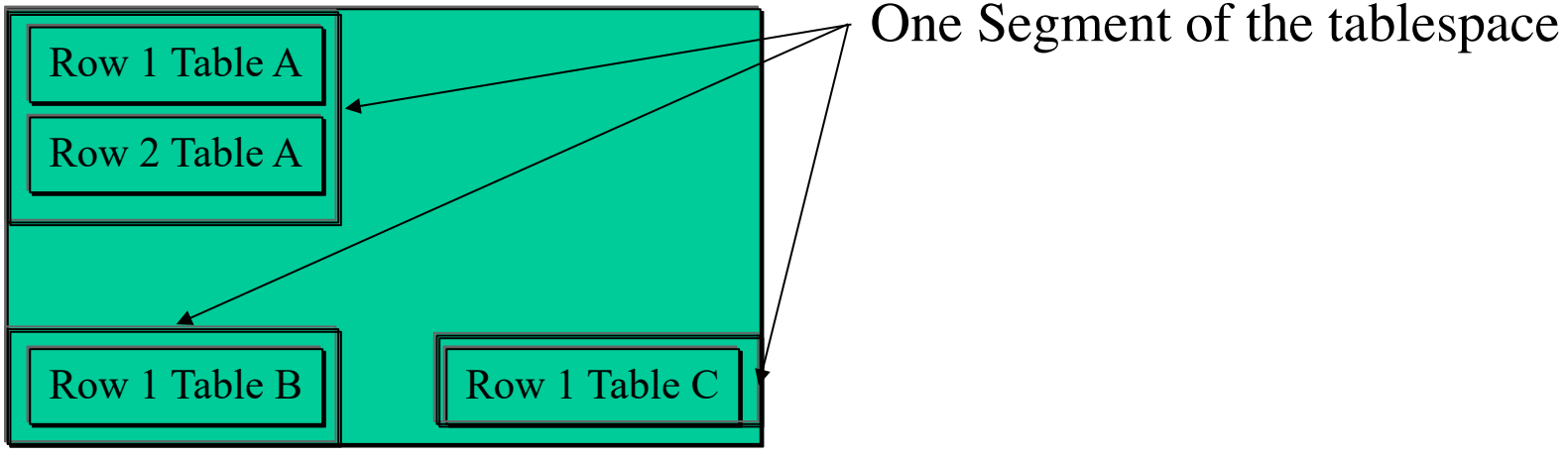
Their sizes were also limited to various degrees on the type of table. However, you were limited to a “number” of VSAM datasets for a non-partitioned index (NPI) and a “size” for a partitioned index in one VSAM dataset. These “number” and “size” were highly correlated to the page size, index page size and/or partition count of the underlying tablespace.





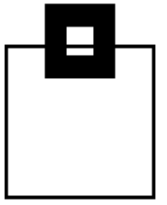
Tablespaces – A brief history

In DB2 2.1 the Segmented Non-partitioned tablespace appeared:



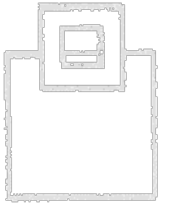
Can contain many tables, all rows of one table are stored in Segments. Segments could be from 4 pages to 64 pages in size. Greatly speeded up processing especially for mass delete and insert. Still limited to only 32 VSAM datasets…

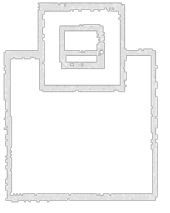
Tablespaces – A brief history

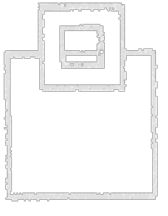


In DB2 5.1 Partitioned spaces got an upgrade with the MEMBER CLUSTER keyword and the LARGE syntax.

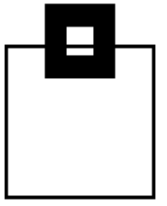
This enabled up to 254 Partitions each of 4GB giving 1016GB as maximum size.







Tablespaces – A brief history

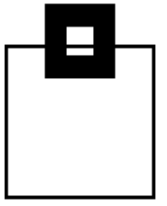


In DB2 6.1 Partitioned spaces got a major upgrade with the DSSIZE keyword. Interestingly enough, the LARGE clause was also then deprecated!

