DB2 Partitioning
Choices, choices, choices

Dan Lohmeier, BMC Software – presenter

Phil Grainger, BMC Software - author

Platform: DB2 for z/OS
PRE - DB2 VERSION 8
Index Based Partitioning

- **Index Controlled Partitioning**
  - Partitioning is controlled by an index called a *partitioning index*. When you define a partitioning index on a table in a partitioned table space, you specify the partitioning key and the limit key values in the PARTITION clause of the CREATE INDEX statement.
  - **Non-large**
    - The high-limit key is not enforced
    - Partition limits – 1-16, 17-32, 33-64, 65-254
  - **Large**
    - The high-limit key is enforced
    - Partition limits – based on page size/DSSIZE
DB2 VERSION 8
Table Based Partitioning

• Table controlled partitioning
  • Partitioning is not controlled by an index. You specify the partitioning key and the limit key values for a table in a partitioned table space by using the PARTITION BY clause and the PARTITION ENDING AT clause of the CREATE TABLE statement. When you use this type of partitioning, an index is not required for partitioning.

• Version 8 introduced TABLE BASED PARTITIONING
  • What we had before was INDEX BASED PARTITIONING

• Both are currently supported
  • DB2 can almost invisibly switch from index-based to table-based.....
Table Based Partitioning

- A partitioned table NO LONGER needs a partitioning index
- The table itself is defined as partitioned with the relevant key ranges
  - Note that the tablespace in which the table will reside must also have the same number of partitions
- `CREATE TABLE table_name (col1, col2, col3), PARTITION BY (col1, col2),
  ( PART 1 VALUES (aaa,xxx),
  PART n VALUES (bbb,yyy);`
Table Based Partitioning

• Conversion happens when an ALTER is issued that is valid for table based partitioning only
  • ALTER INDEX NOT CLUSTER for the partitioning index
  • ALTER TABLE ADD PART
  • CREATE INDEX PARTITIONED
  • Etc.
• IBM suggest CREATE a PARTITIONED SECONDARY INDEX (DPSI) with the DEFINE NO, DEFER YES keywords
  • This does nothing and does not affect availability
• And then DROP this new index
Table Based Partitioning

• Because partitioning does not use an index, CLUSTERing does not necessarily relate to PARTITIONing any more

• With INDEX BASED partitioning secondary indexes contain keys for all partitions
  • This severely affects availability and operation of utilities

• With TABLE BASED partitioning secondary indexes can also be partitioned on the same keys
  • However they could NOT be UNIQUE.....
Data Partitioned secondary indexes

- A Data Partitioned Secondary Index (DPSI) then
  - Is an index on a partitioned table which is also physically partitioned
  - It MUST be non-unique (for DB2 V8) as potential duplicates could be in any one of 4,095 other DPSI partitions
  - If you need a Unique secondary index then it couldn’t be partitioned
  - DPSIs may not be ideal for access paths requiring a sequential index scan
    - For example, a SELECT of all customers in “AK” for example
      - Partitioned by account number
    - DB2 would have to scan ALL partitions of the DPSI, despite STATE being the high order key column
    - A traditional NPI would be better
Data Partitioned secondary indexes

- On the other hand, DPSIs are great for utility concurrency
- All utilities can run at partition level
Adding Partitions

• A partition can be added at the end of an existing partitioned table
  • But only up to the maximum number of partitions
  • From 4,096 DOWN to 254, “depending” on tablespace attributes

• The partition number itself is not specified
  • It is chosen by DB2

• But the new limit keys must, of course, be specified
  • Not so simple if current last partition contains ALL remaining key values
Adding Partitions

• All plans, packages and cached dynamic SQL statements are invalidated

• We now need to be aware of a VERY subtle difference between LOGICAL partition number and PHYSICAL partition number
  
  • LOGICAL refers to the partition number of the TABLE (counting keyranges)
  • PHYSICAL refers to the Annn node of the pageset name
Rotating Partitions

