Information Management

IBM

Multidimensional Clustering (MDC) Tables in DB2 LUW

DB2Night Show
January 14, 2011

Pat Bates, IBM Technical Sales Professional, Data Warehousing

Paul Zikopoulos, Director, IBM Information Management Client Technical Professionals

Agenda

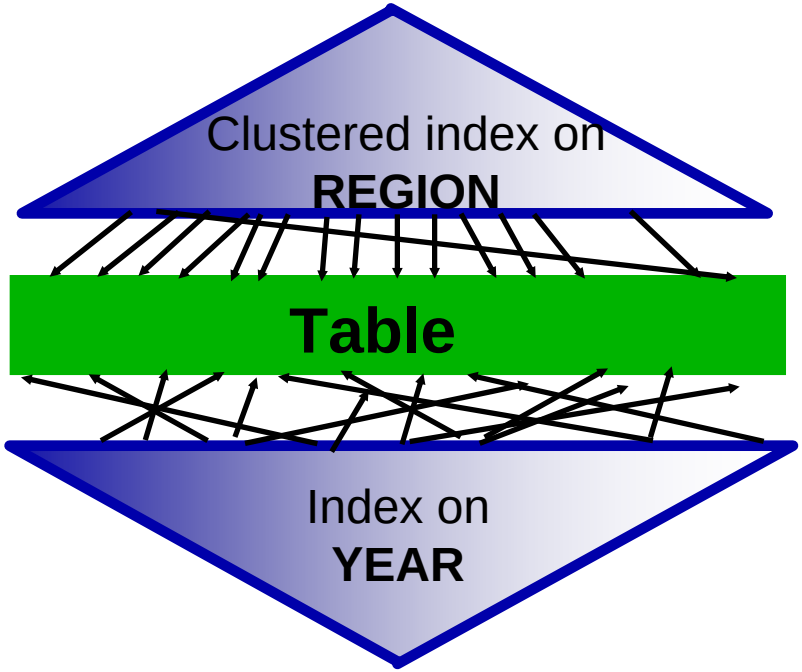
- **Multi-Dimensional Cluster Tables (MDCs)**
  - Why are they useful?
  - MDC Precursor: Traditional Clustered Indexes
- **MDC Benefits**
- **How They Work**
- **A History of Enhancements**
- **Customer Experiences**
- **DB2 for Data Warehousing and Business Intelligence**
  - Breaking down I/O barriers

Multidimensional Clustering Tables (MDCs)

- **MDCs are a unique object in DB2 LUW that provide many advantages over other indexes**
  - Particularly regular clustered indexes
- **Provide continuous, flexible and automatic clustering of data on disk**
- **Yield significant improvements in**
  - Query performance
  - Disk space efficiency
  - Data management overhead
- **"Dimensional" and great for warehousing / BI**
  - Great for OLTP too

Before MDCs – Traditional Clustered Indexes

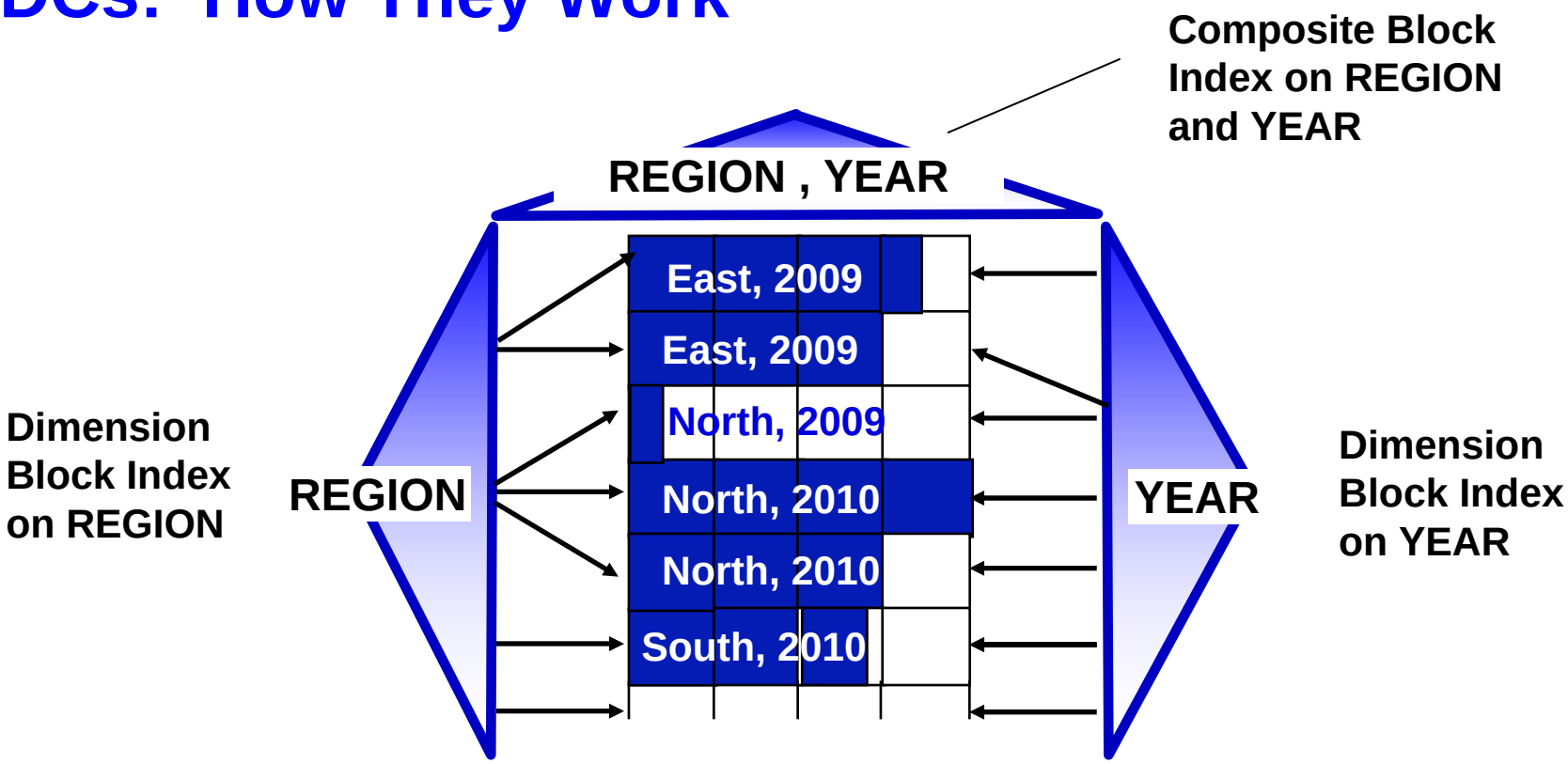
- **Data physically clustered according to the cluster column**
- **Efficient access on one dimension, but….**
- **Can only cluster on one column**
  - RID-based indexing on other columns doesn't benefit from ordering
- **Heavy maintenance load**
  - Inefficient disk clustering over time
  - Monitor and re-org to reclaim lost space
- **Large RID-based index overhead**
  - Excessive index space requirements



