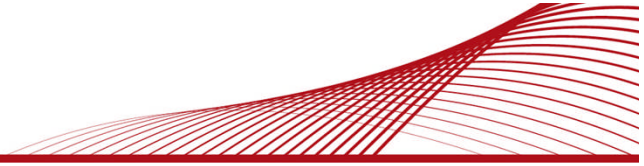
Partitioned Tables: First Introduced in DB2 9.1 Significant Enhancements in DB2 9.7

Mike Winer - *IBM*
mikew@ca.ibm.com

Special Edition for the DB2Night Show!
September 10, 2010. 11:00am - 12:00pm EST.



Partitioned Indexes – Benefits and Value

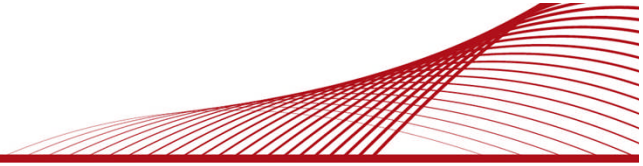
- **Streamlined and efficient roll-in and roll-out with ATTACH and DETACH**
  - Partitioned indexes can be attached/inherited from the source table
    - Matching partitioned indexes to target are kept, additional indexes are dropped from source
  - SET INTEGRITY maintains nonpartitioned indexes, creates missing partitioned indexes
    - Avoids time consuming process and log resource utilization
  - Partitioned indexes detached and inherited by target table of DETACH
  - No Asynchronous Index Cleanup after DETACH for partitioned indexes
- **Storage savings**
  - Partitioned indexes do not have the partition ID in each index key entry
    - Savings of 2 bytes per RID entry
  - Total size of partitioned indexes often smaller than nonpartitioned index
- **Performance**
  - Storage saving typically can translate to better performance
    - less I/O, better buffer pool utilization, etc.
  - Benefits the most when being used with partition elimination
    - especially for queries on a single partition
- **Facilitate partition independent operations**
  - Partition level and concurrent partition data and index reorganization.
  - Partitioned MDC block indexes (DB2 9.7 FP1)

DESCRIBE Command (also ADMIN_CMD)

- **DESCRIBE INDEXES FOR TABLE** command is extended to report if an index is partitioned or not with a new column “Index Partitioning” which can have values:
  - 'N‘ = This is a nonpartitioned index on a partitioned table
  - 'P‘ = This is a partitioned index on a partitioned table.
  - ‘ ’ = Index is not on a partitioned table
- Sample output: DESCRIBE INDEXES FOR TABLE *myDPartT*

```
Index           Index           Unique    Number of    Index
schema          name            rule      columns      Partitioning
--------------  --------------  --------  -----------  ------------
NEWTON          IDXNDP          D                   1  N
NEWTON          IDXDP           D                   1  P
```

- **DESCRIBE DATA PARTITIONS FOR TABLE** now includes the new column **IndexTblSpId**; the table space ID associated and used for the index partition.
- **ADMIN_GET_INDEX_INFO()** updated to report if an index is partitioned or not.



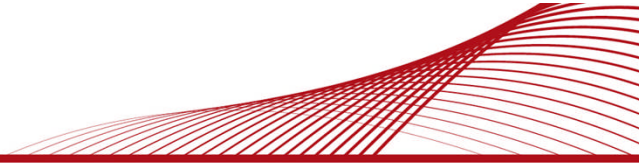
Reorg Table and Indexes

- REORG TABLE for a partitioned table is always offline
- REORG INDEXES ALL on a partitioned table is always offline
- REORG INDEX reorganizes a nonpartitioned index, supporting all access modes

- **Partition level reorg table and indexes are available in 9.7 FP1**
  - REORG TABLE <t-name> **ON DATA PARTITION** <part-name>
    - **ALLOW NO/READ ACCESS** applies to part-name, not the entire partitioned table
  - **Without** nonpartitioned indexes (except xml path), the ALLOW READ ACCESS mode is the default behavior with full read/write access to all other partitions
  - **With** nonpartitioned indexes (except xml path), ALLOW NO ACCESS is the default and only supported mode.

  - REORG INDEXES ALL FOR TABLE <t-name> **ON DATA PARTITION** <part-name>
    - **ALLOW NO/READ/WRITE ACCESS** applies to part-name, not the entire partitioned table
    - **ALLOW WRITE ACCESS** is *not* supported for MDC tables (w/ mdc block indexes).

- ***Concurrent partition reorg (TABLE and INDEXES ALL) supported when there are no nonpartitioned indexes (excluding XML column paths index) and ALLOW NO ACCESS is specified***



Examples on Defining a Partitioned Table

```
CREATE TABLE sales (sale_date DATE, customer INT, invoice XML)
        PARTITION BY RANGE (sale_date  NULL FIRST)
          IN Tabsp1, Tabsp2   INDEX IN  Tabsp3  LONG IN TabspL
        (                   STARTING MINVALUE  ENDING '12/31/1999' INCLUSIVE,
         PARTITION P1 STARTING '1/1/2000'   ENDING '3/31/2000'   INDEX IN tbsp1       LONG IN TabspL1,
                                             ENDING '6/30/2000'  INDEX IN idxTbsp2  LONG IN TabspL2,
          PARTITION p4                       ENDING '12/31/2004' IN Tabsp4 INDEX IN idxTbsp4,
          STARTING '1/1/2005'                ENDING '12/31/2006'  EVERY 2  MONTHS );
```

- Use **STARTING** … **ENDING** … to specify ranges
- Can combine the short syntax with long syntax:
  - **STARTING** …**ENDING** … **EVERY** ..
- Use **MINVALUE** or **MAXVALUE** to specify open ended (similar to infinity)
- Use **INCLUSIVE** and **EXCLUSIVE** to qualify bounds
- Specify partition name. For example **PARTITION P1, PARTITION P4**
- Specify default table space(s) for partitions and nonpartitioned indexes table level
- Specify table space(s) for DATA, partitioned indexes, long data at partition level
- NULL value placement (not in example)
- Multi-column partitioning: **PARTITION BY RANGE (year, month)** (not in example)