Consultingwerk
software architecture and development

What's happening with my temp-tables?

Peter Judge
peter.judge@consultingwerk.com

Peter Judge

Writing 4GL since 1996, working on a variety of frameworks and applications.

Active member of the OpenEdge community and speaker at international conferences

Focusing on integration with OpenEdge applications, PASOE, software architecture and web technologies

Modernization in Focus

Modernization of Legacy OpenEdge Applications

Deep Technical Expertise

Global IT Partner with Local Presence

More than Consulting – We Deliver Tools & Solutions

Agenda

- Introduction
- Startup parameters
- Temp-table VSTs
- Detecting deep copies
- Rowids & holders

Temp-Tables

- Temp-Tables are the most widely used complex data-structure in the ABL
  - Introduced in v7.0 in 1991 along with internal procedures and Report Builder
  - Replaced worktables (there since v4 / 1987)
- Different use cases
  - "Sneaky" caching with GLOBAL SHARED
  - Aggregation of data for reports
  - Caching of aggregated data
  - Foundation for XML and JSON export and import
  - Serialization of application data between AppServer and Client
  - Logical schemas for Business Entities / OERA

ProDatasets

- ProDatasets provide functional extensions to temp-tables as a wrapper or container for one or more temp-tables
  - Data-Relations
  - Declarative read and update routines based on Data-Source object handle
  - Paged data-reading
  - Tracking of changes (create, update, delete)
  - Mapping of field-names between Database and Application logic/interfaces (pt_mstr.pt_fr_class)
  - More complex XML and JSON import/export

Defining a temp-table

- Static: DEFINE TEMP-TABLE statement
- Dynamic: CREATE TEMP-TABLE statement
  - Care needed to avoid memory leaks
  - There are cases where it is very hard to avoid leaks
- Static temp-tables always scoped to a compile unit (procedure, class instance or class static)
  - If more than one program needs / passes a temp-table, you'll typically see them defined in include files to avoid runtime errors
  - Each instance – handle – has a copy in memory
- OOABL introduced STATIC temp-tables

Agenda

- Introduction
- Startup parameters
- Temp-table VSTs
- Detecting deep copies
- Rowids & holders

Management of Temp-Tables

- Temp-Tables managed in similar way as tables in a Database
  - Records organized in blocks
  - Buffer pool; -Bt specifies number of blocks in temp-table buffer pool
  - Block size: -tmpbsize parameter, 1k, 4k or 8k
- Temp-Tables are kept in buffer pool until buffer pool is fully occupied with data, then dumped into DBI file

Temp-Table related AVM startup parameters

Parameter Name	Description
-Bt	Number of Buffers for Temporary Tables
-tmpbsize	Temporary Table Database Block Size
-T	Temporary Files Directory
-t	Save Temp Files

-Bt and -tmpbsize

- Temp-Tables are organized like DB tables in blocks in the DBI file (referred to as the temp-table database)
- DBI file is used when –Bt buffer is full, -Bt between 10 and 50000
- DBI file only grows, never shrinks
- -tmpbsize: Blocksize 1k, 4k (default) or 8k
- Memory used: ~ 1.1 * (value of –Bt) * (value of –tmpbsize)
- -t makes temp-files (like DBI) visible in file-system
  - Mainly for Unixes
  - Can't be used with TDE
- How much –Bt do I need? **As much as needed to avoid DBI file growth**