Benefits of MDCs

- **Efficient I/O == Performance**
  - 3-4X average query performance improvement, 10X+ for some queries
- **Automated dimensional index creation & management**
  - DB2 automatically creates and manages dimensional indexes
- **Never REORG an MDC table for re-clustering**
  - Only reorganize an MDC table to perform space reclamation
- **Up to 64 Clustered Indexes per table (Not just the one)**
- **90+% dimension index compression because of the on-disk nature of a MDC table and its associated block pointers**
  - You can mix MDC indexes with traditional RID indexes
- **Administration-free rolling ranges**
  - No manual ATTACH or DETACH for range cycling: just load the data and MDC automatically provides the clustering

MDCs: How They Work



- Data is ordered along extant boundaries according to dimension (clustering) values
- DB2 creates n+1 block indexes when MDC table is created
- BIDs point to blocks of rows, not rows, so very compact

Processing a SELECT on an MDC Table (AND)

1. **Index lookup done on each dimension block index**
2. **Join with block ANDing**
3. **Mini-relation scan of result blocks**

SELECT * FROM MDCTABLE WHERE COLOR='BLUE' **AND** NATION='USA'

Block Index on Color

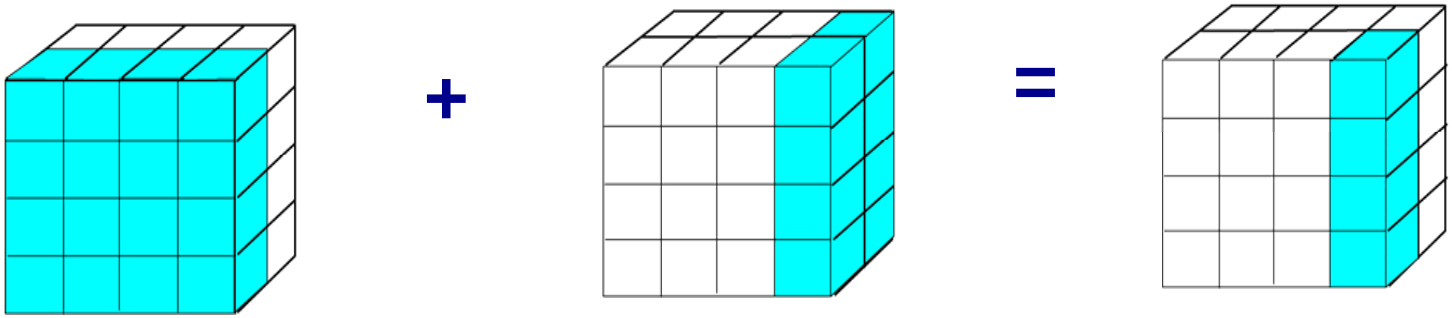
Blue	4,0	12,0	48,0	52,0	76,0	100,0	216,0

Block Index on Nation

USA	12,0	76,0	92,0	100,0	112,0	216,0	276,0

Resulting List of Blocks to Scan

12,0	76,0	100,0	216,0



Processing a SELECT on an MDC Table (OR)

1. **Index lookup done on each dimension block index**
2. **Join with block ORing**
3. **Mini-relation scan of result blocks**

SELECT * FROM MDCTABLE WHERE COLOR='BLUE' **OR** NATION='USA'

Block Index on Color

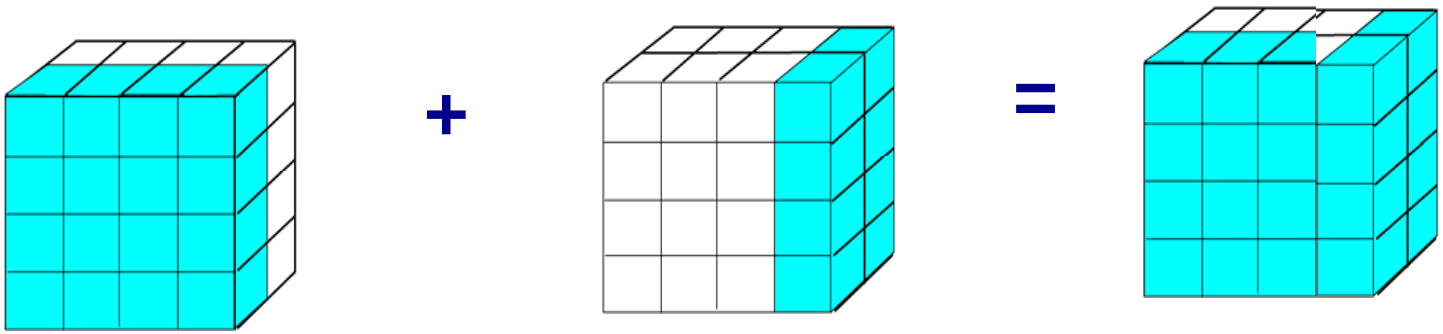
Blue	4,0	12,0	48,0	52,0	76,0	100,0	216,0

Block Index on Nation

USA	12,0	76,0	92,0	100,0	112,0	216,0	276,0

Resulting List of Blocks to Scan

4,0	12,0	48,0	52,0	76,0	100,0	112,0	216,0	276,0



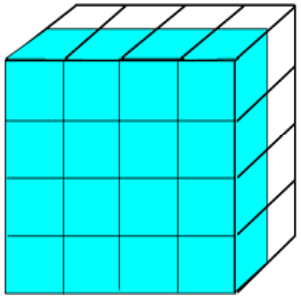
Processing a SELECT on an MDC Table (with RID)

