In-memory Tables

Larry Strickland
Chief Product Officer
Abstract

In-Memory tables have the promise of drastically improving the performance of an application, so much so, that previously rejected application designs may become possible. This presentation will introduce In-Memory tables; how to find applications that may benefit from In-Memory tables; how to adapt applications to use In-Memory tables; and, discuss what needs to be considered when designing a system to manage the In-Memory tables. Finally some real world results will be presented showing performance improvements as a result of the use of In-Memory tables.
• Introduction to IN-MEMORY tables
• Designing IN-MEMORY tables
• Where IN-MEMORY tables are most useful
• Some performance results
• Adapting for IN-MEMORY
• Identifying IN-MEMORY table Candidates
• Summary
Introducing IN-MEMORY tables
IN-MEMORY – Shortest Path to Data

• Obtaining the shortest path to Data is essential in improving performance of applications

• There are two major components to that path
  1. Accessing data from permanent storage
  2. Executing the machine instructions necessary to find and retrieve the right data
Putting data in memory

- Clearly putting data into memory can improve the performance

- There are a number of methodologies that have achieved this:
  - In memory tables
  - Data buffers
  - Caches
• Even after data has been moved to some sort of storage in memory the access path can still be very long:
  • Often the same code is used for all sorts of access, including on disc. The large number of possible branches increases the code length.
• Only systems designed to be optimized for in memory access are going to reduce the code path.
IN-MEMORY table manager

Shortest and fastest path to data

Which path would you prefer?
Designing IN-MEMORY tables
Designing in-memory system

- Memory Control Block
- One or more tables
- Fixed length rows
- Table Control Block
- One or more indexes
- Arbitrary location, but continuous key
- Rows
- Index
- Allocated Memory Block
Multiple Organizations/Retrieval

- Organizations
  - Sequential (asc, desc)
  - Hash
  - Random
  - User Defined Sequence
  - B-tree

- Retrieval Methods
  - Serial
  - Binary
  - Queued Sequential
  - Bounded Binary
  - Hash

Creating the index

Using the index
Hash Organization and Retrieval

- Hash Organization and Retrieval uses a hash algorithm – many algorithms available.
  - Ideally looking for low collisions, and not too sparse an index – there is much in the literature about the best algorithms for this.
- However, when you are looking for speed:
  - The time to calculate the value on retrieval is very important.
  - May need to use less efficient algorithm from a collision point of view, to maintain speed.
Fixed row length means very fast access:
  - row_start_address = data_start_address + (row_number-1) * row_length
  - key_position = row_start_address + key_offset
  - Index holds key_position

Reads – very fast

Updates
  - If key changes – need to recalculate index (in part)

Insert
  - Add row to end of row data (or unused row)
  - Need to extend indexes – but can take advantage of block moves

Deletes
  - Similar to Insert – simpler if memory is not recovered
• Avoid I/O
• Avoid OS services in general
• Avoid getmains
• Avoid locking (assumption of mostly read only)
• Use Cookies to get to existing position (index entry) – for subsequent calls
• Implicit open (assumed tables are backed)
• Consider page size (multiples of 4k work well)
Shareable IN-MEMORY tables

- Eliminate the loading of local data space on initialization
- Ensure all applications have identical data
- Share data between applications
- Overcome processing bottlenecks between applications
- Use tables as a write through cache, message queues
- Tables in a shared data space outlive their transaction or application
Where to use IN-MEMORY tables
Where is this most useful?

Three common scenarios:

1. Temporary data
2. Frequently read data
   (note, this is frequently read rows, not frequently read tables)
3. Rules Engines
   Replacing the if...then...else with table lookups
• Temporary Data Tables
  • Load all data before generating indexes
  • Use indexes to perform virtual sorts, and data organization

• Examples:
  • Replace Cobol SORT or Sort All
  • Build reports for display, such as consolidated bank statements
Frequently Read Data

• Data that is read the most frequently provides the greatest opportunity for improvement

• Consider
  • A credit card transaction system of a billion transactions a day, with 100 different card types.
  • During reconciliation, each transaction is read once, however, each row in the card type table is read 10 million times (on average)
  • The card type table will benefit greatly from optimized in-memory access
  • The transaction table won’t!
Reference data
- Is 5-15% of your total data
- Changes infrequently
- Is accessed often, may represent as much as 80% of your accesses

Temporary data
- Is created, processed and then deleted
- Generates a high volume of data accesses for the volume of data

Remaining data
- The largest volume of data
- Read often followed by a write
- The lowest number of accesses
Rules Engines