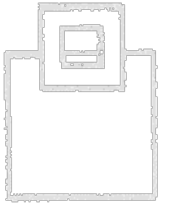
This enabled up to 254 Partitions each of 64GB giving 16256GB or 16TB as maximum size.

Of course, you needed to have the double extended setup in DFSMS 1.5 or higher to get larger than 4GB datasets (extended format & extended addressability).

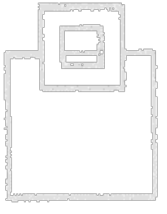
Tablespaces – A brief history



In DB2 8.1 Partitioned spaces got another major upgrade with the ability to have 4096 partitions!

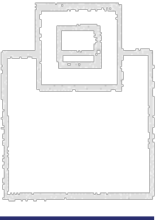


This enabled, if you used 32KB Page size and a DSSIZE of 32GB with the maximum number of 4096 Partitions, a total table size of 128TB.

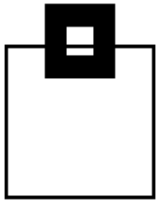


The calculation to find out how many partitions with what DSSIZE and page size were still a major headache of course!

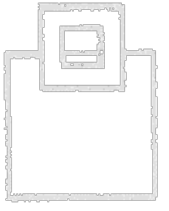
It was also around this time that people started getting problems with NPIs as they could get very large and had only “pieces” to control the dataset allocation.



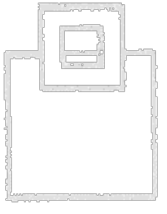
Tablespaces – A brief history



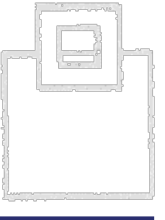
In DB2 8.1 Partitioned spaces got yet another major upgrade with the ability to have no partitioning index defined!



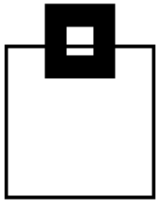
It was termed “table based partitioning” and simply put the partitioning key into the table definition. This saved the complete PI from being created – If you only wanted to partition but not use the PI for data access then you did not bother to create it!



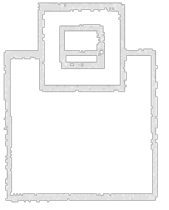
This release also brought in the DPSI (Data-Partitioned Secondary Index) which is still causing grief to this day and renamed our beloved NPI to NPSI (Non-Partitioned Secondary Index).

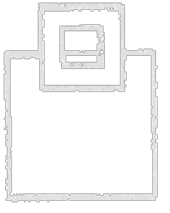


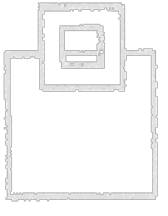
Agenda



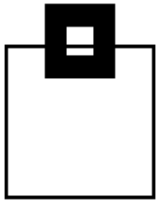
- Tablespaces – A brief history
- Universal Tablespaces – A brief history
- Present UTS situation
- Future UTS situation
- Q&A







Universal Tablespaces – A brief history



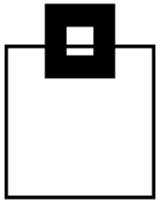
In DB2 9.1 Partitioned spaces got a completely new type – The Universal Tablespace (UTS).

UTS is basically a cross between Segmented and Partitioned. It gives the mass delete speed and spacemap performance boost of Segmented with the ability to partition and get extremely big!

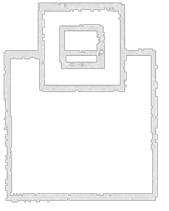
Delivered in two flavors:

1) Partitioned-by-Growth (PBG)
2) Partitioned-by-Range (PBR)

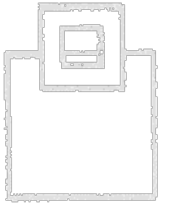
Universal Tablespaces – A brief history

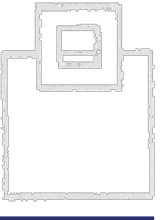


PBGs caused problems from the get-go. The ability to define MAXPARTITIONS at 4096 "just because you could" was quickly found out to be a disaster!

