Efficient Bulk Deletes for Multi Dimensional Clustered Tables in DB2

Bishwaranjan Bhattacharjee, Timothy Malkemus

IBM T.J. Watson Research Center

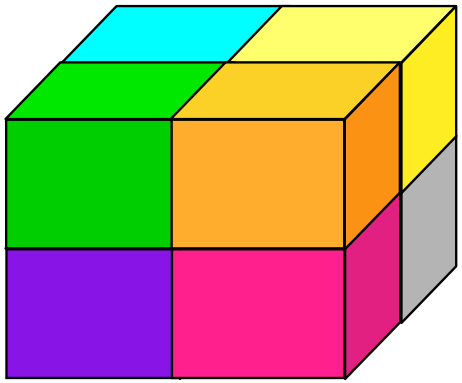
Sherman Lau, Sean McKeough, Jo-anne Kirton

Robin Von Boeschoten, John P Kennedy

IBM Toronto Laboratories

bhatta@us.ibm.com

Bulk Deletes
(aka Mass Delete, Rollout)



- *Frequent in Data Warehouses*
- *Often multi dimensional*
- *Maintenance windows for it are slowly diminishing*
- *Customers expect system availability when rolling out*

An online, multi dimensional rollout mechanism is very important for a db engine

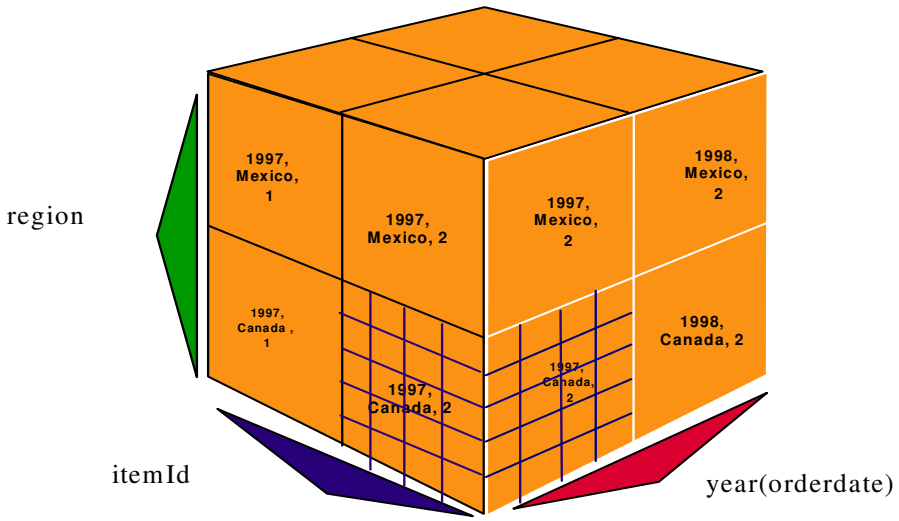
Major Issues In Rollout

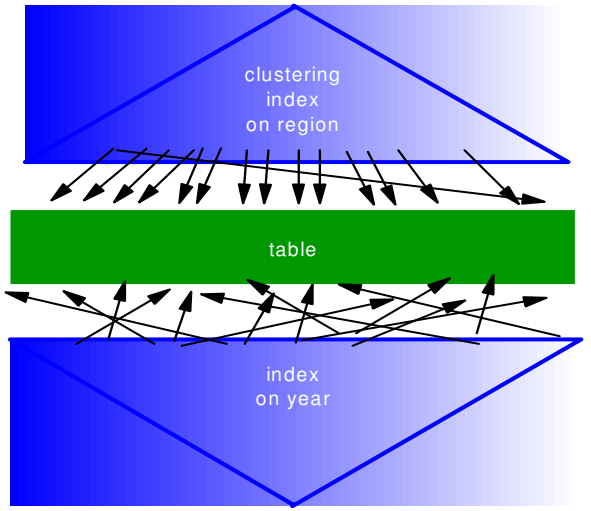
- *Response time of Rollout and Rollback : Maintenance windows shrinking*
- *Locks : Lock escalation a problem*
  - *Bad for concurrency*
  - *Impacts response time*
- *Logging : Complicates applications*
  - *Have to use Fetch First n Rows Only (FFnRO)*
  - *Bad for concurrency*
  - *Impacts response time*
- *Secondary Index Update :*
  - *Severely impacts response time*
  - *Consumes resources like log space, CPU*
  - *Results in lots of synchronous IO*

Road Map

- *Multi Dimensional Clustering (MDC) in DB2*
- *MDC Rollout*
- *Performance Evaluation of MDC Rollout*
- *Related Work*
- *Conclusion*

MDC Motivation: Multidimensionality





Single dimensional index clustering is inadequate

1. "*Efficient Query Processing for Multi-Dimensionally Clustered Tables in DB2.*", VLDB 2003
2. "*Multi-Dimensional Clustering: A New Data Layout Scheme in DB2*", SIGMOD 2003
3. "*Automating the design of multi-dimensional clustering tables in relational databases*", VLDB 2004
4. "*Predicate Derivation and Monotonicity Detection in DB2 UDB*", ICDE 2005
5. "*Performance Study of Rollout for Multi Dimensional Clustered Tables in DB2", EXPDB 2006*

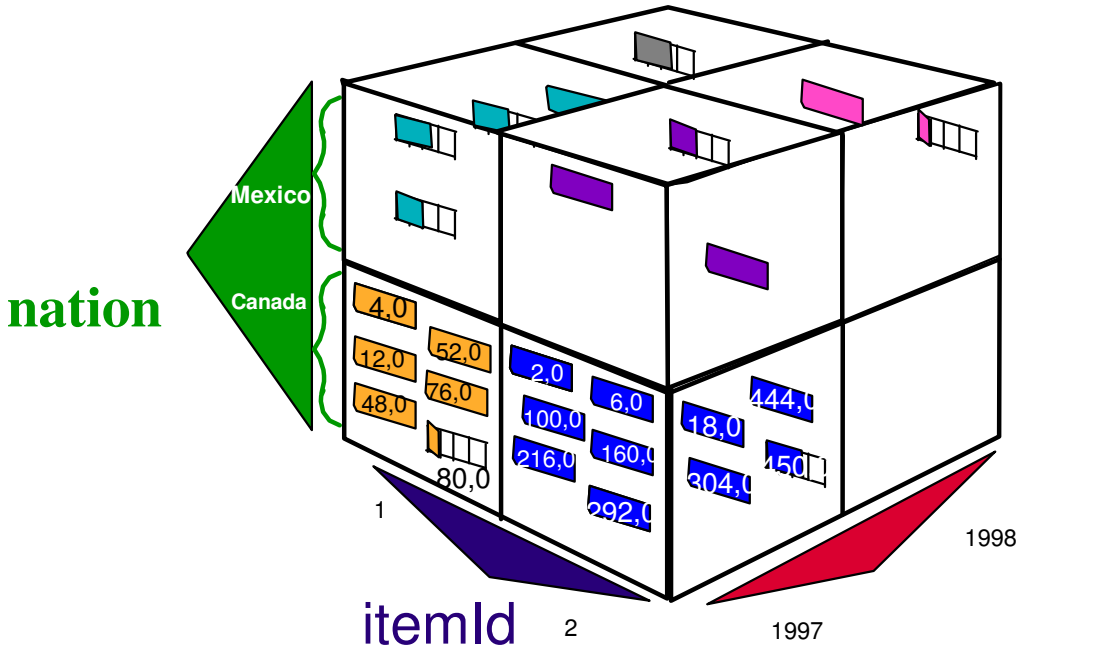
MDC Table Syntax

```
CREATE TABLE MDCTABLE (
    orderDate DATE,
    Nation CHAR(25),
    itemId  INT,
     ... )
    ORGANIZE BY( orderDate, Nation, itemId )

CREATE TABLE MDCTABLE2 (
    orderDate DATE,
    Nation CHAR(25),
    itemId  INT,
    orderYear generated always as ((INTEGER(orderDate)/10000),
    ... )
    ORGANIZE BY( orderYear, Nation, itemId )
```