- Moving logic from programs into rules based tables is great for flexibility, however it can really hurt performance.
- If rules tables are used appropriately with optimized in-memory methods, both speed and flexibility is possible.
- Example construct:
  - Inputs are converted to a vector.
  - Vector is used as the key to rules table (matching zero or more).
  - Inputs are given to programs identified by matching rules (could be a start address).
    - May be in parallel or series.
    - Can be iterative (the output of a program goes back into the rules engine).
IN-MEMORY some performance results
CICS supports CICS managed tables (CMDT) backed by a VSAM data set

For this test:
• The VSAM data set (of 5000 records for this test) was loaded into CICS managed tables in memory and measurements taken
• The same 5000 records were loaded into IN-MEMORY tables and measurements taken
• There was no buffering optimization for the VSAM KSDS file.
• Note that CMDT retrieves data only by key

Retrieving data from IN-MEMORY tables is faster then CMDT
For this test:

- The VSAM files are KSDS files buffered with Batch LSR. The elapsed time to access the data from IN-MEMORY tables includes the time to load the IN-MEMORY tables.
- 999,900 records in VSAM KSDS file
- 999,300 records were retrieved 50 times
- It takes fewer EXCPs to load the same amount of data into IN-MEMORY tables as it does to load the data into VSAM direct access.
- The percentage reduction in elapsed time is more significant as you increase the number of retrievals because the elapsed time includes the time taken to load the IN-MEMORY tables.

The speed of direct VSAM access was compared with IN-MEMORY access.

The more data retrieved from IN-MEMORY tables, the greater the benefit.
Faster than DB2 buffer pools

- Sequential Browse: Up to 9 times greater
- Random Access (hash): Up to 23 times greater
- Random Access (sequential): Up to 19 times greater
- Populate Tables: Up to 30 times greater

Number of Reads / Inserts per Second

- CPU Seconds Consumed
  - 86% less
  - 95% less
  - 91% less
  - 93% less

In-memory tables vs. DB2 with buffer pooling

© 2014 DataKinetics Ltd. All rights reserved.
Adapting for IN-MEMORY tables
Example

4 tables joined; two transactional tables and two reference tables

```sql
SELECT T2.LAST_NAME, T2.FIRST_NAME, T1.BALANCE_DUE, T3.ZIP_CITY, T4.STATE_NAME
INTO LAST-NAME, FIRST-NAME, BALANCE-DUE ZIP-CITY STATE-NAME
FROM CUSTACCT T1,
     CUSTINFO T2,
     ZIPLIST T3,
     STATELIST T4
WHERE T1.CUSTID = T2.CUSTID
AND T2.ZIP5 = T3.ZIP5
AND T2.STATE = T4.STATE;
```

```sql
EXEC SQL
SELECT T2.LAST_NAME, T2.FIRST_NAME, T1.BALANCE_DUE, T2.ZIP5, T2.STATE
INTO LAST-NAME, FIRST-NAME, BALANCE-DUE, T2-ZIP5, T2-STATE
FROM CUSTACCT T1,
     CUSTINFO T2
WHERE T1.CUSTID=T2.CUSTID
END-SQL
```

Call API using TB_PARM
TB-ZIPLIST-CMDAREA
TB-ZIPROW
T2-ZIP5
Move TB-ZIP-CITY to ZIP-CITY

Call API using TB_PARM
TB-STATE-CMDAREA
TB-STATEROW
T2-STATE
Move TB-STATE-NAME to STATE-NAME
EXEC SQL DECLARE CUR01 CURSOR FOR
SELECT A.ACCT_NBR
FROM DKLDB001.USB_ACCOUNT A,
     DKLDB001.USB_PRODUCT P
WHERE A.CLNT_ID = :W-CLNT-ID
AND A.BNK_NBR = :W-BNK-NBR
AND A.AGT_NBR = :W-AGT-NBR
AND A.PRODUCT_ID = P.PRODUCT_ID
AND A.BNK_NBR = P.BNK_NBR
AND P.CARD_TYP_CDE = :W-CARD-TYP-CDE
FOR FETCH ONLY
END-EXEC.
Example - After

```sql
EXEC SQL DECLARE CUR01 CURSOR FOR
    SELECT ACCT_NBR, CLNT_ID, BNK_NBR, PRODUCT_ID
    FROM DKLDB001.USB_ACCOUNT
    WHERE CLNT_ID = :L-CLNT-ID
    AND BNK_NBR = :L-BNK-NBR
    AND AGT_NBR = :L-AGT-NBR
    FOR FETCH ONLY
END-EXEC.
```

```plaintext
*****************************************************************************

MOVE PRODUCT-ID TO IN-MEMORY-PRODUCT-ID.
MOVE L-BNK-NBR TO IN-MEMORY-BNK-NBR.
MOVE L-CARD-TYP-CDE TO IN-MEMORY-CARD-TYP-CDE.
CALL ‘API’ USING W-IN-MEMORY-PARM
W-IN-MEMORY-COMMAND-AREA
IN-MEMORY-PRODUCT-REC.
```
Example – CPU Consumption