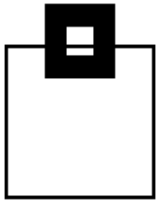


"Although physical data sets are not defined when the MAXPARTITIONS value is issued, there can be storage and CPU overhead. If an increase in the number of partitions is expected by using the MAXPARTITIONS clause, be aware that specifying a value larger than necessary, such as 4096 (the maximum value), as a default for all of your partition-by-growth table spaces can cause larger than expected storage requests."

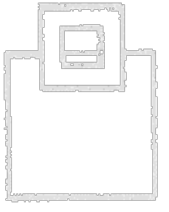


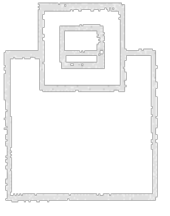


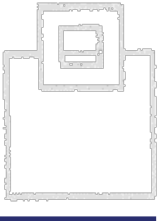
Universal Tablespaces – A brief history



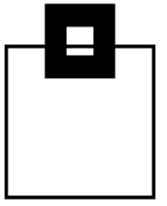
PBRs on the other hand were welcomed with open arms! Migrating to them was a bit of a pain but we will see that that got worse over time…







Universal Tablespaces – A brief history



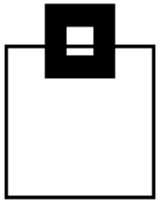
UTS is so good because IBM DB2 development has mentioned, on multiple occasions, that any new DB2 features will only be developed for UTS usage.

This has been seen with the developments of:

- CLONE spaces
- HASH spaces
- Currently committed locking behavior
- Pending DDL
- Inline LOBs
- XML multi-versioning
- ALTER TABLE with DROP COLUMN

This list just keeps on growing.

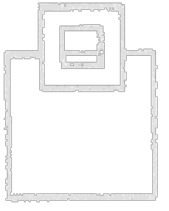
Universal Tablespaces – A brief history

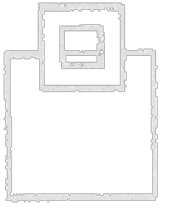


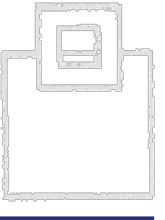
The maximum allowed size of any given partition is dependent on whether or not LARGE / DSSIZE and/or more than 254 partitions were used on the CREATE TABLESPACE DDL.

If LARGE not used:

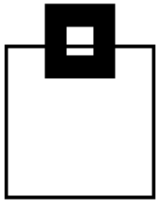
NUMPARTS	Maximum Partition Size (default for DSSIZE)
1 to 16	4GB
17 to 32	2GB
33 to 64	1GB
65 to 254	4GB





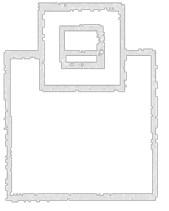


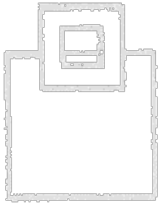
Universal Tablespaces – A brief history



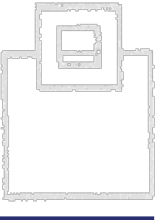
If NUMPARTS > 254 then the size is dependent on the page size:

Page size	Maximum Partition Size (default for DSSIZE)
4KB	4GB
8KB	8GB
16KB	16GB
32KB	32GB

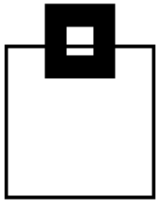




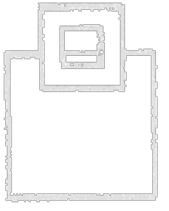
If you specified DSSIZE then the maximum number of partitions was constrained…



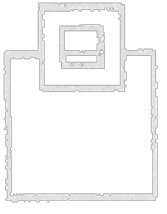
Universal Tablespaces – A brief history



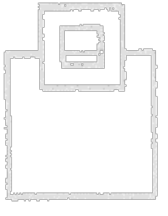
In DB2 10 Partitioned spaces got an upgrade, the DSSIZE keyword got expanded to allow 128GB or 256GB.