* no need to plan for or define explicit range boundaries

How MDC Works : Blocks, Cells, Slices



Blocks

- Pages (2 - 256) of records
- BID (block id) = <first pool relative page of block, 0>

Block Indexes

- Structurally, B-Tree indexes..
- Key has list of BIDs
- Small compared to RID indexes

Cells

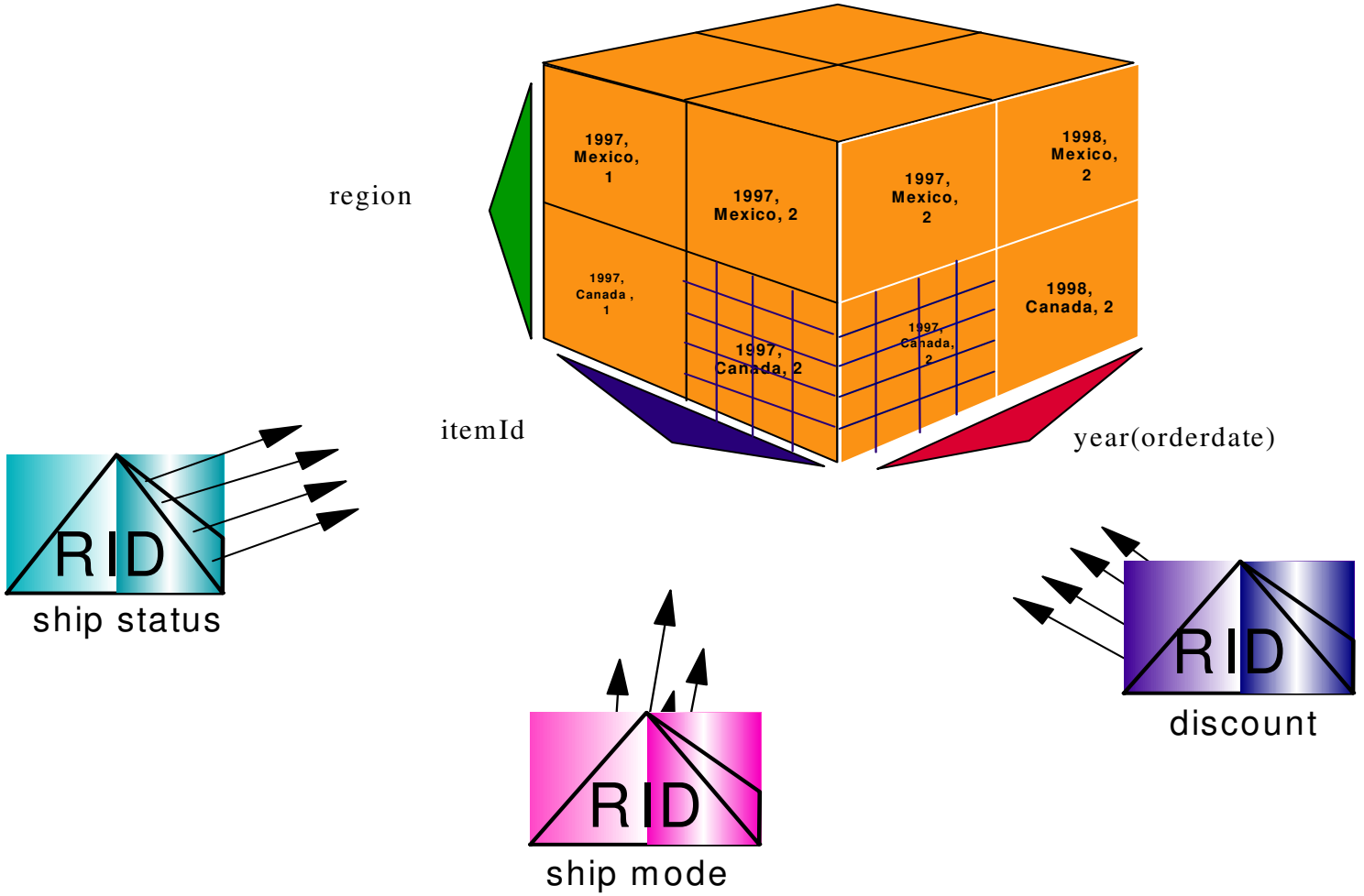
- 0 or more blocks
- Entry in composite block index when cell has blocks

Key for Canada:

Keypart

Canada	2,0	4,0	6,0	12,0	18,0	48,0	52,0	76,0	80,0	100,0	160,0	216,0	292,0	304,0	444,0	450,0

MDC Supports Additional RID Indexes



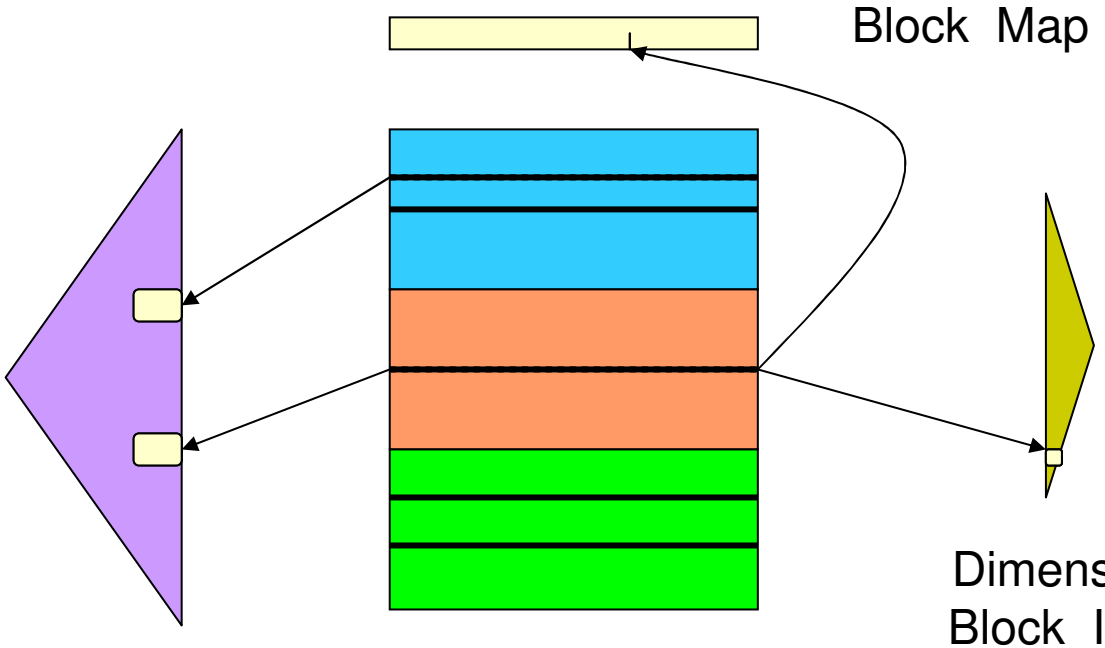
Please see previous papers for more details on MDC and MDC query performance