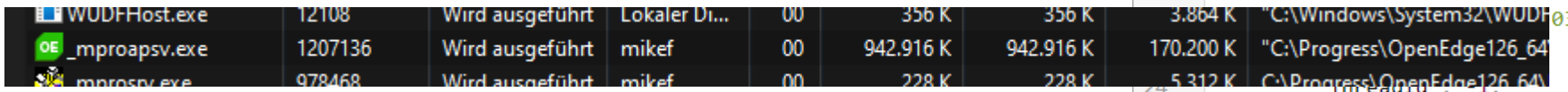
-Bt and –tmpbsize on PASOE

- -Bt and –tmpbsize are allocated per session on PASOE multi-session agent

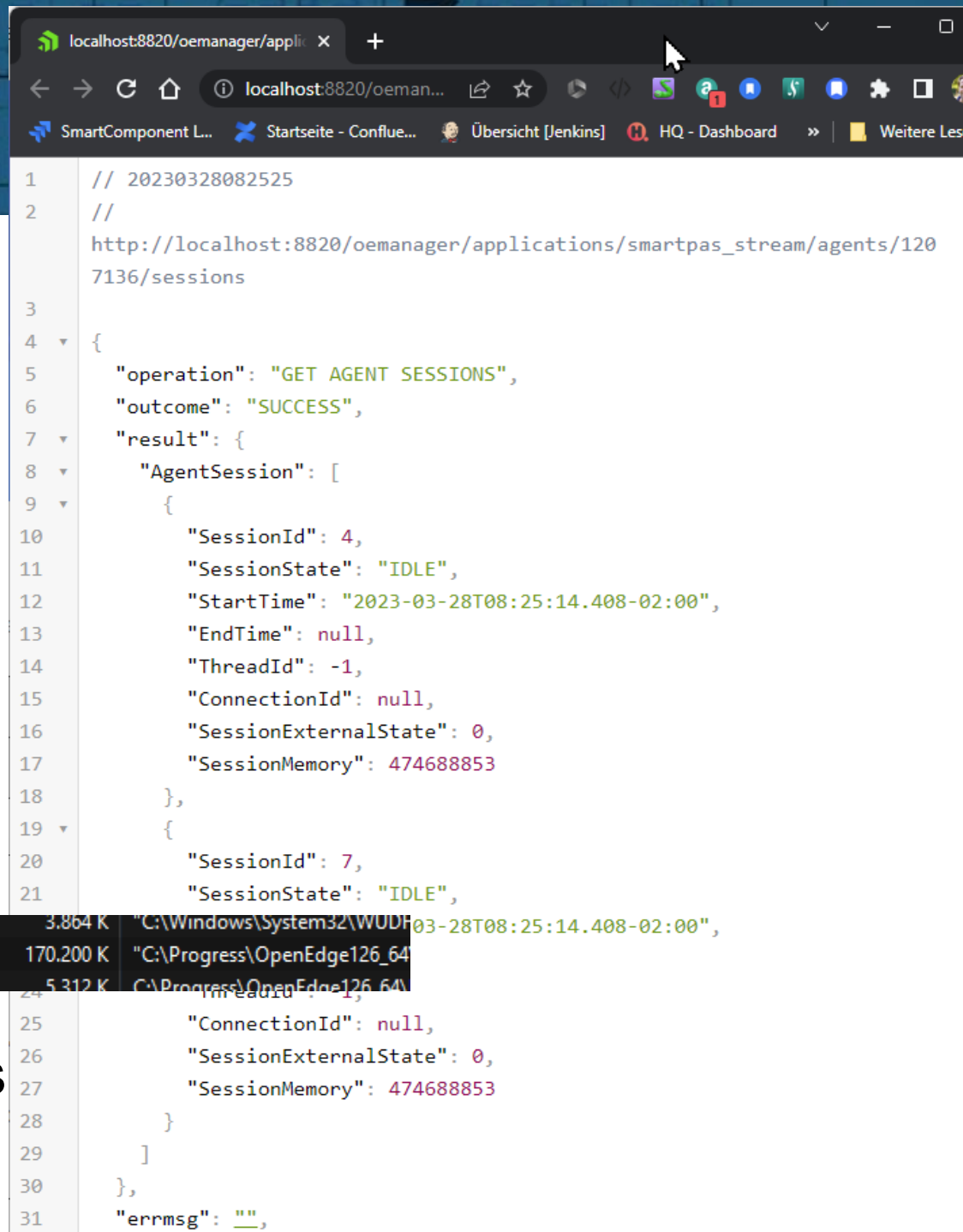
Value	-Bt 10 -tmpbsize 8	-Bt 50000 -tmpbsize 8
# of sessions	2	2
Agent Memory	110,120 K	942,916 K
Session Memory after startup	45,7 MB * 2	452,7 MB * 2

-Bt and –tmpbsize on PASOE

- -Bt 50000 (maximum)
- -tmpbsize 8
- http://localhost:8820/oemanager/applications/smartpas_stream/agents/1207136/sessions
- ~ 452,7 MB per session!