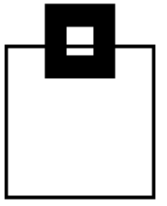
For PBGs due to the tie-in to page size and MAXPARTITIONS the physical maximum size stayed at 128TB.



For PBRs the maximum Partition size (DSSIZE) depends on how many partitions there are defined and their page size.



Universal Tablespaces – A brief history



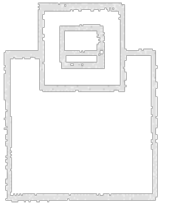
With DB2 10 DSSIZE and page size examples:

4KB page size with DSSIZE 64GB maximum NUMPARTS is 256.

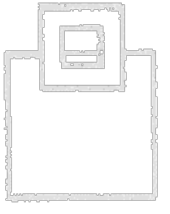
8KB page size with DSSIZE 64GB maximum NUMPARTS is 512.

16KB page size with DSSIZE 256GB maximum NUMPARTS is 256.

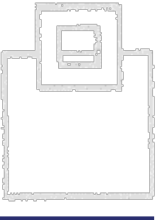
32KB page size with DSSIZE 128GB maximum NUMPARTS is 1024.



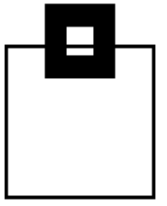
The maximum tablespace size was with 32KB page size and 32GB to 256GB DSSIZE with a variety of NUMPARTS limits but all with 128TB maximum size.



All in all an extremely confusing mish-mash of numbers and limits! It used to drive me crazy!

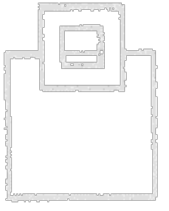


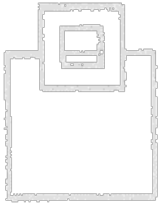
Universal Tablespaces – A brief history

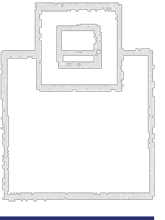


Problems with UTS:

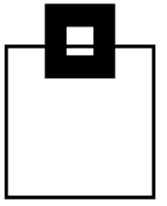
- Changing the size of a single partition or partitioning index simply not possible as “one size fits all”. You must change all partitions and then do a TS/IX level REORG to apply.
- Insert space map processing in PBGs is a disaster as the number of partitions grows. Even worse if you have all descending key columns. Then you can get empty partitions if you are unlucky!
- Utility parallelism not feasible with PBG – One huge space.
- Understanding the links between LARGE/DSSIZE/NUMPARTS and page size.





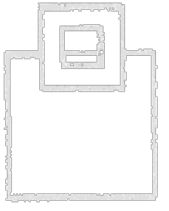


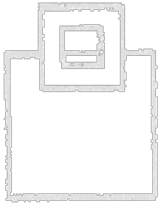
Universal Tablespaces – A brief history

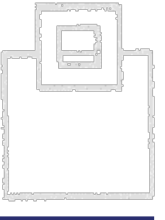


In Db2 11 all non-UTS spaces got deprecated. Only UTS should be used from now on.

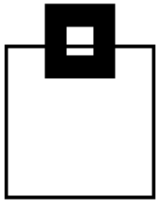
IBM (John Campbell et al) started recommending using lower and lower MAXPARTITIONS.



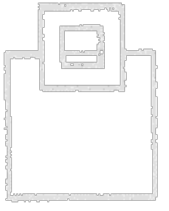


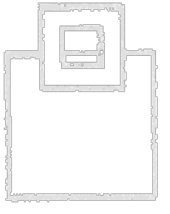


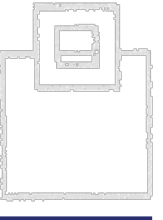
Agenda



- Tablespaces – A brief history
- Universal Tablespaces – A brief history
- Present UTS situation
- Future UTS situation
- Q&A







Present UTS Situation

In Db2 12 another new tablespace appeared: The UTS PBR Relative Page Number (RPN). This was, for me, the best thing in Db2 12.

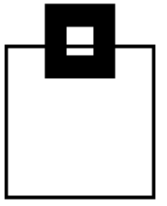
Normally, the first bits of the five byte RID contain the partition number and as this number got larger the available bits to point to a page number shrunk, hence the huge number of tables describing the different limits.