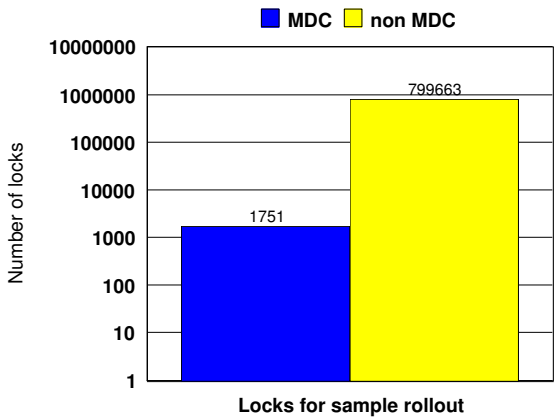
Road Map

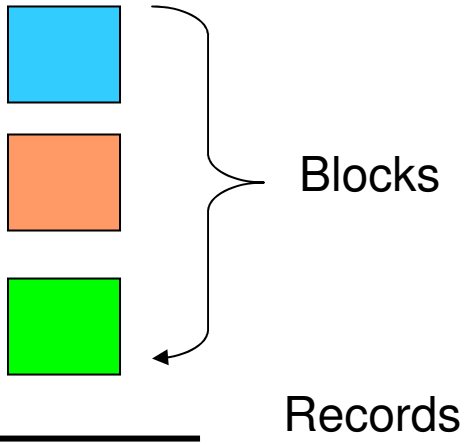
- *Multi Dimensional Clustering (MDC) in DB2*
- *MDC Rollout*
- *Performance Evaluation of MDC Rollout*
- *Related Work*
- *Conclusion*

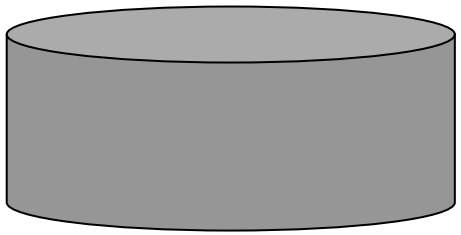
How MDC Delete Works

Only Block Locks For Full Cell Deletes



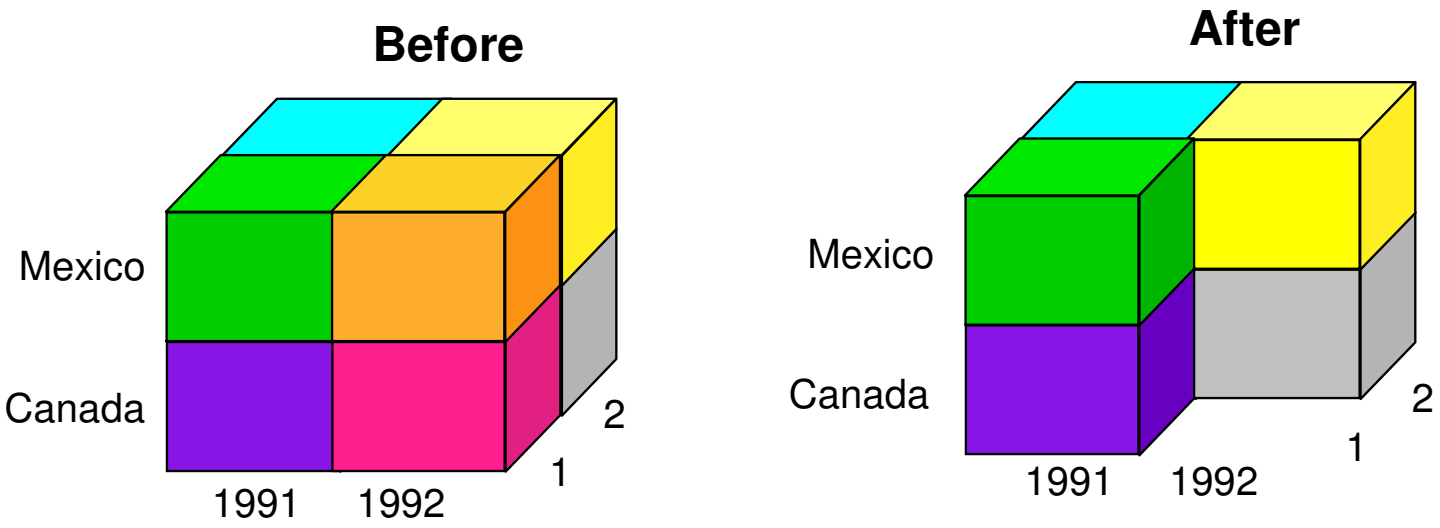




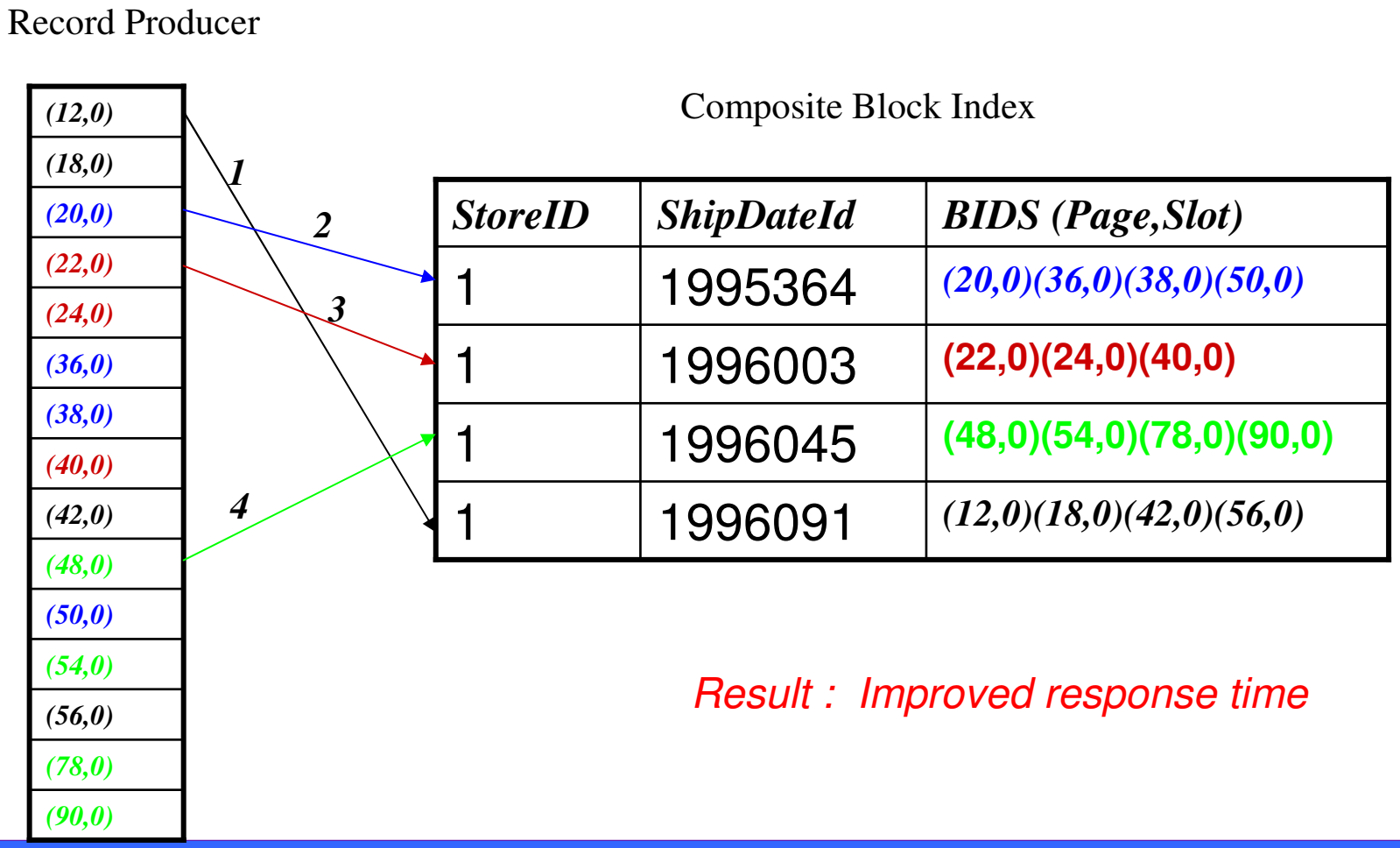


MDC Immediate Rollout *(GA DB2 V8.2.2)*

- Faster DELETE along cell or slice boundaries
- Compiler determines if DELETE statement qualifies for ROLLOUT
  - No need for a specialized statement or command
- Example: MDC table with 3 dimensions (nation, year, product ID)
  - DELETE FROM table WHERE year = 1992 and product_id = 1