- DB2 CPU Seconds
- SP CPU Seconds
- Total CPU Seconds

- With in-memory tables
- Without in-memory tables
IN-MEMORY usage

• Replacing CICS tables and TS queues with IN-MEMORY tables.

• Replace hard coded module tables with IN-MEMORY tables.

• Replace VSAM (KSDS and RRDS) files with IN-MEMORY tables.

• Unload high referenced DB2 table data into IN-MEMORY tables.

• Replace BDAM files with IN-MEMORY tables.

• Use IN-MEMORY tables for summarizing and grouping data for reports.

• IN-MEMORY is used for storing cursor results for DB2 cursor scrolling in CICS.

• IN-MEMORY can be used to provide session continuation data for online applications.
How to identify potential candidates
Tools

• SMF records
  – VSAM – SMF64 records
  – Jobs SMF 30 subtype 4
  – IAM – IAM SMF Records - (IAMSMTFVS IAM EXCP Report)
• DB2
  – SQL query against SYSTABLES & SYSTABLESPACESTATS
  – Vendor tools, APPTUNE or DETECTOR
• IMS – Change accumulation statistics
• CICS – Shutdown statistics
• Measurement tools
  – STROBE
  – APA
  – Monitors
## Criteria for identifying tables to be optimized

<table>
<thead>
<tr>
<th>Item</th>
<th>Criteria</th>
</tr>
</thead>
</table>
| 1    | The ratio of Reads to Writes needs to be high, generally a ratio of 100:1 is good and 500:1 would be great. This applies to:  
  - DB2  
  - IMS  
  - VSAM |
<p>| 2    | The ratio of Reads to Rows in the table needs to be good |
| 3    | The reads per day should be significant |
| 4    | Limit table size. <em>Consider Memory required (real or virtual). Normally smaller tables gain more.</em> Say &lt;2G |
| 5    | GLOBS are generally bad |</p>
<table>
<thead>
<tr>
<th>Data Set name</th>
<th>JOB NAME</th>
<th>DD NAME</th>
<th>EXCPS</th>
<th>INSERTS</th>
<th>DELETES</th>
<th>UPDATES</th>
<th>RETRIEVALS</th>
<th># RECORDS</th>
<th>LRECL</th>
<th>GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKL FE.P.LPS.APOR.DAT A</td>
<td>CICSP6P3</td>
<td>APOR</td>
<td>24,883,173</td>
<td>1,123</td>
<td>940</td>
<td>41,308</td>
<td>121,193,241</td>
<td>52,946</td>
<td>600</td>
<td>0.029586</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.N127XFD.R.A</td>
<td>ZOODZICA</td>
<td>N127XFD</td>
<td>28,821,494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>114,137,907</td>
<td>231,896</td>
<td>4089</td>
<td>0.883102</td>
</tr>
<tr>
<td>DKL XW.P.ENTR.DX.VPF.DK</td>
<td>XWSRVP</td>
<td>SYS00001</td>
<td>2,319,082</td>
<td>95,220</td>
<td>163,653</td>
<td>70</td>
<td>104,925,834</td>
<td>585,381</td>
<td>136</td>
<td>0.074144</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.PE01RCR.X00001</td>
<td>ZOODZIAA</td>
<td>PE01RCRX</td>
<td>26,092,817</td>
<td>10,561</td>
<td>2</td>
<td>0</td>
<td>92,958,864</td>
<td>2,952,090</td>
<td>18</td>
<td>0.049488</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.PG15IDF.X00021</td>
<td>ZOODZIGA</td>
<td>PG15IDFX</td>
<td>32,675,941</td>
<td>97,947</td>
<td>73,731</td>
<td>0</td>
<td>88,488,668</td>
<td>6,452,284</td>
<td>38</td>
<td>0.228348</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.E10FRP.M.A00015</td>
<td>ZOODZIAA</td>
<td>E10FRPMA</td>
<td>3,180,727</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68,269,497</td>
<td>130,461</td>
<td>8185</td>
<td>0.994488</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.E104CIF.A00004</td>
<td>ZOODZIAA</td>
<td>E104CIFA</td>
<td>5,942,502</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>63,679,458</td>
<td>173,605</td>
<td>4089</td>
<td>0.661110</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.N127XFD.R.A</td>
<td>ZOODZICA</td>
<td>N127XFD</td>
<td>24,330,441</td>
<td>17,670</td>
<td>19,696</td>
<td>0</td>
<td>61,776,087</td>
<td>4,364,814</td>
<td>40</td>
<td>0.162602</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.E103CIF.A00003</td>
<td>ZOODZIAA</td>
