Temp-Table Monitoring & Tuning

Dan Foreman, White Star Software

Abstract: TEMP-TABLEs are typically one of the crucial components in a Progress application. But for many years it was difficult, if not impossible, to capture useful information about TEMP-TABLE activity and behavior.

This presentation discusses the options that are available for monitoring and tuning TEMP-TABLEs in all versions of Progress. Especially exciting are the new parameters and keywords that were added in OE 11.
Temp-Table Monitoring & Tuning

Dan Foreman, White Star Software (sort of)
danf@prodb.com
Dan Foreman – Short C.V.

• Progress user since 1984

• Author
  – *Progress Performance Tuning Guide*
  – *Progress System Tables*
  – *Progress Database Administration Guide*
  – And lots of other minor publications
  – Electronic & (traditional) paper versions available
Non-Progress Interests

• Cycling
Non-Progress Interests

- Basketball
Etiquette

• Mobile phones on mute please!
Before TT there were WORK-FILEs

- 100% Memory resident
- No indexes
- Record access is sequential
- FIND FIRST > FIND LAST > FIND FIRST can be very inefficient
- Increasing the size of a record can be very expensive
- Can potentially crash a system with three 4GL statements.......

White Star Software
Before TT there were WORK-FILEs

DEFINE WORK-FILE junk LIKE customer.
REPEAT :
  CREATE junk.
/* END is not required in a single block program */
Use Case for WORK-FILEs

- Limited number of records created; I’d recommend no more than a few thousand
- The number of records won’t increase or will have a minimal increase
- Indexing is not important
- In these cases WORK-FILEs can actually be superior (i.e. much faster) compared to TEMP-TABLEs
The Future for WORKFILEs

- There is a good chance that Progress will stop supporting and totally remove this language construct.
Temp-Table Basics

• Similar in many ways to Database Tables
• CRUD (Create, Read, Update, Delete)
• Changes are not logged to the Before Image or After Image files
• Subject to Local Before Image (LBI) activity unless NO-UNDO is used
• Type 2 Storage
• No record locking
• No latching (in DB shared memory); this is important when comparing the performance of Temp-Tables and Database tables
TEMPTABLE Basics

• TT data is written to a Client temporary file with a DBI prefix
• The DBI file is similar to a Database but has no AI, BI, LG, LK files
• The DBI file is private to a Client, i.e. not shared
• Location of Client Temp Files can be set with –T parameter
• Client Temp Files are hidden on *ix; make them visible with -t Client Startup Parameter
Fast Method to Empty a TEMP-TABLE

- EMPT Y TEMP-TABLE *tablename* [ NO-ERROR ]
- Many **times** faster than deleting records in a loop
- One limitation: “When you empty a temp-table that is defined as UNDO *within a transaction*, the AVM deletes the records individually. This is less efficient than emptying the temp-table outside the transaction, where the AVM deletes all records in the temp-table as a unit.”
NO-UNDO

• NO-UNDO on the TEMP-TABLE Definition
• 21% performance improvement in a simple benchmark when adding the NO-UNDO option
TT Related Client Startup Options

- `-tmpbsize` – the block size used for the TT “database”
  - The default value has changed more than once so we recommend setting it explicitly
  - A value of 8(k) is the historic recommendation but new benchmarks will appear shortly

- `-Bt` – memory buffers for Temp-Tables (Similar to DB `-B` parameter)
  - Client Startup Parameter
  - Memory Cost = (-Bt * -tmpbsize) * (# of Clients)
TT Related Client Startup Options

- 
  -T – Location of DBI file (and other Client Temp Files)
    – Make sure these files are pointed to the fastest device available
Benchmark of -tmpbsize

- Platform: Linux
- Version: V11.6 SP2
- Define TT with 2 fields, INT & CHAR
- Create 400k TT records
- Update 400k TT records (add 15 bytes to a string)
- Delete 400k TT records (with a FOR EACH)
Benchmark of -tmpbsize