- DB2 Version 8 also allows partition ROTATING
- This is where the partition that is currently the “first” (the one with the lowest keys) is rotated to become the “last” (which will now contain the highest keys)
- All rows will be deleted from the rotated partition
  - And all SYSCOPY & SYSLGRNX rows for it will be purged
- Also, the new limit keys for the “new” partition will be specified on the ALTER ... ROTATE statement
Rotating Partitions

- When we rotate FIRST to LAST, we effectively remove the old first partition and what was the SECOND one now becomes the first
- Remember, partition keys are only specified ONCE for each partition
  - We list the HIGHEST key that can reside in any partition
- So, after ROTATING, a row that would have been placed into PART 1 will now be placed into PART 2 instead!
Rotating Partitions

- Remember logical vs physical a few pages back?
- We could start with the same 10 partition table
  - That has physical datasets A001 – A010
- If we “ROTATE FIRST TO LAST”
  - Logically we still have partitions 1-10 (it’s just that the old part 2 is now part 1 etc)
  - BUT DB2 doesn’t rename the pagesets
  - So logical partition 1 is now physical pageset A002
  - Logical partition 9 is now physical pageset A010
  - And logical partition 10 is physical pageset A001(!)
Rotating AND Adding

• This is where it will get messy
• Start
  • with Parts 1-10 (A001-A010)
• Rotate first to last
  • Now we have A002-A010 followed by A001
• Add a partition
  • Now there’s A002-A010, then A001, finally A011
• Got all that???
• LOGICAL and PHYSICAL partition numbers are no longer guaranteed to be the same....
Unique DPSIs

- A DPSI can now be unique
- IF its high order columns include the partitioning key
- Keeps potential duplicates in the same partition

- Otherwise, DPSI must still be non-unique
Universal table spaces

- A Universal table space is one that is both SEGMENTED and PARTITIONED
  - Note, simple table spaces are on their way out
  - Do NOT drop one by mistake ..... 
  - Or even on purpose
  - **Impossible** to recreate

- All the benefits of a segmented table space with partitioning
Universal table spaces

- There are also now TWO types of partitioning
- Partition by RANGE
  - Key range partitioning that we’ve always had
- Partition by GROWTH
  - New partitions are added as data grows
  - This is NOT “partitioning” that we understand
UTS – Partition by range

- This is no different to Version 8 (and prior) partitioning
- A UTS range partitioned table space is created just like a normal partitioned table space
  - But with a SEGSIZE clause
UTS – Partition by growth

- Starts off as a one partition table space
- As data is inserted/loaded the partition will fill up
- When it is full, another partition is automatically created
  - No -904
  - No partitioning key
- You specify the limit to the number of partitions on the CREATE table space statement
  - MAXPARTITIONS
UTS – Partition by growth

• In fact MAXPARTITIONS is the ONLY keyword that denotes a PBG table space
• The maximum number of partitions is the same as the maximum number of parts for a PBR table space
  • Depends on pagesize and DSSIZE
  • From 256 to 4,096
• PLEASE don’t default to the maximum
UTS – Partition by growth

• NOTE

• Because rows are in partitions arbitrarily
  • Not based on a key

• Some partition operations make no sense
  • Rebalance
  • LOAD INTO PART
  • ALTER LIMITKEY
  • Add/Rotate partitions
  • Etc
UTS – Partition by growth

• Of course, secondary indexes can be built on PBG table spaces
• These will ALWAYS be Non Partitioned Indexes
  • Because there is NO concept of a partitioning key
UTS – Partition by growth

- Allocation of new partitions is NOT rolled back
  - Even if the inserting UOW is
- Reorganisation may reclaim space IN a PBG table space
  - But in DB2 9, empty datasets are not deleted
- If LOB or XML columns are involved, rows cannot move between partitions
  - Space IN a partition may be reclaimed
UTS – Partition by growth

- COPY SHRLEVEL CHANGE will NOT copy a partition that has “appeared” during the copy process
## Table space specifications

<table>
<thead>
<tr>
<th>NUMPARTS</th>
<th>SEGSIZE</th>
<th>MAXPARTITIONS</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Segmented table space (segsize 4)</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Universal Partition by Growth table space</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Segmented table space</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Simple partitioned by range table space</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Not valid</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Universal table space (partitioned by range)</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Universal Partition by Growth table space</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not valid</td>
</tr>
</tbody>
</table>