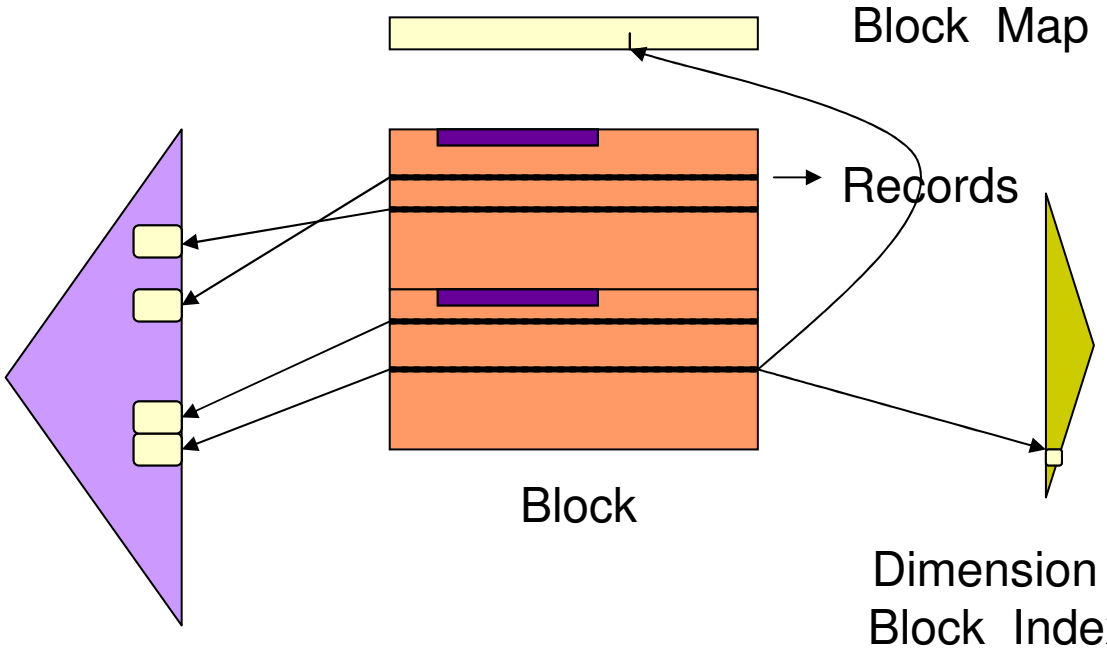


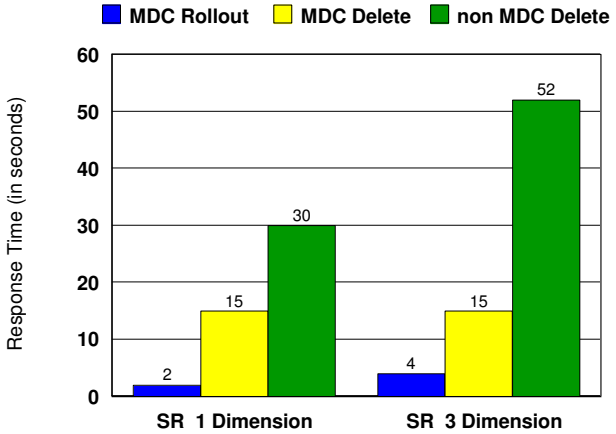
How MDC Immediate Rollout Works

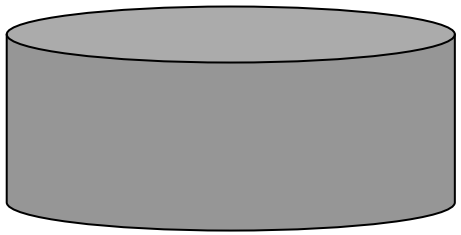


Result : Improved response time

How MDC Immediate Rollout Works In A Block







Impact:

Log space savings

Improved response time

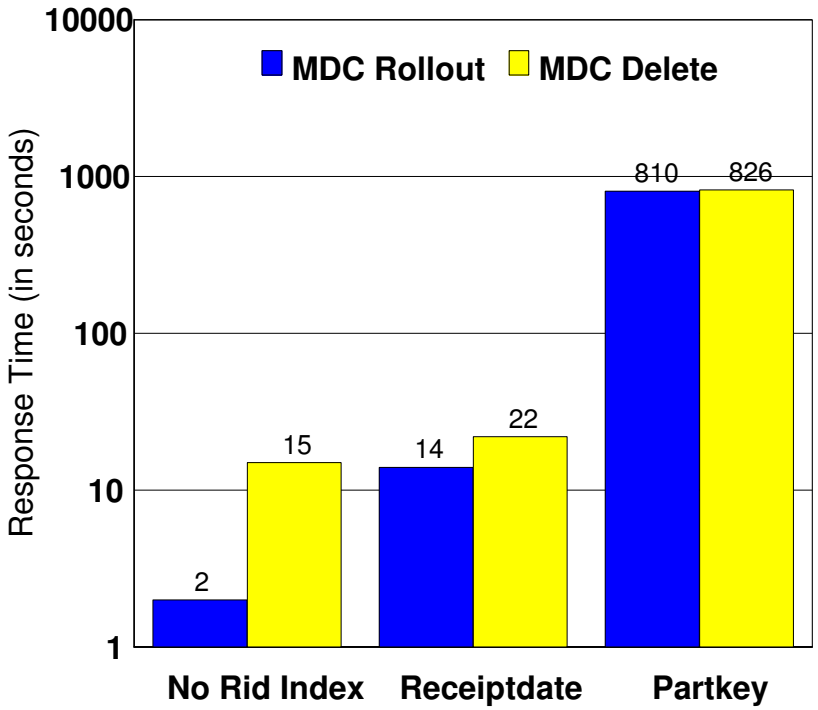
Same transaction does not reuse delete space before commit

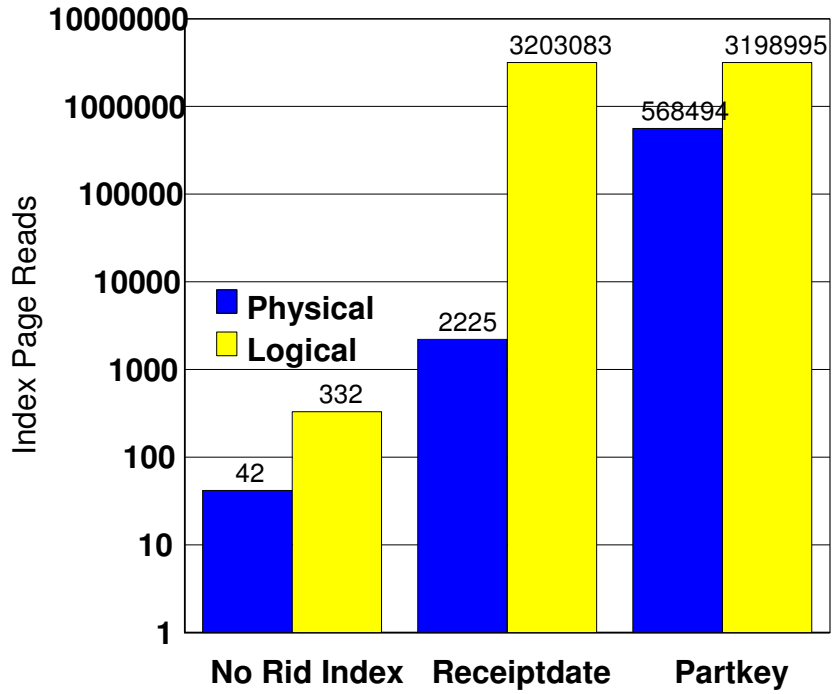
MDC Immediate Rollout Working Summary

- Performance improvements by avoiding per-row logging and by path length reduction
  - Clears the slot directory on each page in the block
  - Writes one small log record per page - rather than a log record per row (containing row data)