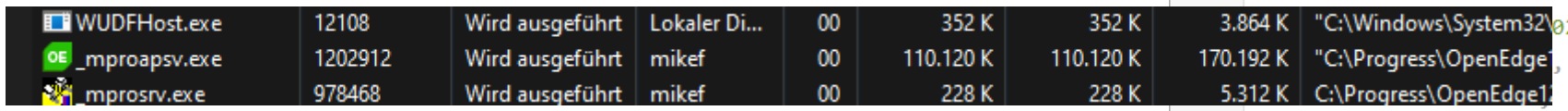


- 50000 * 8192 * 1,1 = 450,560,000 bytes per session



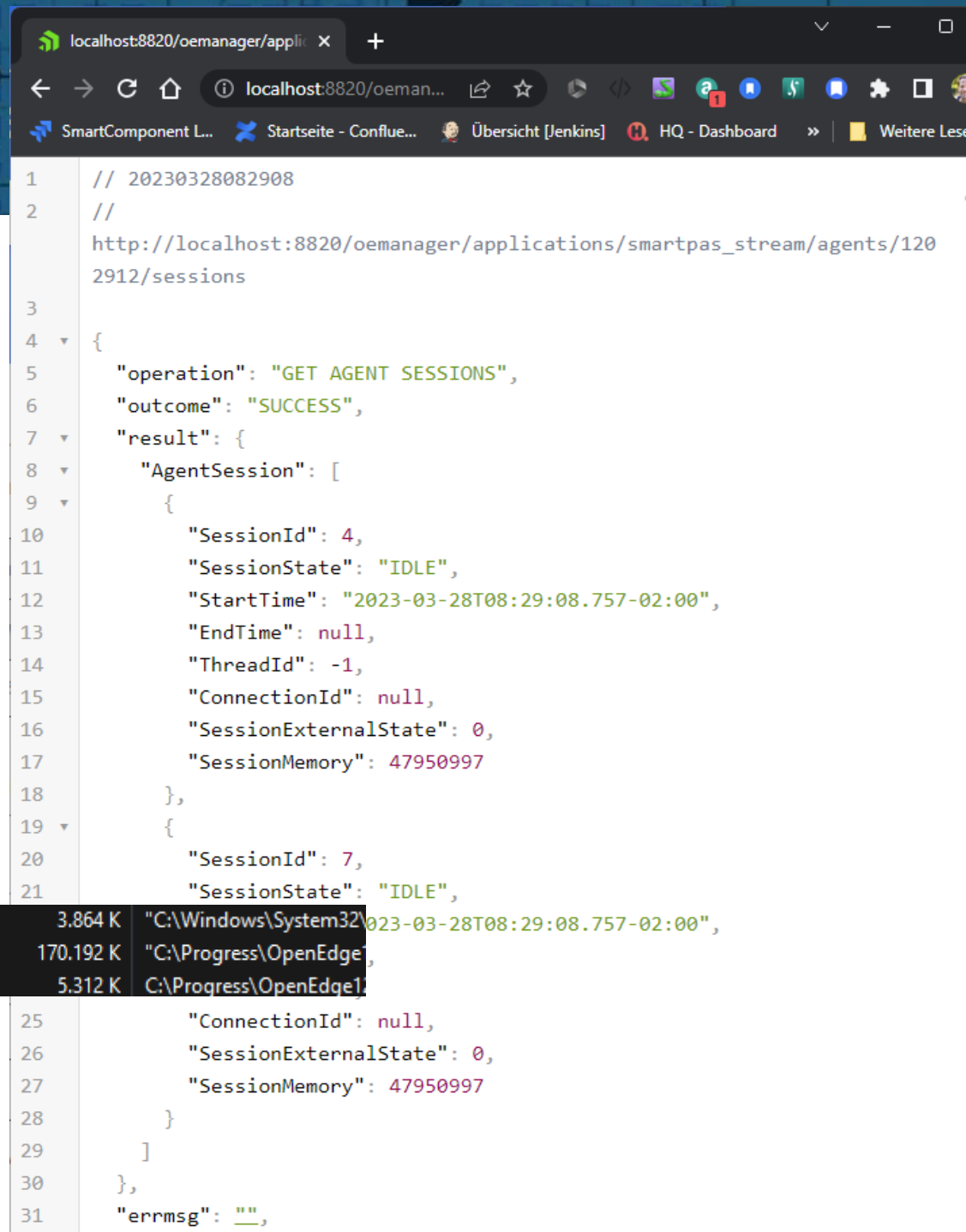
-Bt and –tmpbsize on PASOE

- -Bt 10 (minimum)
- -tmpbsize 8
- http://localhost:8820/oemanager/applications/smartpas_stream/agents/1202912/sessions
- ~ 45,7 MB per session