1. **Block index lookup and RID index lookup**
2. **Block and RID ANDing**
3. **Result is row id's in qualifying blocks**

SELECT * FROM MDCTABLE WHERE COLOR='BLUE' **AND** PARTNO < 1000

Block Index on Color

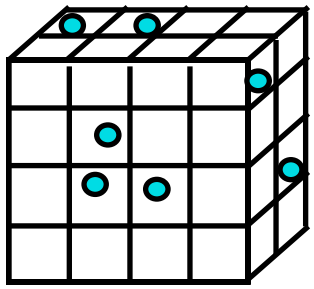
Blue	4,0	12,0	48,0	52,0	76,0	100,0	216,0



+

RIDs from Index

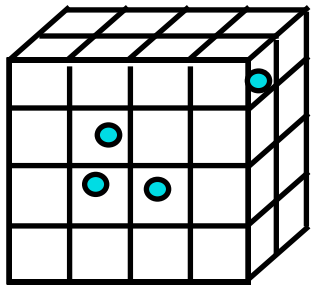
6,4	8,12	50,1	77,3	107,0	115,0	219,5	276,9



=

Resulting RIDS to fetch

6,4	50,1	77,3	219,5



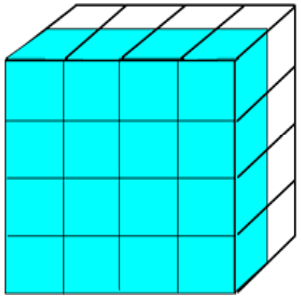
Processing a SELECT on an MDC Table (with RID)

1. **Block index lookup and RID index lookup**
2. **Block and RID ORing**
3. **Result is row id's in qualifying blocks**

SELECT * FROM MDCTABLE WHERE COLOR='BLUE' **OR**
PARTNO < 1000

Block Index on Color

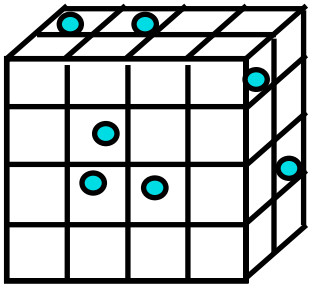
Blue	4,0	12,0	48,0	52,0	76,0	100,0	216,0



+

RIDs from Index

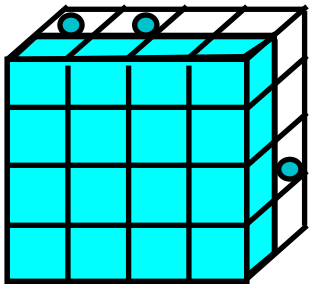
6,4	8,12	50,1	77,3	107,0	115,0	219,5	276,9



=

Resulting RIDS to fetch

	8,12	107,0	115,0	276,9			
Blue	4,0	12,0	48,0	52,0	76,0	100,0	216,0



MDC Benefit – Faster Queries

- Queries can take advantage of block indexes
- Quickly and easily narrow down a portion of the table having particular dimension values or ranges of values (block elimination)
- Block indexes are small for very fast index lookups
- Relation scans of blocks are faster than RID based retrieval
- Prefetch entire blocks of data for block index scanner - no need for sequential detection on data access
- Block-level index ANDing and ORing; mixed with RIDs
- Data is guaranteed to be clustered on extents – much faster retrieval
- BIDs provide additional access plan choices and do not prevent the use of traditional access plans (rid scans, joins, table scans, etc)

MDC Index Columns (Dimension) Selection

When choosing dimensions for a table, consider:

- **First, which queries will benefit? Examine workload and look for:**
  - Columns in equality or range queries
  - Columns with coarse granularity
  - FKs in fact tables – consider generated columns to group continuous values like employee numbers
- **Second, consider expected density of cells based on expected data**
  - # possible cells = cross product of dimension cardinalities (use stats)
  - Possibility of sparsely populated blocks/cells
- **Third, manipulate for optimal cell density**
  - Vary the number of dimensions
  - Vary the granularity of one or more dimensions (rollup to higher grain)
  - Vary the block (extent) size
- **Or… Use the DB2 Design Advisor!**