In Db2 12 UTS PBR RPN, all that changed as now the RID went to seven bytes and completely decoupled anything with partitions and page size and DSSIZE!

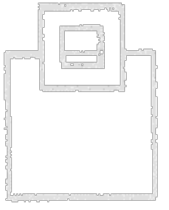
Great blog from Haakon Roberts all about it is here:

PBR RPN explained (idug.org)

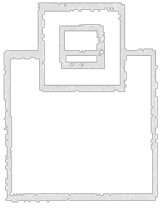
Present UTS Situation



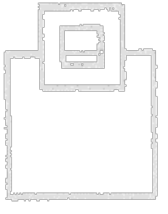
Furthermore, the attributes of a PBR were extended down to the partition level, so you have the ability to allocate *each* partition from 1GB to 1024GB using any value, e.g. 117GB - no more binary steps!



Each Partitioned Index can also be allocated with a DSSIZE just like the Partition.



The best bit, however, is that you can ALTER both of these values on-the-fly while a LOAD is running and it works! No REORG Pending, in fact no REORG: As long as you add a few GBs it is an immediate ALTER and the LOAD carries on. This was fantastic!



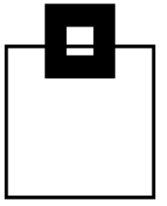
Plus all Partitions and Index Partitions can have a different DSSIZE!

Present UTS Situation

Of course it is not so great if you wish to reduce the size…then it is a PENDING DDL with a TS/IX level REORG. But whose data shrinks???

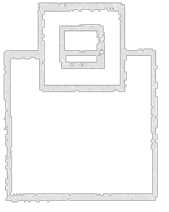
The worst problem was the migration from UTS PBR to UTS PBR RPN as it required a TS level reorg with TP level inline image copies. Now most people's PBRs were big and so allocating 1000's of tape units just didn't really fly at all… This has been fixed with the ICLIMIT_TAPE/ICLIMIT_DASD REORG utility parameters nowadays! Hoorah!

Present UTS Situation

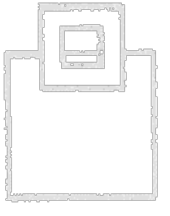


The official recommendations around UTS spaces are:

- For PBGs allocate MAXPARTITIONS 1 DSSIZE 64GB.
- For PBR use RPN for all new spaces and migrate current PBRs to the new RPN format.

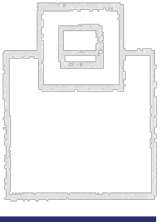


In Db2 12 FL508 a new ALTER TABLESPACE MOVE TABLE syntax was introduced to enable an easier migration from Simple and Segmented tablespaces to PBG spaces.

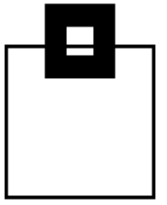


Full instructions are here:

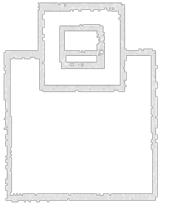
Moving tables from multi-table table spaces to partition-by-growth table spaces - IBM Documentation

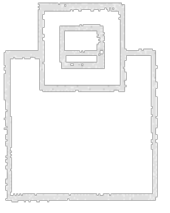


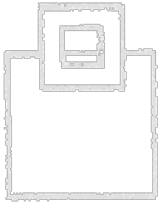
Agenda



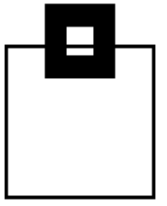
- Tablespaces – A brief history
- Universal Tablespaces – A brief history
- Present UTS situation
- Future UTS situation
- Q&A



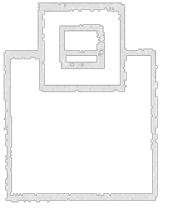




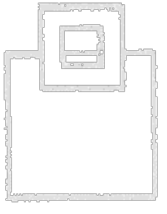
Future UTS Situation

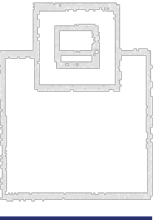


In Db2 13 IBM introduced the ability to convert from PBG to PBR tablespaces. Why?

