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
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:

<table>
<thead>
<tr>
<th>Keypart</th>
<th>2,0</th>
<th>4,0</th>
<th>6,0</th>
<th>12,0</th>
<th>18,0</th>
<th>48,0</th>
<th>52,0</th>
<th>76,0</th>
<th>80,0</th>
<th>100,0</th>
<th>160,0</th>
<th>216,0</th>
<th>292,0</th>
<th>304,0</th>
<th>444,0</th>
<th>450,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

- Dimension Block Index
- Block Map
- Table
- Log
- Blocks
- Records

Rid Index (optional)

Number of locks:
- MDC
- non MDC

- Locks for sample rollout

© 2007 IBM Corporation
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

Record Producer

<table>
<thead>
<tr>
<th>(12,0)</th>
<th>(18,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,0)</td>
<td>(22,0)</td>
</tr>
<tr>
<td>(24,0)</td>
<td>(36,0)</td>
</tr>
<tr>
<td>(38,0)</td>
<td>(40,0)</td>
</tr>
<tr>
<td>(42,0)</td>
<td>(48,0)</td>
</tr>
<tr>
<td>(50,0)</td>
<td>(54,0)</td>
</tr>
<tr>
<td>(56,0)</td>
<td>(78,0)</td>
</tr>
<tr>
<td>(90,0)</td>
<td></td>
</tr>
</tbody>
</table>

Composite Block Index

<table>
<thead>
<tr>
<th>StoreID</th>
<th>ShipDateId</th>
<th>BIDS (Page,Slot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1995364</td>
<td>(20,0)(36,0)(38,0)(50,0)</td>
</tr>
<tr>
<td>1</td>
<td>1996003</td>
<td>(22,0)(24,0)(40,0)</td>
</tr>
<tr>
<td>1</td>
<td>1996045</td>
<td>(48,0)(54,0)(78,0)(90,0)</td>
</tr>
<tr>
<td>1</td>
<td>1996091</td>
<td>(12,0)(18,0)(42,0)(56,0)</td>
</tr>
</tbody>
</table>

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

Response Time (in seconds)

Index Cluster Ratio

Receiptdate : 38%
Partkey : 4%
Comparison Of Logging Space Used In Immediate Rollout

<table>
<thead>
<tr>
<th>Number of RID Indexes</th>
<th>Rollout (in Bytes)</th>
<th>Delete (in Bytes)</th>
<th>% Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>668554</td>
<td>42635130</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>28898680</td>
<td>71118627</td>
<td>59</td>
</tr>
<tr>
<td>4</td>
<td>58441270</td>
<td>100960863</td>
<td>42</td>
</tr>
</tbody>
</table>
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 just update any existing rid indexes to delete the table records and update the block indexes.
- AIC will prefetch index leaf pages.
- Rid index cleanup to be performed by an asynchronous cleaner.
- One cleaner per rid index. Will become active when block needs cleaning.
- 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

- **64 Bit ROBB Lvl 1**
  - Should fit in register

- **8192 Bit ROBB Lvl 2**
  - Should fit in the data cache

- **8192 Pointers in ROBB Lvl 3**

Sub bitmaps:
- 00001000100
- 00001000100
- 00001000100
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

Percentage of table deleted

- Delete
- Rollout (Immediate)
- Rollout (Deferred) + AIC
- Rollout (Deferred)
IO Waits Incurred

![Chart showing IO waits incurred for different percentages of table deleted.]

- **Delete**
- **Rollout (Immediate)**
- **Rollout (Deferred)**
Index Logical Reads

Percentage of table deleted

Index Logical Reads

 Millions

Delete
Rollout (Immediate)
Rollout (Deferred + AIC)

© 2007 IBM Corporation
Log space consumption

![Bar chart showing log space consumption with different percentages of table deletion. The chart compares 'Delete', 'Rollout (Immediate)', and 'Rollout (Deferred)' scenarios. The y-axis represents the relative log space consumption in %, and the x-axis represents the percentage of table deleted, ranging from 0.3% to 97%. The chart indicates that the relative log space consumption increases as the percentage of table deleted increases.](chart.png)
Workload Performance
*(start clock, delete, query, stop clock)*

![Chart showing Immediate Rollout and Deferred Rollout comparisons for various operations like table scan, block iscan fetch, rid iscan fetch, and rid iscan only.](chart.png)
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*