<td>E103CIFA</td>
<td>4,433,407</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60,130,646</td>
<td>124,992</td>
<td>4089</td>
<td>0.475992</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.N127XFD.R.A</td>
<td>ZOODZIGA</td>
<td>N127XFD</td>
<td>12,132,415</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54,528,900</td>
<td>156,271</td>
<td>4089</td>
<td>0.595108</td>
</tr>
<tr>
<td>DKL E6.OS.PAY.E.IN.DX.CICS.DAT</td>
<td>CE6070RP</td>
<td>HBDD004</td>
<td>79,206</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>46,876,875</td>
<td>147,422</td>
<td>100</td>
<td>0.013733</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.PG0W0IF.X00032</td>
<td>ZOODZIFA</td>
<td>PG0W0IFX</td>
<td>11,813,441</td>
<td>86,676</td>
<td>0</td>
<td>0</td>
<td>42,542,171</td>
<td>2,882,518</td>
<td>34</td>
<td>0.091275</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.N113PAC.A00039</td>
<td>ZOODZIAA</td>
<td>N113PACA</td>
<td>4,467,981</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35,373,809</td>
<td>87,734</td>
<td>10233</td>
<td>0.836125</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.PD01RCR.A00001</td>
<td>ZOODZIAA</td>
<td>PD01RCRA</td>
<td>9,576,590</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35,305,591</td>
<td>19,983</td>
<td>4089</td>
<td>0.041825</td>
</tr>
<tr>
<td>DKL I9.P1.BHI.E10FRP.M.A00015</td>
<td>ZOODZIBA</td>
<td>E10FRPMA</td>
<td>496,074</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>34,503,793</td>
<td>130,461</td>
<td>8185</td>
<td>0.994488</td>
</tr>
<tr>
<td>TABLE NAME</td>
<td>GETPAGES COUNT</td>
<td>GETPAGES TIME</td>
<td>SYNC I/O COUNT</td>
<td>SYNC I/O TIME</td>
<td>READ EXECUTIONS</td>
<td>NON-READ EXECUTIONS</td>
<td># OF ROWS</td>
<td>ROW LENGTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITBPREF</td>
<td>139,102,000</td>
<td>12:02.4</td>
<td>6590</td>
<td>00:03.2</td>
<td>93,327,028</td>
<td>0</td>
<td>1755</td>
<td>210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_DELIVERY_CHANNEL</td>
<td>72,922,988</td>
<td>07:11.9</td>
<td>31</td>
<td>00:00.0</td>
<td>41,195,936</td>
<td>0</td>
<td>105</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCESS_LOCATION</td>
<td>39,691,687</td>
<td>03:30.5</td>
<td>10</td>
<td>00:00.0</td>
<td>26,434,286</td>
<td>0</td>
<td>396</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGIONS</td>
<td>39,564,353</td>
<td>03:30.6</td>
<td>3</td>
<td>00:00.0</td>
<td>26,376,234</td>
<td>0</td>
<td>108</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUSTOM_PHOTO_RULES1</td>
<td>31,777,736</td>
<td>02:22.8</td>
<td>214</td>
<td>00:00.2</td>
<td>15,888,868</td>
<td>0</td>
<td>19</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL_TIMES_TB</td>
<td>21,738,834</td>
<td>02:17.8</td>
<td>1928</td>
<td>00:01.0</td>
<td>12,037,498</td>
<td>0</td>
<td>366</td>
<td>167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITBSPRI</td>
<td>9,633,519</td>
<td>00:48.8</td>
<td>32</td>
<td>00:00.1</td>
<td>6,422,346</td>
<td>0</td>
<td>8</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLCTN_PREFN_DFNTN</td>
<td>7,361,714</td>
<td>00:43.2</td>
<td>65</td>
<td>00:00.1</td>
<td>4,907,787</td>
<td>0</td>
<td>6</td>
<td>145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITBDKLC</td>
<td>15,389,568</td>
<td>01:10.0</td>
<td>2602</td>
<td>00:01.2</td>
<td>3,080,381</td>
<td>0</td>
<td>53</td>
<td>250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD_CORP</td>
<td>18,696,038</td>
<td>00:54.8</td>
<td>14006</td>
<td>00:06.4</td>
<td>1,696,716</td>
<td>0</td>
<td>487</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST_TB</td>
<td>5,205,402</td>
<td>00:13.1</td>
<td>634</td>
<td>00:00.3</td>
<td>1,671,653</td>
<td>0</td>
<td>727</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCT_RELCODE_PROCESS</td>
<td>2,833,120</td>