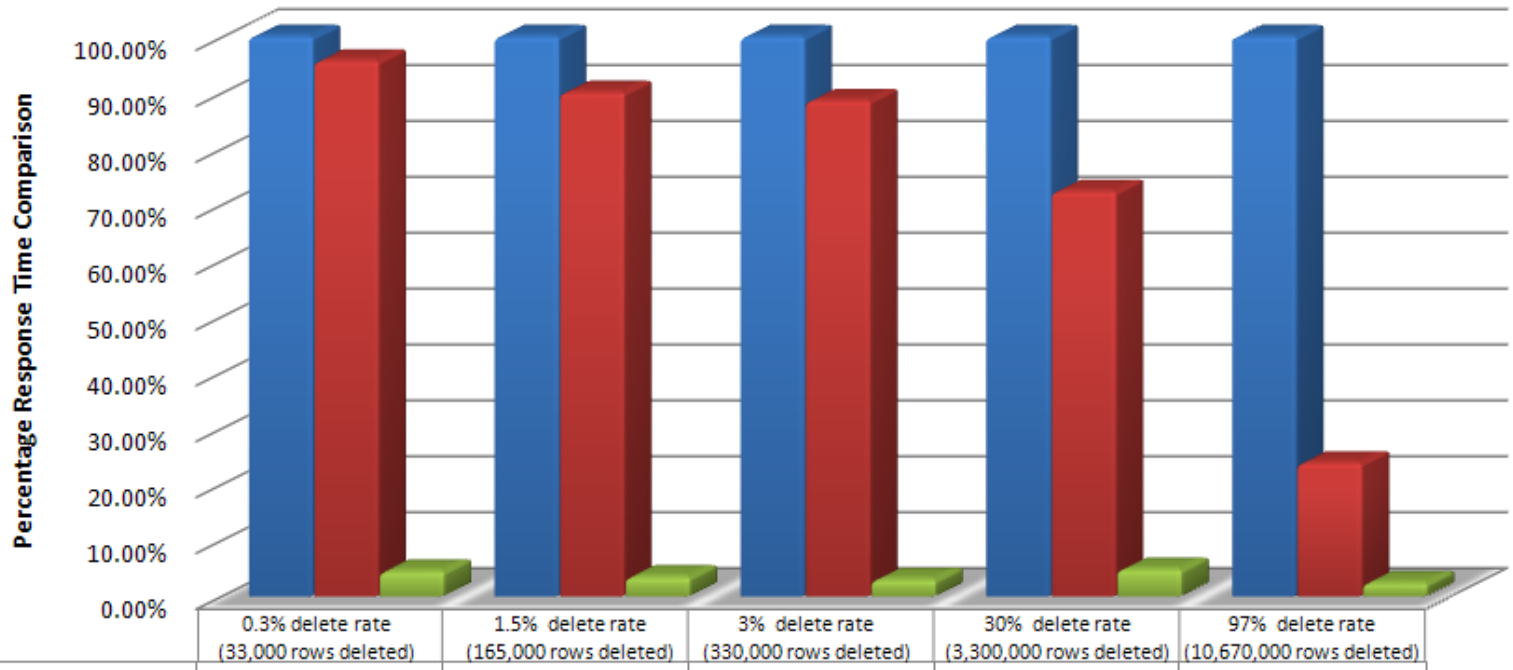
MDCs INSERT, UPDATE, DELETE

- **INSERT**
  - Find the Dimension combo in the composite block index – search those blocks for space
  - If not found or full: assign a new block (reuse empty block first)
- **DELETE**
  - Find the matching dimension combo
- **UPDATE**
  - Non-Dimension values: in-place update
    - If variable length column and no more space, find another block
  - Dimension column value: write to a different block
    - Convert to DELETE then INSERT
- **LOAD Utility**
  - LOAD and IMPORT work just like regular tables

MDC Block Delete Enhancements Since 8.1

- **8.2 – Design Advisor for dimension selection**
  - Let the Design Advisor do the work for you
- **8.2.2 and 9.1 – Block-by-block delete optimization**
  - Fast BID update and page-by-page delete
  - Secondary RID index update slow
    - Probe RID index, key-by-key deletes, write to log per index key deleted
  - Secondary indexes could result in long ROLL-out times
- **9.5 – Improved delete with asynchronous RID index cleanup**
  - Reduced I/O algorithm and page-by-page logging makes it very fast
  - Fully parallelized for multiple index updates
  - Perform all DELETE activity as a single unit for work - cleanup in a single pass of the data
- **Continuous improvement from 8.2 through 9.7 and beyond**

MDC Bulk Deletion Results in DB2 9.5



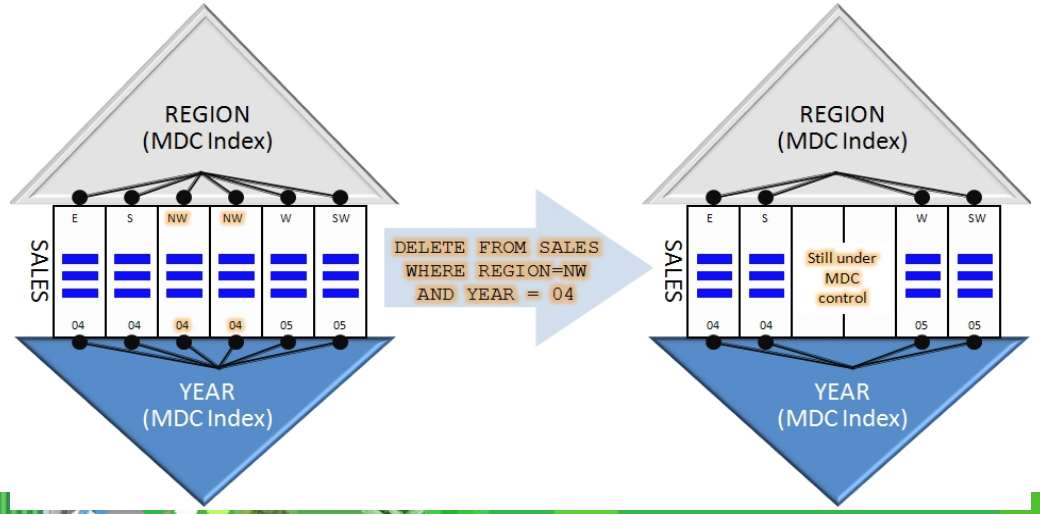
	0.3% delete rate (33,000 rows deleted)	1.5% delete rate (165,000 rows deleted)	3% delete rate (330,000 rows deleted)	30% delete rate (3,300,000 rows deleted)	97% delete rate (10,670,000 rows deleted)
NON- Optimized DELETE (DB2 8)	100.00%	100.00%	100.00%	100.00%	100.00%
IMMEDIATE DELETE (DB2 V8.2.2 & DB29)	95.65%	90.00%	88.57%	72.30%	23.73%
DEFERRED DELETE (DB2 9.5)	4.35%	3.33%	2.86%	4.69%	2.27%

NON- Optimized DELETE (DB2 8) IMMEDIATE DELETE (DB2 V8.2.2 & DB29) DEFERRED DELETE (DB2 9.5)

- **11,000,000 row table with 134,260 16KB pages and 8 RID indexes on a 4 node cluster**

MDC Sparse Table Enhancements in DB2 9.7

- **Pre-DB2 9.7, blocks remained 'property' of MDC table after DELETE**
  - Sole option was to perform classic REORG to reclaim space
    - Think of this as a high water marker issue for the space that MDC tables occupy
  - Even with empty extents in the MDC table, could have a full table space
- **Needed to perform full offline REORG to give space back to table space**
  - Table reconstruction necessary without concurrent WRITE access to the table
- **Operations on MDC table could reuse this space however**
  - Example: Insert new E and 04 dimensional set



Sparse MDC Tables From DB2 9.7 Onward

- **DB2 9.7 extends the `REORG` command with a new `RECLAIM EXTENTS ONLY`**

```
REORG TABLE <mdc table name> RECLAIM EXTENTS ONLY…
```

- **Can include in automated table maintenance**
- **Not really a REORG: no COPY phase, no shadow copy, etc.**
- **Allows you to free space back to the table space in a minimum amount of time with maximum concurrency**
  - Storage is freed incrementally during processing
  - Can control concurrency with `ALLOW` keyword during processing
    - `ALLOW WRITE` (default) allows concurrent transactions to read and write
  - Default to run on all partitions (range or hash): can override for specific partition
- **Very fast: done in-place with no data movement with minimal logging**
  1. Find empty blocks in block map
  2. Marks new empty block in the table's block map as unallocated
     - MDC table no longer thinks those pages belong to it
  3. Marks blocks as unallocated in the table space's space map pages SMPs
     - Now the table space thinks it can use them