- *Secondary indexes still updated synchronously (immediately)*
  - *Must scan the rows (as usual) to update each index to remove keys*
  - *Index logging is unchanged*

Impact of RID Index Cluster Ratio On Immediate Rollout





Index Cluster Ratio *Receiptdate* : 38% *Partkey* : 4%

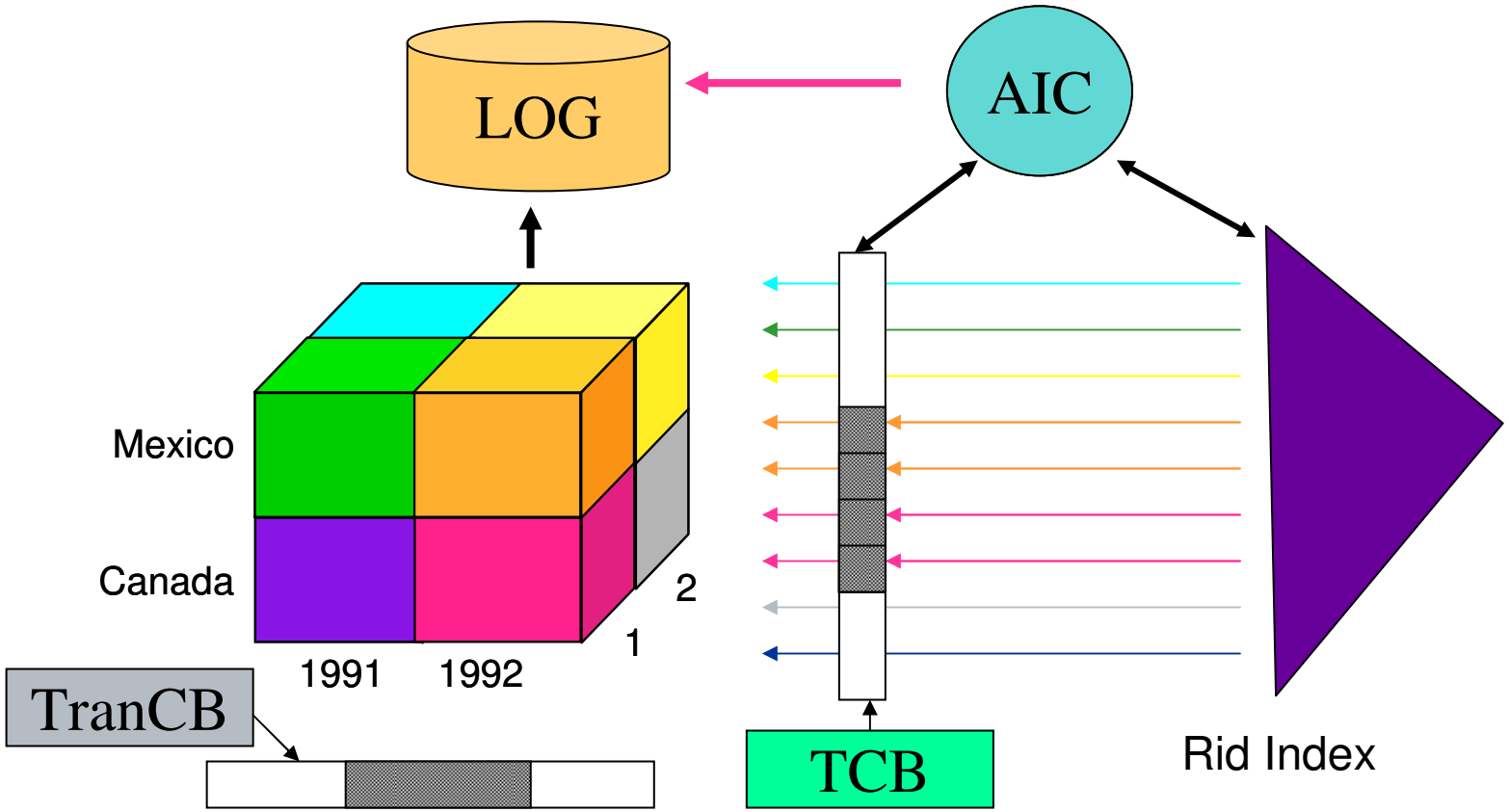
Comparison Of Logging Space Used In Immediate Rollout

Number of RID Indexes	Rollout (in Bytes)	Delete (in Bytes)	% Gain
0	668554	42635130	98
2	28898680	71118627	59
4	58441270	100960863	42

MDC Deferred Rollout Aims in DB2 Viper 2

- *Response time of Rollout with rid indexes to improve significantly*
- *Index page IO to reduce significantly*
- *Index log space requirement to come down significantly*
  - *Will simplify application logic. No need for FFnRO*
- *Table scanners and Block Index scanners will not be impacted*
- *Rid index scanners could be slowed down slightly*