WUDFHost.exe	12108	Wird ausgeführt	Lokaler Di...	00	352 K	352 K	3.864 K	"C:\Windows\System32\
_mproapsv.exe	1202912	Wird ausgeführt	mikef	00	110.120 K	110.120 K	170.192 K	"C:\Progress\OpenEdge
_mprosrv.exe	978468	Wird ausgeführt	mikef	00	228 K	228 K	5.312 K	C:\Progress\OpenEdge1

- 10 * 8192 * 1,1 = 90112 bytes per session



Temp-Table related AVM startup parameters

Parameter Name	Description
-zdsreuse	Allow objects with temp-tables and ProDatasets in –reusableObjects cache (OpenEdge 12.1)
-nochkttnames	No check temp-table names
-ttmarshal	Temp-table Schema Marshal

Statistics

# idx	DB records	-Bt	-tmpbsize	DBI file size	Populating	FOR EACH by Name	Total
1	201122	10	1	41,4 MB	3,3 sec	2,1 sec	5,4 sec
1	201122	50000	1	empty	1,8 sec	0,6 sec	2,4 sec
1	201122	10	8	34,3 MB	2,2 sec	2,1 sec	4,3 sec
1	201122	50000	8	empty	2 sec	0,5 sec	2,5 sec

- Size of –Bt has largest impact on runtime
- When writing to DBI file, larger blocksize increases performance
- Unindexed sort-access primarily influenced by –Bt, not -tmpbsize

Agenda

- Introduction
- Startup parameters
- Temp-table VSTs
- Detecting deep copies
- Rowids & holders

Temp-table VSTs

- Measure temp-table activity in a **session**
  - Statistics gathered for temp-tables covered by –tt* startup parameters
  - VSTs are similar to those available for DB tables: table and index stats, user table and index stats and more. The schemas of the VSTs may differ.
- Available since OE v11.0
- Accessible in ABL via the Progress.Database.TempTableInfo class
- Logging of stats enabled with the TTSTATS log entry type
  - Only certain data is written (table and index stats)

Enabling Temp-Table statistics

Parameter Name	Description	Notes
-ttbaseindex	Temp-table Base Index	0 = off ; use 1
-ttbasetable	Temp-table Base Table	0 = off ; use 1
-ttindexrangesize	Temp-table Index Range Size	Max = 65536 ; use 32767
-tttablerangesize	Temp-table Table Range Size	

There's a practical limit of 32767 temp-tables in a session

https://community.progress.com/s/article/getting-error-9287-number-of-active-temp-tables-reached-limit-of-32767

Enabling Temp-Table statistics

- Turn on via -ttbasetable and -ttbaseindex parameters with a value > 0
- To enable keeping historical statistics, the ArchiveTableStatistics and ArchiveIndexStatistics properties must be set to TRUE
  - Properties of Progress.Database.TempTableInfo
  - Can be toggled on and off in a session
  - Typically turn on and leave on, but when off, certain VSTs may be emptied – allowing the statistics to be reset
- Both sets of statistics are necessary to get an accurate view of temp-table usage

Available Temp-Table VSTs

ActBufferId property	ActIndexId property	ActIOFileId property
ActIOTypeId property	ActOtherId property	ActRecordId property
ActSpaceId property	ActSummaryId property	BlockId property
BuffStatusId property	DbStatusId property	FileListId property
IndexStatId property	MstrBlkId property	StatBaseId property
TableStatId property	TransId property	UserIOId property
UserIndexStatId property	UserTableStatId property	

- Nearly all listed as properties of the Progress.Database.VSTTableId class
- Stats per temp-table
  - TableStat and IndexStat (currently in-memory)
  - Also TableHistoryStat and IndexHistoryStat (no longer in-memory)
- Global / per session statistics
  - FileList: Info about the DBI file like size, block size, location
  - ActSummary: summary reads, writes, transactions, etc
  - DbStatus: temp-table DB info