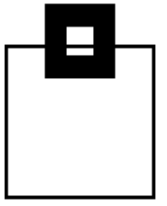


"PBG and PBR universal table spaces (UTS) are the strategic table space types for tables in Db2 for z/OS. PBG table spaces are the default UTS type, and they are well-suited for small to medium-sized tables. However, if an existing table in a PBG table space grows too large, performance degradation or data and index management issues might arise. Consider converting from PBG to PBR when that occurs.



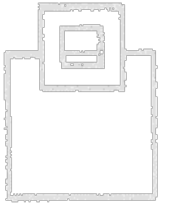


Future UTS Situation

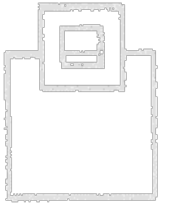


To do this, issue an

ALTER TABLE tablename ALTER PARTITIONING TO
  PARTITION BY RANGE (col1, col2,...)
    (Partition 1 ending at (aaa,bbb,...),
     Partition 2 ending at (ccc,ddd,...),
     Partition 3 ending at (MAXVALUE or MINVALUE));

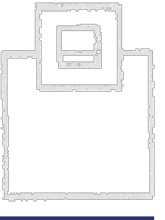


Once the REORG has run your PBG is now a PBR RPN space!

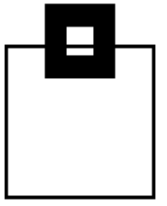


All the details are here:

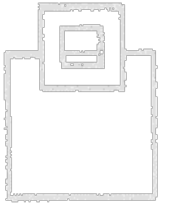
Converting tables from growth-based to range-based partitions - IBM Documentation



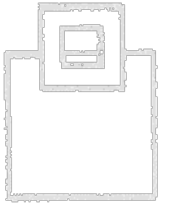
Future UTS Situation



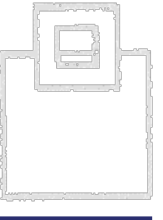
However there are, as always, a few limitations...



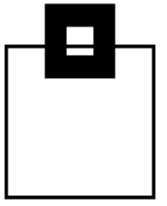
- No XML support
- No LOB support
- Not in a CLONE relationship (Who uses CLONEs these days?)
- Not a HASH table (Deprecated anyway!)



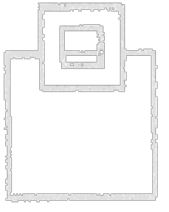
In the IBM Aha system there is a request to get LOB supported.
Logon and go to DB24ZOS-I-1379 to vote for it.



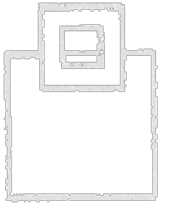
Future UTS Situation



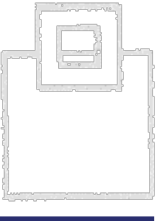
As of now, with Db2 13, we can define a UTS PBR RPN with DSSIZE 1024GB to give us a maximum table size of 4PB which can contain 280,375,465,082,880 rows of data (Over 280 trillion!).



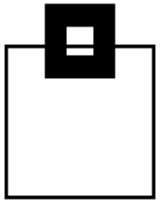
But this is not the end…



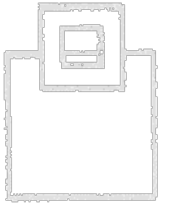
If you look into the definitions IBM left themselves room to grow!



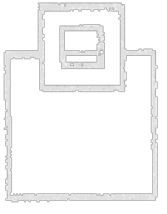
Future UTS Situation



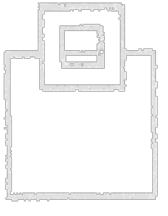
Most of the fields holding the number of 4KB pages are four byte full-words… Up until now the highest used value is x'10000000' which leaves a *lot* of free room.



The seven-byte RID gives us two bytes for partition number. That is 32,767 if using a signed half-word.



I have no idea if any of these will ever be realized but just think how large it could all get!



It looks to me like the big problem is going to be the OBID…

Questions & Answers