MDC Deferred Rollout HLD

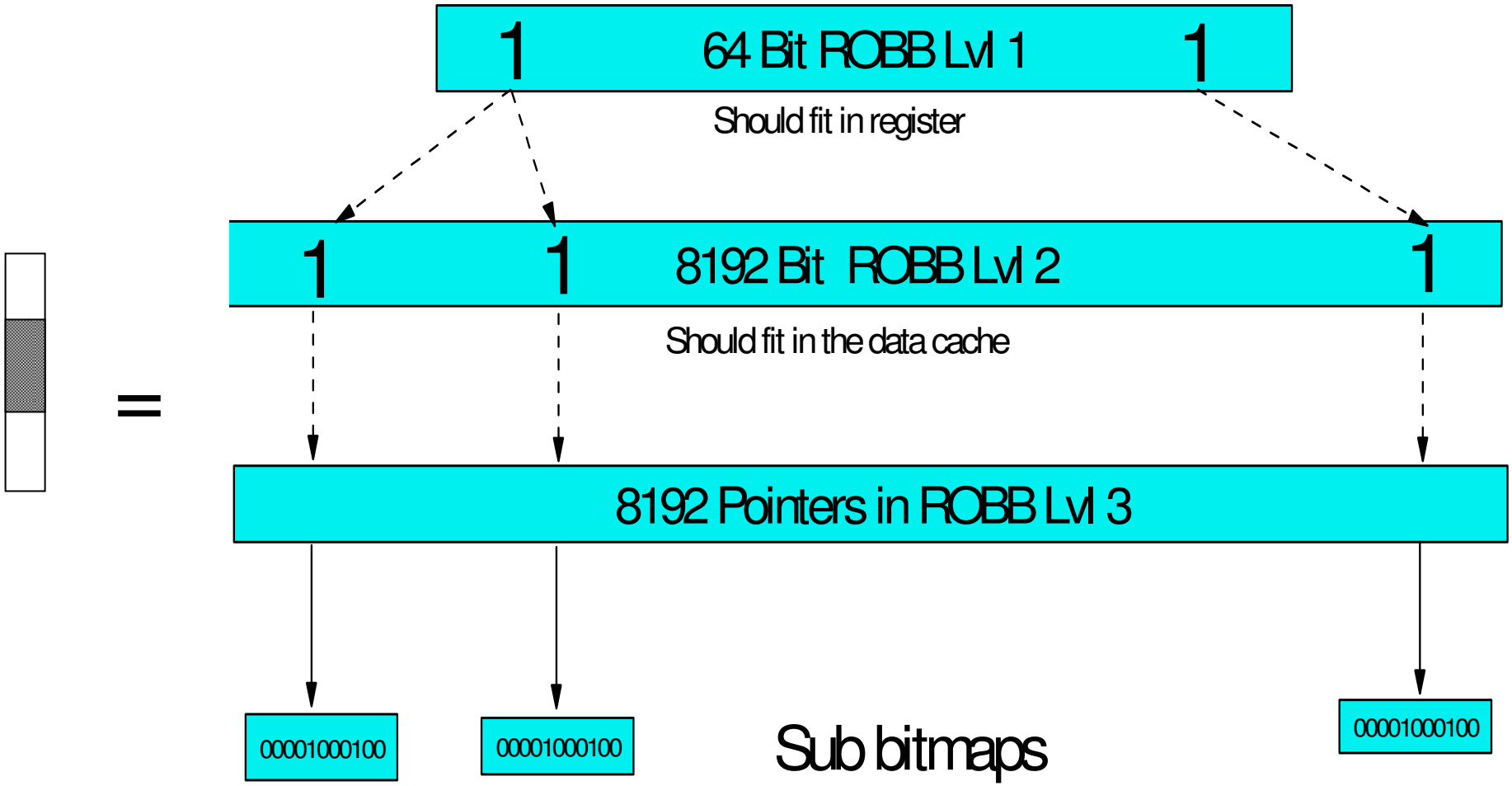


- *Rollout will not update any existing rid indexes*
- *Rid index cleanup to be performed by a asynchronous background cleaner*
- *Rid index scanners will check an in memory data structure to skip rolled out blocks*
- *AIC will write its resume position and log it. It can restart from that spot*

Rollout Block Bitmap (ROBB) Design Considerations

- *Fast Probes*
- *Memory Considerations*
- *Commit/Rollback Memory Restrictions*
- *ROBB Operations*
  - *Probe (Query, AIC)*
  - *Set and Clear (Rollout, AIC)*
  - *Merge and Subtract (Commit, Rollback)*
  - *Recreate (Recovery)*

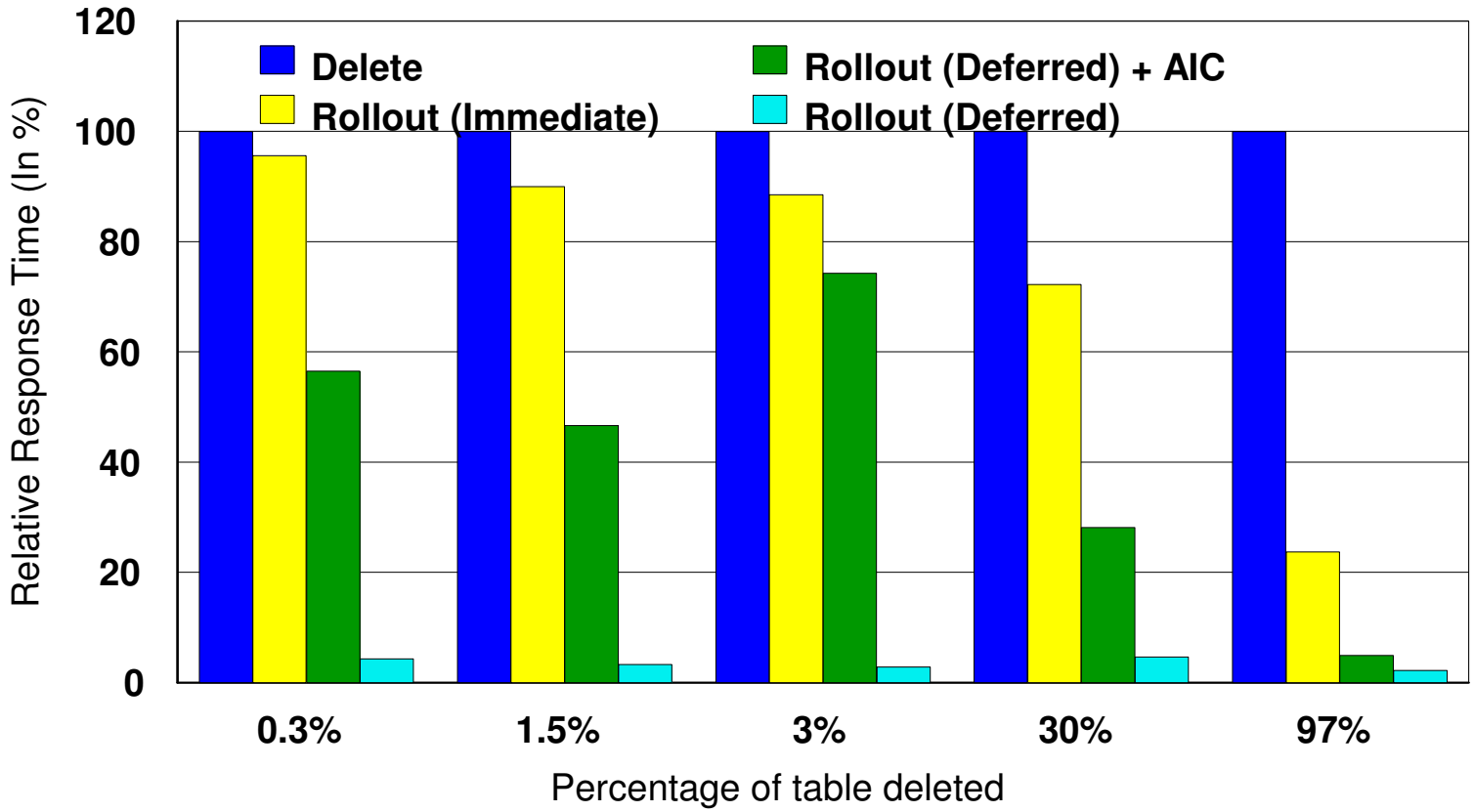
Example Rollout Block Bit Map (ROBB) Design