Sparse MDC Tables

- **When should you REORG like this?**
  - Could make it auto-REORG

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <DB2AutoReorgPolicy
3 xmlns="http://www.ibm.com/xmlns/prod/db2/autonomic/config" >
4 <ReorgOptions dictionaryOption="Keep" indexReorgMode="Online"
5 useSystemTempTableSpace="false reclaimExtentsSizeForMDCTables="512" />
6 <ReorgTableScope maxOfflineReorgTableSize="0">
7 <FilterClause>TABSCHEMA NOT LIKE 'SYS%'</FilterClause>
8 </ReorgTableScope>
9 </DB2AutoReorgPolicy>
10
```

- **New RECLAIMABLE_SPACE column added to `ADMIN_GET_TAB_INFO()` function to help you make that decision**
  - Provides information that isn't available to catalog tables

- **Monitoring Examples**
  - Show me the amount of reusable space in my MDC table

```
SELECT reclaimable_space as SPACE_AVAILABLE FROM TABLE
 SYSPROC.ADMIN_GET_TAB_INFO_V97 ( 'paulz', 'emp'))
 AS RECLAIMABLE_SPACE_FOR_THIS_TABLE
```

  - Show me all MDC tables that have more than 10 MB of reusable space

```
SELECT tabschema, tabname, reclaimable_space
 FROM sysibmadm.admintabinfo
 WHERE reclaimable_space > 10,000,00
```

Performance Optimization for MDC Tables in 9.7

- **Pre-9.7: Empty extents could be brought into the buffer pool**
  - Starting point of new scan
  - Sequential pre-fetching algorithm grabs the data
- **Pre-9.7: Wasted memory and resources to bring a block (extent) of pages into the buffer pool that have no value**
- **DB2 9.7 & Onward: Empty blocks returned to the table space helps performance by avoiding these scenarios**

MDC Customer Experiences Sample

Database | IBM DB2 Universal Database

eWEEK.com

We saw performance improvements of as much as 40 percent when performing multidimensional queries.