• Used –Bt 32768 regardless of –tmpbsize setting

<table>
<thead>
<tr>
<th>Action</th>
<th>-tmpbsize 1</th>
<th>-tmpbsize 8</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>2152ms</td>
<td>3202ms</td>
<td>33%</td>
</tr>
<tr>
<td>UPDATE</td>
<td>1611</td>
<td>2261</td>
<td>29%</td>
</tr>
<tr>
<td>DELETE</td>
<td>1897</td>
<td>2576</td>
<td>26%</td>
</tr>
</tbody>
</table>

• Repeated the test with the default –Bt with similar results
TEMP-TABLEs versus Database Tables

• No Word Indexes for TT
• TT’s are typically faster because there is no AI or BI or Record Lock or Latch activity
• TT’s can be created (and deleted) dynamically
Pre-V11 TT Monitoring Options

- None – at least nothing that’s built into Progress
- Monitoring the size of the DBI file is about the only useful metric available
What’s New in V11

- New Client Startup Parameters
- Delayed Instantiation of TT’s on the Client
- Temp-Table Logging & Monitoring
  - LOG-MANAGER support for Temp-Tables
  - Virtual System Tables for TT’s
  - New Classes to support the logging & monitoring
Client Startup Parameters

- `-ttbaseindex`
- `-ttindexrangesize`
  - Default is 0; no statistics are recorded if the parameter is not used
  - Default for DB “equivalent” – `indexrangesize` is 50
  - With `indexrangesize` it is not a good idea to set it too high because it increases DB Shared Memory consumption but that problem doesn’t seem to exist with `-ttindexrangesize`
  - Remember that the memory consumption is PER CLIENT
Client Startup Parameters

• -ttbasetable
• -tttablerangesize
  – Default is 0; no statistics are recorded if the parameter is not used
  – Default for DB “equivalent” –tablerangesize is 50
  – With –tablerangesize it is not a good idea to set it too high because it increases DB Shared Memory consumption but that problem doesn’t seem to exist with -tttablerangesize
  – Remember that the memory consumption is PER CLIENT
Memory Consumption of –tt* Parameters

- Tested on Windows
- V11.5
- Character Client

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Only -tttablerangesize (process size)</th>
<th>Add –ttindexrangesize</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6296k (baseline)</td>
<td>6296k (baseline)</td>
</tr>
<tr>
<td>1</td>
<td>6300k</td>
<td>6292k</td>
</tr>
<tr>
<td>100</td>
<td>6304k</td>
<td>6320k</td>
</tr>
<tr>
<td>1000</td>
<td>6372k</td>
<td>6472k</td>
</tr>
<tr>
<td>10000</td>
<td>7084k</td>
<td>8024k</td>
</tr>
</tbody>
</table>
Delayed TT Instantiation

• The app doesn't incur the overhead of instantiating a TT until it's actually referenced
• The pre-V11 model instantiates all TT's when a new program is run, regardless of whether there is CRUD activity or not
• Use -nottdelay to revert to the old functionality
LOG-MANAGER for TT

- logentrytypes directly related to TT
  - TEMP-TABLE
  - TTStats

- Example on the next slide
[17/11/15@19:03:26] P-01267 T-3535 1 4GL -- Logging level set to = 2
[17/11/15@19:03:26] P-01267 T-3535 1 4GL -- Log entry types activated: temp-tables:4>,ttstats:4
[17/11/15@19:03:45] P-01269 T-1100 1 4GL -- Logging level set to = 2
[17/11/15@19:03:45] P-01269 T-1100 1 4GL -- Log entry types activated: temp-tables:4>,ttstats:4