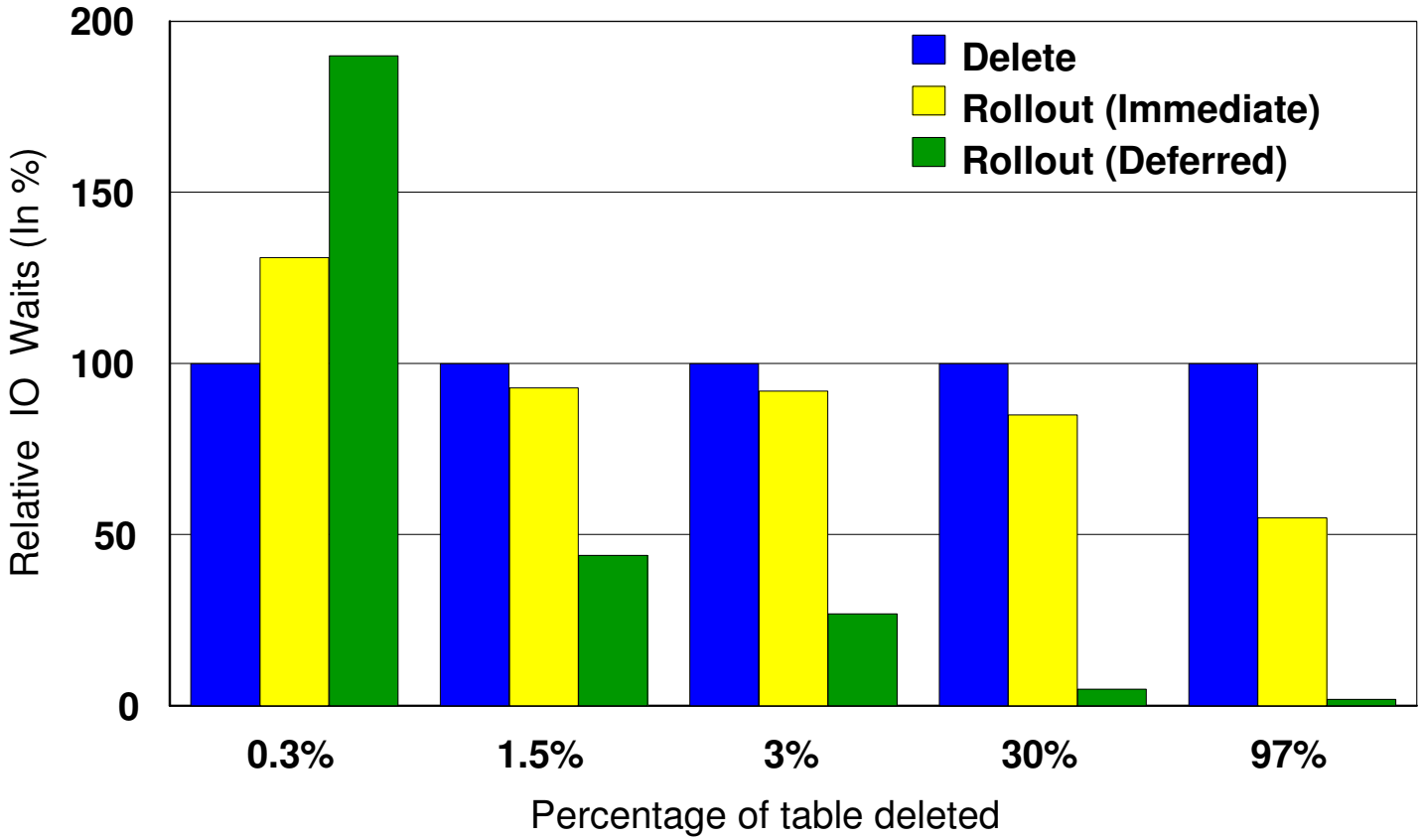
Performance Evaluation Of Rollout

- *Setup similar to that of some customers who run ERP over MDC tables*
- *Experimental Setup*
  - *DB2 UDB Viper 2*
  - *64 bit AIX 5.3.0.0*
  - *IBM 7028-6C4*
    - *16 GB of main memory*
    - *4 x PowerPC_POWER4 @ 1453 MHz*
  - *Table :*
    - *2 Dimensional MDC Fact Table*
    - *11 million rows in 134260 pages*
  - *Indexes :*
    - *9 RID Indexes*
      - *1 Unique RID index of 32716 pages and 8 Non Unique RID index of ~ 4700 pages each*
      - *3 Indexes with < 5% clustering, 2 Indexes with ~ 35% clustering and 4 Indexes with > 95% clustering*
    - *3 Block Indexes*