- **Canadian Astronomy Data Centre**

"With the MDC function of the DB2 database, the customer can run queries on the more than one billion row database in less than a minute. Compared to other database solutions, that represents an acceleration of 20 to 70 percent for such complex queries"

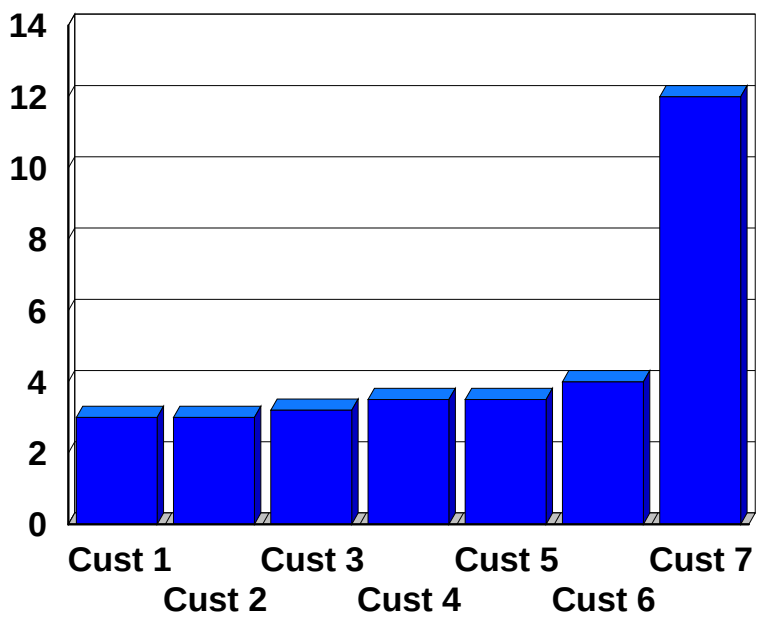
- **Brazil Telecom**

"By using MDCs, we were able to run (in less then 2 minutes) one very important report that will allow our company be more competitive. Such report was impossible to run in our environment because it was requiring too many resources from the system"

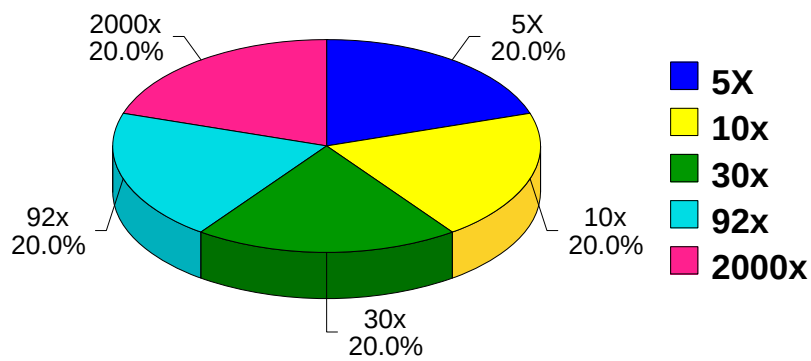
Customers Performance Experiences

- **Query performance results**:
  - Most averaged around or just above 3X query performance improvement
  - Maximum speedup included: 10X, 30X, 100X, 2000+X