[17/11/15@19:03:45] P-01269 T-1100 2 4GL TEMP-TABLE Created TEMP-TABLE test (ID:1 Indexes:1) tt1.p @ 12
[17/11/15@19:03:54] P-01269 T-1100 4 4GL TEMP-TABLE No Statistics for TEMP-TABLE test (ID: 1): 0 records
[17/11/15@19:03:54] P-01269 T-1100 2 4GL TEMP-TABLE Deleted TEMP-TABLE test (ID:1) tt1.p @ -1
[17/11/15@19:03:54] P-01269 T-1100 3 4GL TTSTATS Index: test.default (tt1.p)
[17/11/15@19:03:54] P-01269 T-1100 3 4GL TTSTATS Create: 400901 Delete: 400000 Read: 800003 OS-Read: 1879 Split: 904 Block-Delete: 905
[17/11/15@19:03:54] P-01269 T-1100 2 4GL TTSTATS Table: test (tt1.p)
[17/11/15@19:03:54] P-01269 T-1100 2 4GL TTSTATS Create: 400000 Update: 800000 Delete: 400000 Read: 800000 OS-Read: 9777
Temp Table Logging

• Original Feature Announcement
• “The OpenEdge Logging Infrastructure has been enhanced so that application developers can trace the creation and deletion of temp-tables in their applications. This logging capability strengthens the ability to troubleshoot applications that utilize temp-tables and ProDataSets as their primary data structures.”
Virtual System Tables (VSTs) for Temp Tables

• Original Feature Announcement
• “This ABL enhancement allows clients to gather information about the temp-tables used by the application via Virtual System Tables. These tables give the application access to database activity and status information, enabling an application to understand, debug, and tune the use of temp-tables within their application at runtime.”
Progress.Database.TempTableInfo Class

• “Provides information about a temp-table and its index and provides static properties and methods for retrieving and archiving temp-table information for an ABL session”
TempTableInfo Properties

- ArchiveIndexStatistics
- ArchiveTableStatistics
- TempTableCount
- TempTablePeak
ArchiveTableStatistics Property

• From the documentation: “When ArchiveTableStatistics property is set to TRUE, it creates a dynamic temp-table _TableStatHistory that stores the table statistics for every temp-table in the session range. The temp-table to which the data is archived cannot be deleted. It contains one record for every deleted application temp-table. This is a read-write property.”
TempTableInfo Methods

- GetIndexInfoByID()
- GetIndexStatHistoryHandle( )
- GetTableInfoByID( )
- GetTableStatHistoryHandle( )
- GetTableInfoByPosition( )
- GetVSTHandle( )
Progress.Database.VSTTableId Class

• “Provides static properties that identity the VST for returning specific temp-table information for an ABL session”
VSTTableId Properties

- ActBufferId
- ActIOTypeld
- ActSpaceId
- BuffStatusId
- IndexStatId
- TableStatId
- UserIndexStatId
- ActIndexId
- ActOtherId
- ActSummaryId
- DbStatusId
- MstrBlkId
- TransId
- UserTableStatId
- ActIOFileId
- ActRecordId
- BlockId
- FileListId
- StatBaseId
- UserIOLd
Starter Code

Progress.Database.TempTableInfo:ArchiveTableStatistics = YES.
Progress.Database.TempTableInfo:ArchiveIndexStatistics = YES.
Instant Error

Cannot set Progress.Database.TempTableInfo:ArchiveTableStatistics (15247)

So you investigate Message# 15247:
“Update to the property or field is not allowed. This can happen if the given object only provides read-only access to the property based on its state.”

What ????
Try Again

• Using this sample .pf file

    -ttbaseindex       1
    -ttindexrangesize  100
    -ttbasetable       1
    -tttablerangesize  100
Free Example in ProTop Code

• ttinfo.p has the cool stuff in it
• Test with tttest.p
• Both files are located in $PROTOP/lib
Questions?

• Conference Evaluations are appreciated
• Please thank the conference organizers!!
• Thank You!
• Contact:
  – Dan Foreman
  – danf@prodb.com
  – +1 541 908 3437 (but not right now please)