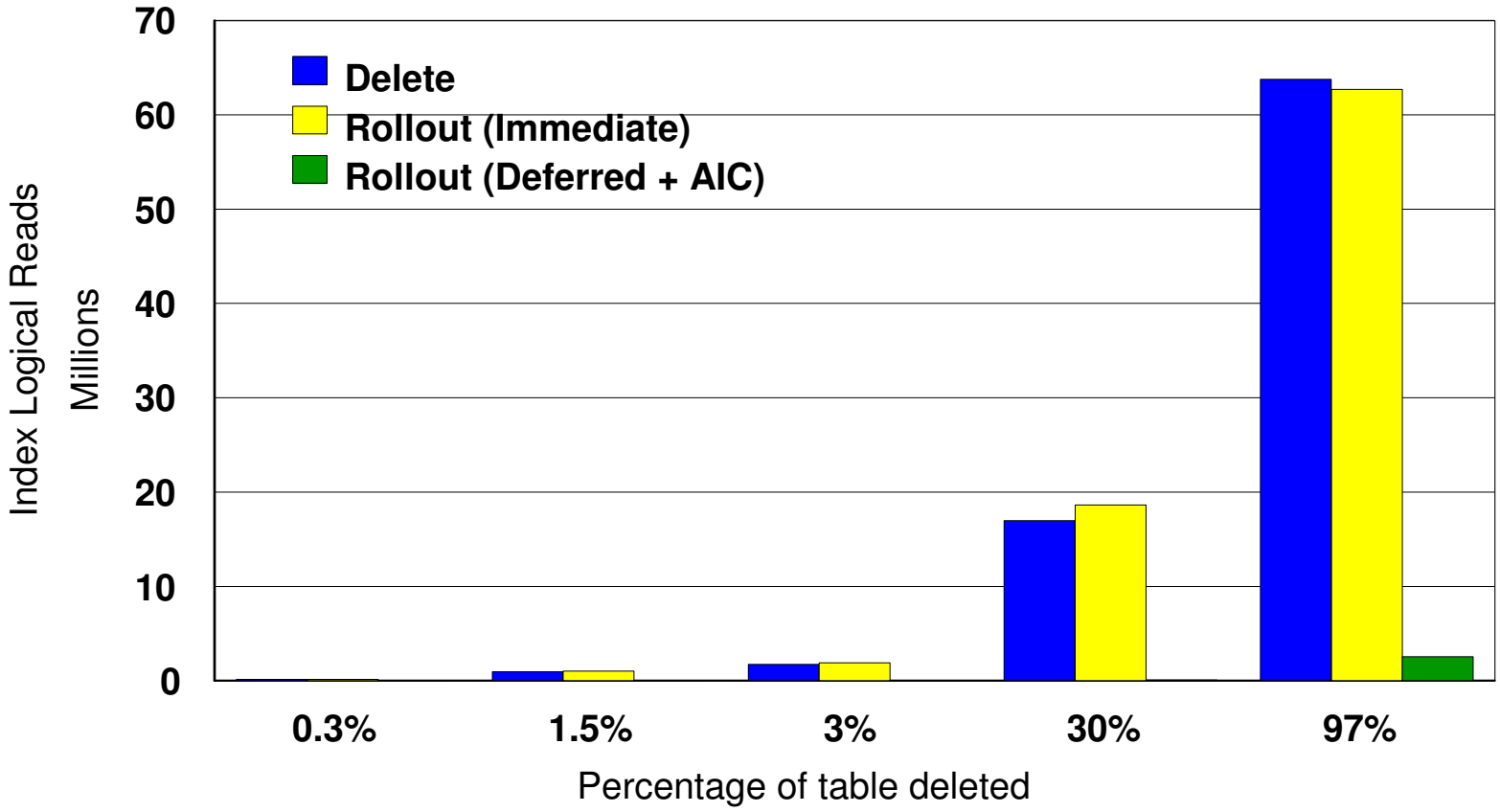
Response Time



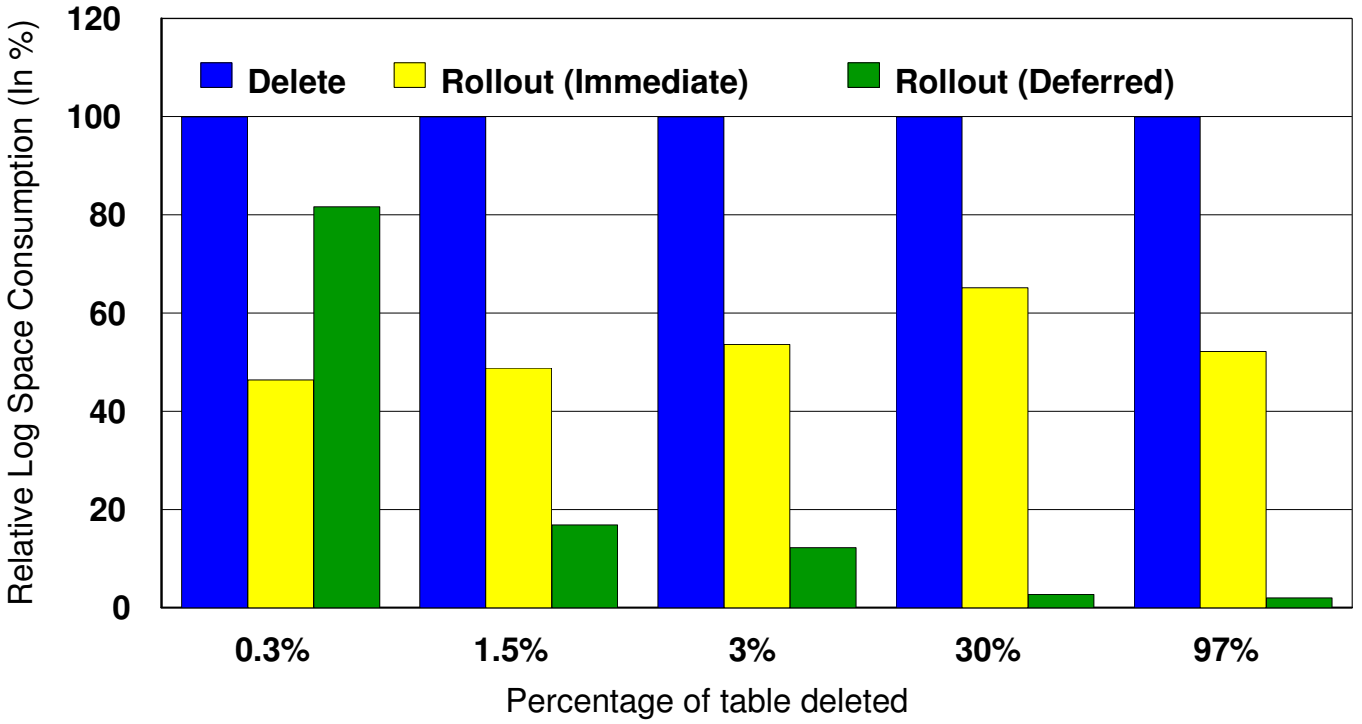
IO Waits Incurred



Index Logical Reads

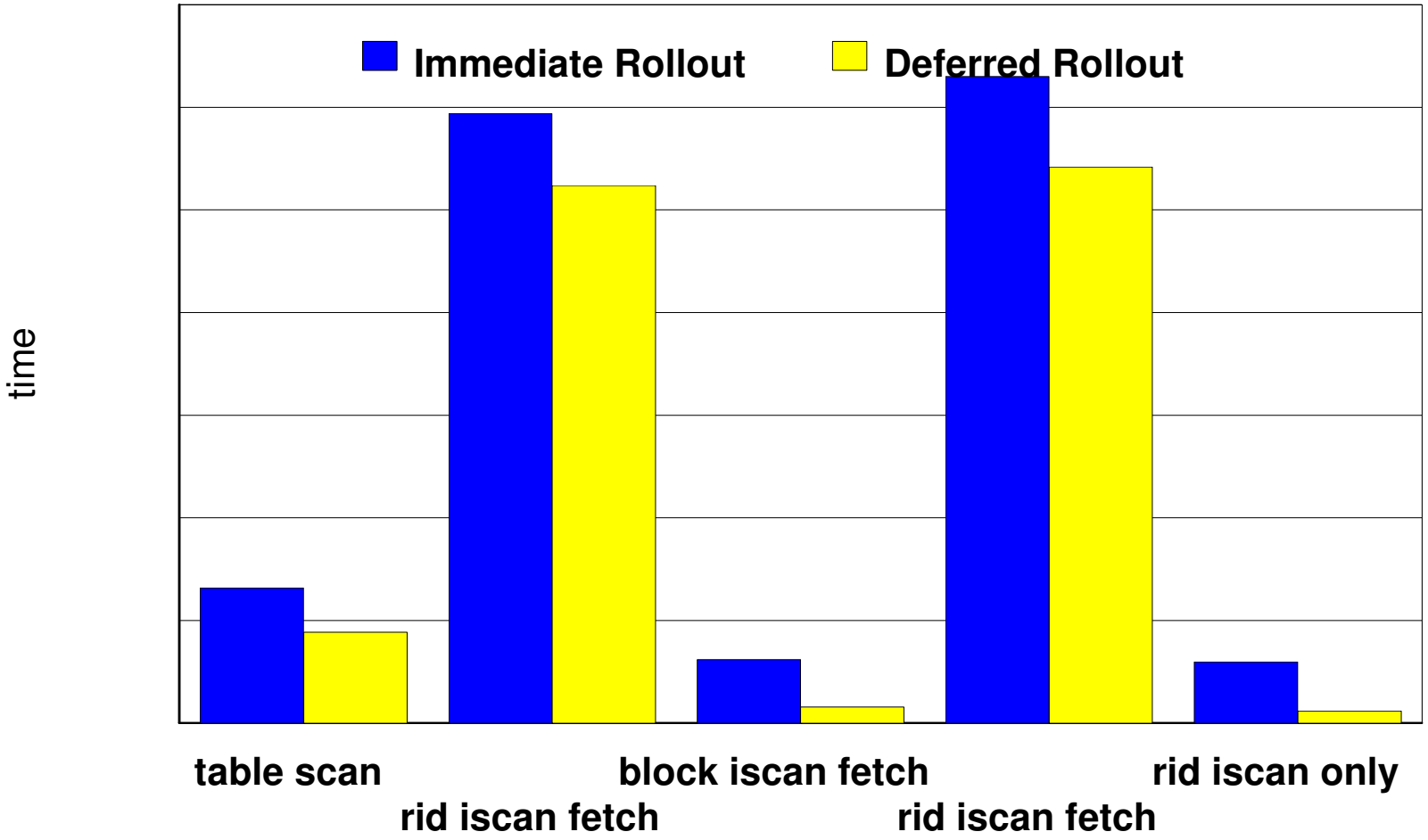


Log space consumption



Workload Performance

(start clock, delete, query, stop clock)



Related Work

- *Horizontal, record based with rid index update one at a time.*
  - *Not optimized for bulk delete*
- *Detach in Range Partitioning*
  - *Generally needs queries to drain*
  - *Special syntax (attach, detach)*
- *Deletes on B+ Tree Tables*
  - *Rid indexes could be deleted horizontally, in parallel in some implementation*
  - *"Online Bulk Deletion", Lilja et al, ICDE 2007*
    - *Optimize B+ Tree Table deletes*
- *Vertical Deletes*
  - *"Efficient Bulk Deletes in Relational Databases", Gartner et al, ICDE 2001*
    - *Assumes table will be x locked and indices would be offline for the delete*
    - *Addresses response time but not locking or logging*
- *Deferred Maintenance*
  - *"Differential Files: Their Application to the Maintenance of Large Databases", Severance et al, ACM TDBS 1971*
    - *Differential file used as a book errata list to identify and collect pending record changes*
    - *When Differential file gets large, reorganization will incorporate changes into the database*

Conclusion

- *MDC Rollout provides a mechanism for mass delete of data which*
  - *Is able to significantly reduce the response time of mass deletes compared to previous deletes in DB2. Even when a lot of badly clustered rid indexes are defined on the table*
  - *While consuming significantly lower amount of system resources (locking, logging, IO) compared to a previous delete in DB2*
- *In this talk we described how these challenges were addressed*