DB2 10
Default tablespace type

- Partitioned by Growth (PBG) is now the default table space type
  - NOT Simple
  - Not even Segmented
- Single table tablespaces can be ALTERed to PBG
- Classic partitioned tablespace is now deprecated
- DSNZPARM can override default segsize to 0
  - To support implicit classic partitioned
  - NOT recommended by IBM
Reorganization of PBGs

- REORG TABLESPACE might leave a table space in COPY-pending status when:
  - The table space is a partition-by-growth table space
  - The table space contains LOB columns
  - Rows are moved to a different partition or a new partition is added to the table space

- Reorg of a partition may fail if the data does not fit back!
Workfile database

- Workfile tablespaces can now be PBGs
  - Or a mixture of PBGs and segmented
- Must drop/recreate
  - Alter is not possible
Catalog and Directory

- Most catalog and directory tablespaces are now PBG
  - And thus SINGLE table tablespaces

- You DID update your disaster recovery plans didn’t you?
ALTER .... ROTATE changes

• ANY partition
  • Apart from the current “last partition”

• Can be rotated “to last”
  • ALTER ...... ROTATE PARTITION nnn TO LAST

• This makes a dangerous hole even more dangerous
  • Physical vs. logical partition numbers

• But sort of provides a “drop partition” capability
DB2 11
No major changes to partitioning

- Have you got the message?
  - UTS PBG is the future for non-PBR tablespaces
  - UTS PBR otherwise
- No segmented
- No simple
- No classic partitioning
Workfile tablespaces: IBM Recommendation

- “If you have not already done so, drop work file table spaces
- and re-create them as partition-by-growth table spaces”
**PBG reorganization changes (1)**

- **REORG_DROP_PBG_PARTS** specifies whether trailing empty partitions are removed.
- **When operating on an entire partition-by-growth table space**
  - ENABLE or DISABLE
- **An empty trailing partition occurs when**
  - REORG moves all data records from a partition
- **Default is DISABLE**
PBG reorganization changes (2)

- **REORG_IGNORE_FREESPACE** controls whether DB2 ignores the PCTFREE and FREEPAGE
  - Can avoid “out of space” conditions during reorgs
    - Especially partition level ones
- **NO** or **YES**
  - Default is **NO**
- The parameter is an **IGNORE**
  - Yes – means IGNORE freespace
  - No - means HONOUR freespace!
No major changes in DB2 11, but

- Altering limitkeys is now a DEFERRED alter
  - `ALTER TABLE ALTER PARTITION n ENDING AT <new key>`
  - Puts partition(s) in AREOR
  - For PBR and classic partitioned with table partitioning ONLY
  - Next online reorg materializes new limit keys
  - PIT recovery to before the reorg is supported – REORP
No major changes in DB2 11 and

- **REORG ... REBALANCE** .. SHRLEVEL CHANGE
  - On-line rebalance
- Partition level in-line copies
AND ....?
Addressing partition limits

- New UTS PBR called UTS PBR Relative Page Number (RPN)
  - *UTS PBR RPN* – Hooray for the TLA 😊
- Pages are numbered PER PARTITION
- Eliminates inverse relationship between number of partitions and individual partition size
  - And DSSIZE is now at partition level
- 7 byte RID (2 for partition, 5 for page number)
  - Up to 1TB partition size
  - Up to 4PB table size (but not finished yet)
  - 256(ish) TRILLION rows per table space
  - Theoretical 64K partitions?
More on PBR RPN

• Zparm for system default
• PAGENUM ABSOLUTE/RELATIVE for DDL
  • Can ALTER – this is a pending change
• Increasing DSSIZE is immediate
  • No reorg needed
• Decreasing DSSIZE is a pending alter
  • Still requires entire tablespace reorg
More on PBR UTS

- Insert partition
  - `ALTER TABLE ADD PARTITION ENDING AT xxx`
  - Data distributed between new and old partitions at reorg
  - Be wary of REALLY old classic partitioned tablespaces
    With truncated limit keys
QUESTIONS?
DB2 Partitioning Choices, choices