<td>00:10.0</td>
<td>14079</td>
<td>00:06.2</td>
<td>1,416,667</td>
<td>0</td>
<td>2</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBTBTRN</td>
<td>1,301,645</td>
<td>00:02.3</td>
<td>176</td>
<td>00:00.1</td>
<td>635,822</td>
<td>0</td>
<td>34</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARD_STATUS_TB</td>
<td>10,275,312</td>
<td>00:36.6</td>
<td>612</td>
<td>00:00.3</td>
<td>619,848</td>
<td>0</td>
<td>11</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITBCHGC</td>
<td>2,949,616</td>
<td>00:05.5</td>
<td>13</td>
<td>00:00.0</td>
<td>589,925</td>
<td>0</td>
<td>262</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>DBD Type</td>
<td>DL/I Total</td>
<td>DL/I Reads</td>
<td>DL/I Updates</td>
<td>GU Count</td>
<td>GN Count</td>
<td>ISRT Count</td>
<td>REPL Count</td>
<td>DLET Count</td>
<td>Other Count</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DATA1</td>
<td>202,585,821</td>
<td>202,472,358</td>
<td>0</td>
<td>202,472,358</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DATA2</td>
<td>38,898,112</td>
<td>38,897,313</td>
<td>799</td>
<td>25,266,537</td>
<td>13,630,776</td>
<td>0</td>
<td>799</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL1</td>
<td>38,742,393</td>
<td>38,658,675</td>
<td>83,718</td>
<td>19,313,741</td>
<td>19,344,934</td>
<td>76,692</td>
<td>0</td>
<td>7,026</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL2</td>
<td>28,702,133</td>
<td>28,701,975</td>
<td>158</td>
<td>15,341,058</td>
<td>13,360,917</td>
<td>46</td>
<td>108</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL3</td>
<td>17,845,734</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL4</td>
<td>7,072,987</td>
<td>132</td>
<td>22</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL5</td>
<td>6,953,822</td>
<td>132</td>
<td>22</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL6</td>
<td>6,846,180</td>
<td>132</td>
<td>22</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DATA5</td>
<td>6,708,378</td>
<td>6,613,242</td>
<td>95,136</td>
<td>6,613,242</td>
<td>0</td>
<td>1,157</td>
<td>93,979</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL8</td>
<td>6,546,443</td>
<td>132</td>
<td>22</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL9</td>
<td>6,519,354</td>
<td>1,821,462</td>
<td>4,697,892</td>
<td>1,821,194</td>
<td>268</td>
<td>4,697,852</td>
<td>0</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL10</td>
<td>6,478,828</td>
<td>132</td>
<td>22</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10/23/2012</td>
<td>DKL11</td>
<td>6,219,972</td>
<td>6,153,359</td>
<td>66,613</td>
<td>3,065,342</td>
<td>3,088,017</td>
<td>46,356</td>
<td>19,754</td>
<td>503</td>
<td>0</td>
</tr>
</tbody>
</table>
Top 5 features of IN-MEMORY tables

- Shortest path
- Optimize search
- Implicit commands
- Dynamic indexes
- Flexible API
Results that IN-MEMORY tables can provide

<table>
<thead>
<tr>
<th>With in-memory tables you can:</th>
<th>Decrease MSU</th>
<th>Decrease elapsed time</th>
<th>Increase flexibility and market adaptation</th>
<th>Reduce maintenance</th>
<th>Enable new paradigms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place reference data in tables</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replace temporary files with temporary tables</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Use tables for rules</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Use tables as a message queue</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Use decision tables to replace logic trees</td>
<td>Often</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Use tables for process control</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Use temporary tables for implementing complex algorithms</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>