Average query speedup - # of X



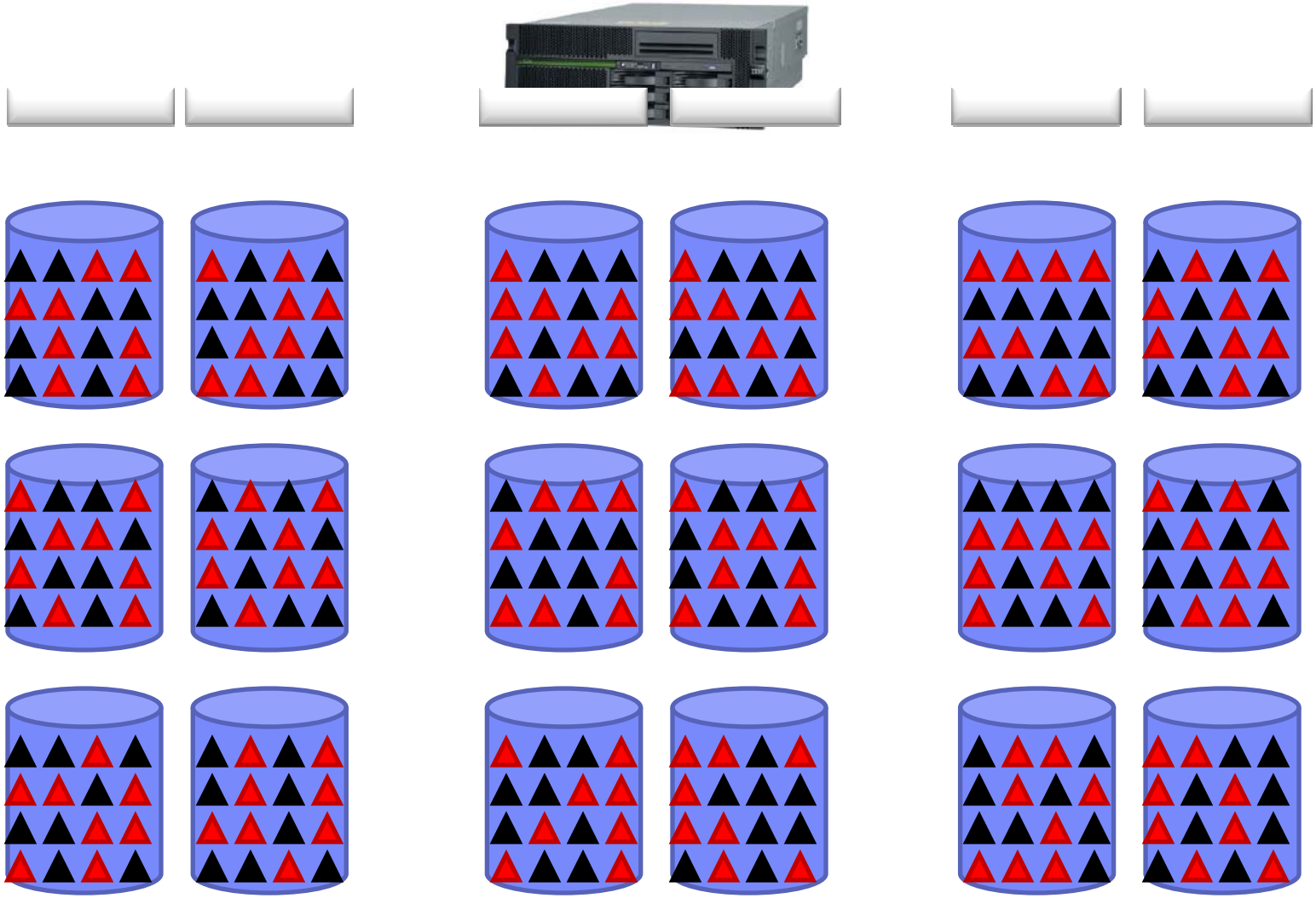
Maximum query speedup



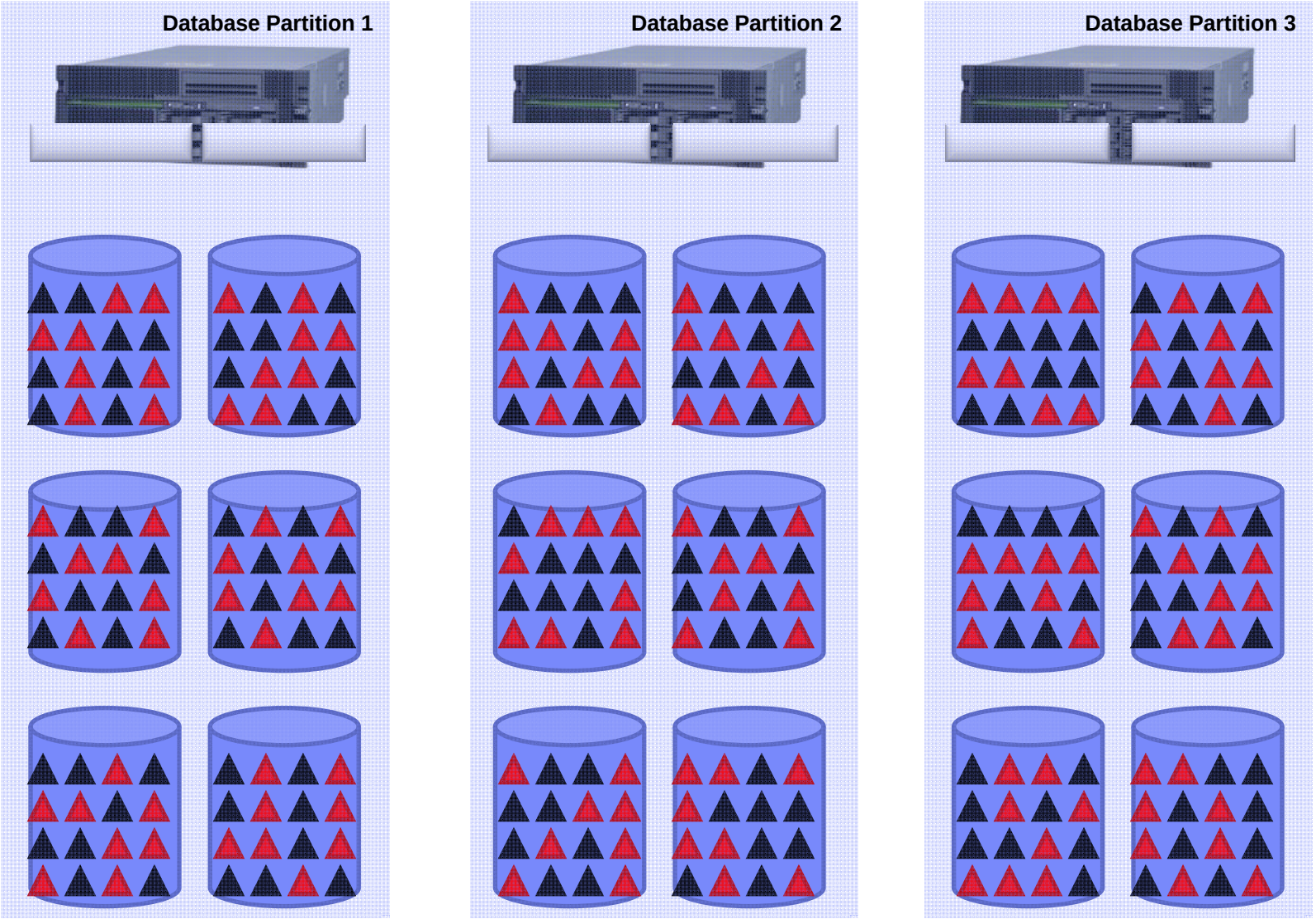
How Does DB2 with MDCs Help BI?

- DB2 has proven technology to break the I/O barrier
- Optimize the pipe with Deep Compression
- Parallelize I/O with Database Partitioning Feature (DPF)
- Reduce I/O with Range Partitioning
- Compact I/O with Multidimensional Clustering Tables (MDC)
- The following will illustrate…..

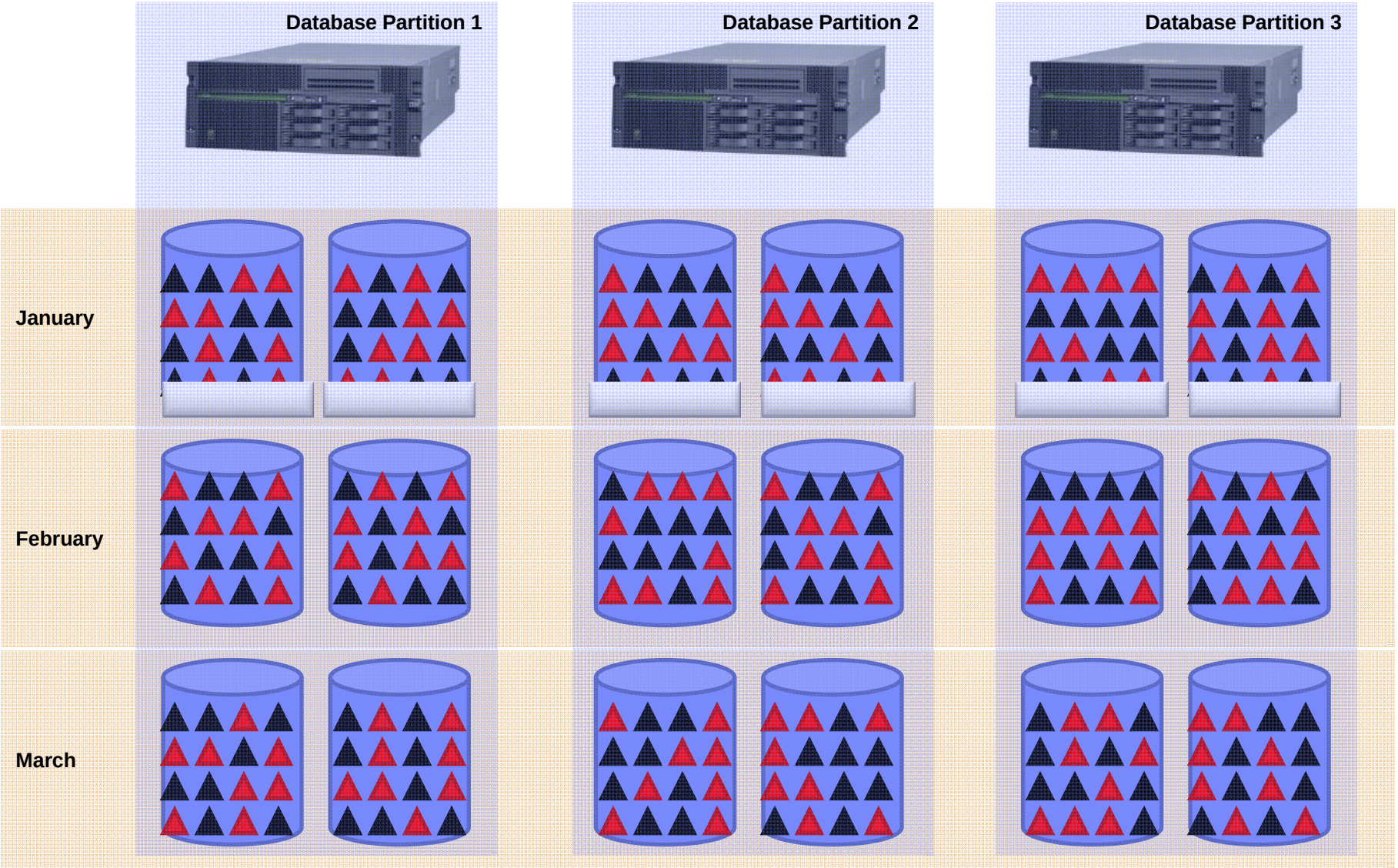
Traditional Large Scans Result in I/O Wait



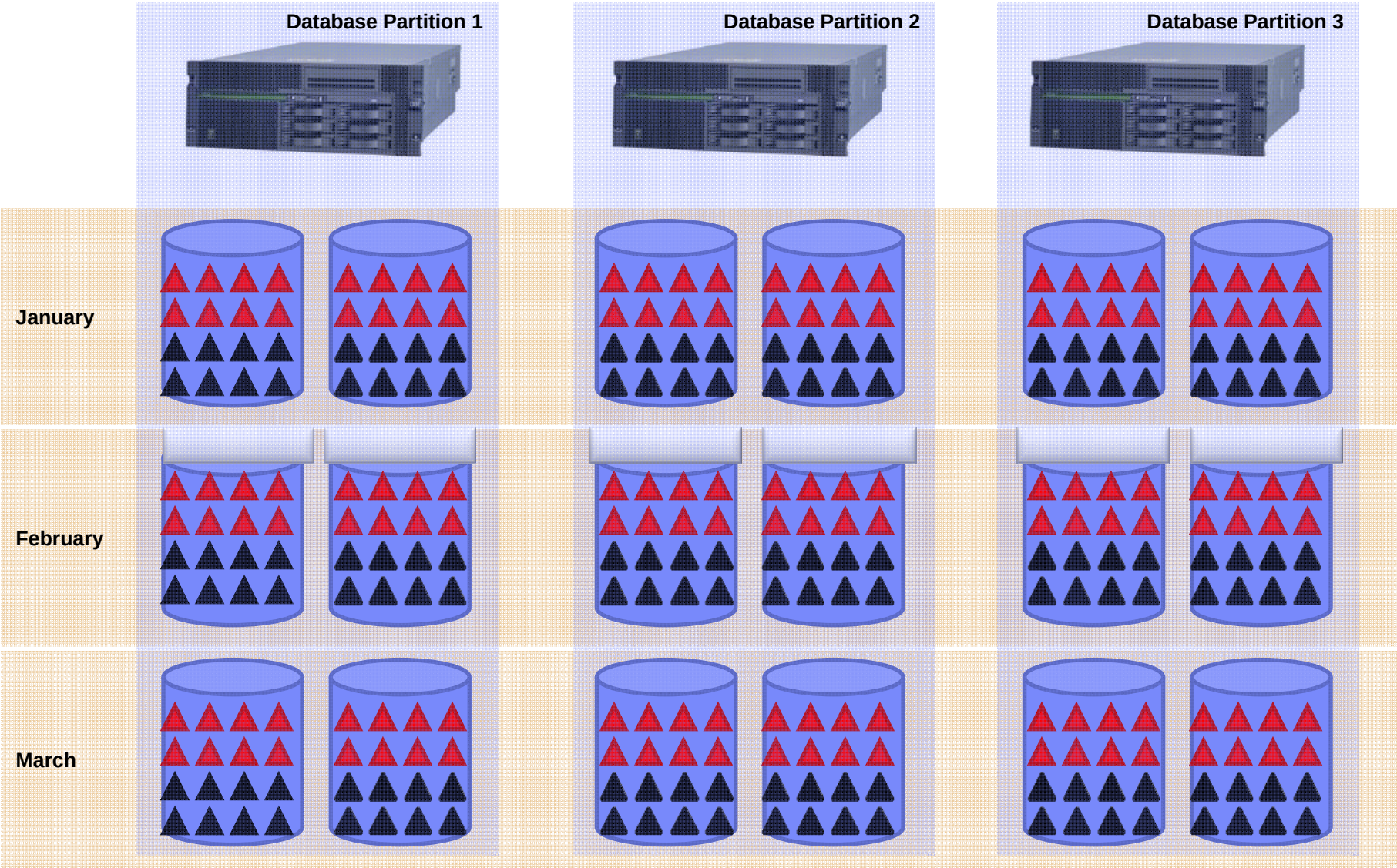
DB2 Database Partitioning Feature = Divide I/O



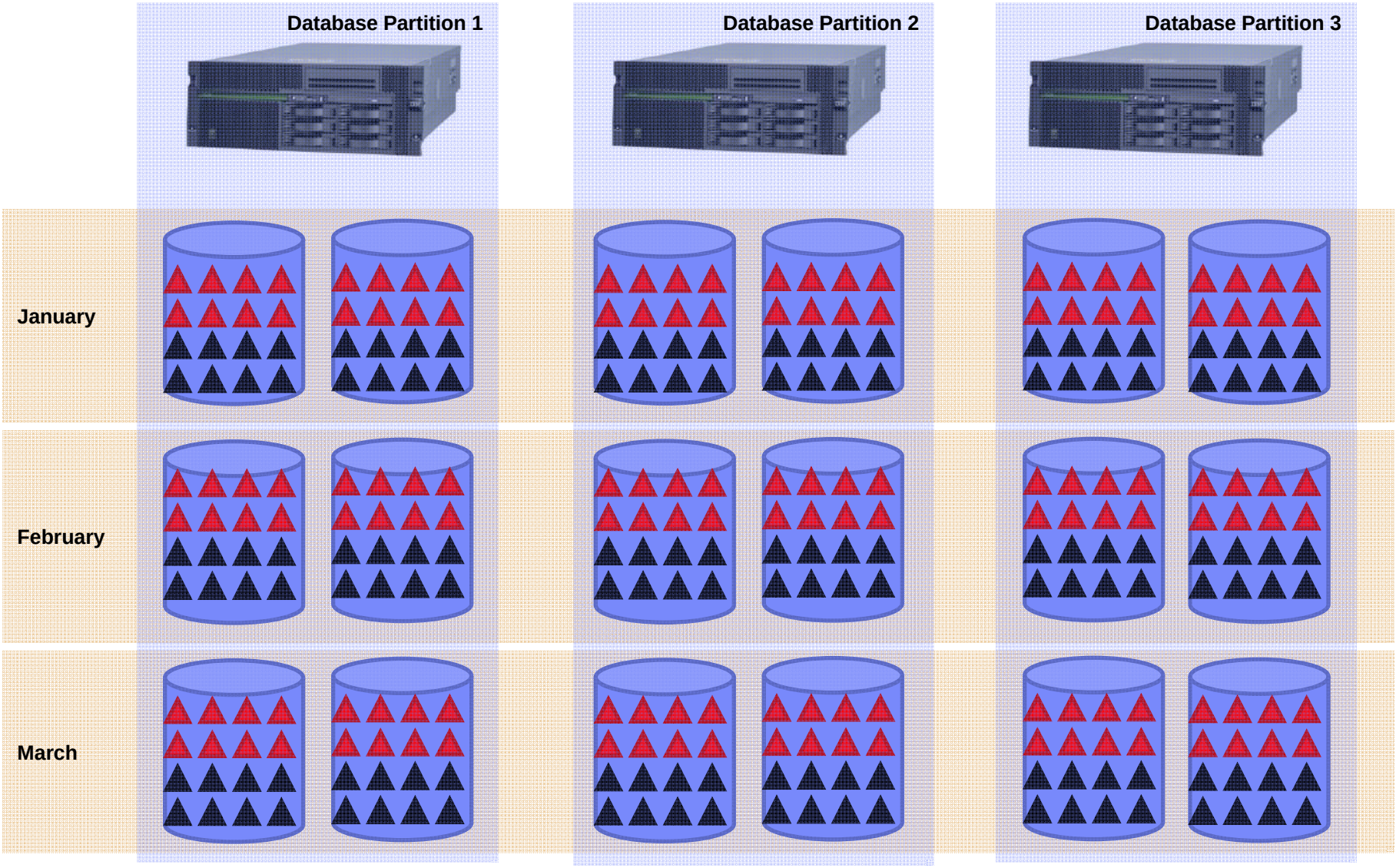
Add Range Partitioning to Further Reduce I/O



Add MDC to Further Reduce I/O



Compression Further Reduces I/O by a Factor of X



Questions and Thank You