Getting Table and Index stats

- The TableStat VST contains an entry for all potential temp-tables ie -tttablerangesize less –ttbasetable
- Determine how many temp-tables are in memory
- With the table handle you can read its contents

```
hVst = Progress.Database.TempTableInfo:GetVSTHandle(Progress.Database.VSTTableId:TableStatId).
hBuffer = hVst:default-buffer-handle.

hQuery:set-buffers (hBuffer).
hQuery:query-prepare (substitute ("preselect each &1":U, hBuffer:NAME)).
hQuery:query-open().

oTable = new JsonObject().
oRows = new JsonArray().
oTable:Add(hTable:serialize-name, oRows).

do while hQuery:get-next():
  if Progress.Database.TempTableInfo:GetTableInfoById (hBuffer::_TableStat-Id,
                                                       output hTable,
                                                       output cProgName) then do:
    oRow = new JsonObject().
    hBuffer:serialize-row("JSON", "JsonObject", oRow, ?, ?, ?, yes).    /* omit outer object */

    oRow:Add("_Table-Name", hTable:name).
    oRow:Add("_Prog-Name", cProgName).

    oRows:Add(oRow).
  end.
end.
```

Getting Table and Index stats

- The TableStat VST contains an entry for all potential temp-tables ie -tttablerangesize less –ttbasetable
- Determine how many temp-tables are in memory
- With the table handle you can read its contents

```
hVst = Progress.Database.TempTabl                                    TableId:TableStatId).
hBuffer = hVst:default-buffer-han

hQuery:set-buffers (hBuffer).
hQuery:query-prepare (substitute
hQuery:query-open().

oTable = new JsonObject().
oRows = new JsonArray().
oTable:Add(hTable:serialize-name,

do while hQuery:get-next():
  if Progress.Database.TempTableI                                    t-Id,


    oRow = new JsonObject().                                         then do:
    hBuffer:serialize-row("JSON",

    oRow:Add("_Table-Name", hTabl                                    * omit outer object */
    oRow:Add("_Prog-Name", cProgN

    oRows:Add(oRow).
  end.
end.
```

```
{
  "_TableStat": [
    {
      "_TableStat-id": 2,
      "_TableStat-read": 4,
      "_TableStat-update": 3,
      "_TableStat-create": 3,
      "_TableStat-delete": 3,
      "_TableStat-OsRead": 0,
      "_TableStat-PartitionId": 0,
      "_Table-Name": "ttData",
      "_Prog-Name": "tt-data.p"
    },
    {
      "_TableStat-id": 3,
      "_TableStat-read": 0,
      "_TableStat-update": 0,
      "_TableStat-create": 6,
      "_TableStat-delete": 0,
      "_TableStat-OsRead": 0,
      "_TableStat-PartitionId": 0,
      "_Table-Name": "ttCustomer",
      "_Prog-Name": "tt-customer.p"
    }
  ]
}
```

Getting Table and Index History stats

- The *StatHistory VST only contains data for temp-tables that are no longer in memory
- Adds the table and program name, and when the table was removed from memory
- History VSTs can be emptied

```
hIndexHistory = Progress.Database.TempTableInfo:GetIndexStatHistoryHandle().
hTableHistory = Progress.Database.TempTableInfo:GetTableStatHistoryHandle().

{"_TableStatHist": [
  {
    "_Table-Name": "ttCustomer",
    "_Prog-Name": "tt-customer.p",
    "_Delete-Timestamp": "2025-08-23T16:38:19.565",
    "_TableStat-read": 0,
    "_TableStat-update": 0,
    "_TableStat-create": 6,
    "_TableStat-delete": 0,
    "_TableStat-OsRead": 0
  },
  {
    "_Table-Name": "ttData",
    "_Prog-Name": "tt-data.p",
    "_Delete-Timestamp": "2025-08-23T16:38:19.564",
    "_TableStat-read": 4,
    "_TableStat-update": 3,
    "_TableStat-create": 3,
    "_TableStat-delete": 3,
    "_TableStat-OsRead": 0
  }
]}
```

Making sense of the data

- At any point in time we can see
  - Which temp-tables are in-memory, and their statistics, and
  - Which tables are no longer in memory, and their statistics

- These statistics are cumulative over time
  - We usually want to report on usage at the end of an event: an AppServer request, a single batch job run, a report
  - We need to calculate the usage per event … in a temp-table, of course

Temp-table stats service

- A program (class) that typically runs for the length of the session
  - Contains a temp-table to store VST data: both total/cumulative counts, and delta
  - Calls the worker class's "delta" method when it needs to know/report the temp-table statistics
  - Reports to a log or other mechanism the statistics
- A worker program populates the temp-table from the current and historical VST data
  - Contains a method to calculate the deltas

Reporting temp-table stats on a PASOE request

- PAS gives us "startup", "before run" and "after run" event procedures: startup, activate and deactivate
- On startup, initialize the temp-table statistics service
- On activate, record the starting timestamp
  - Used to determine which temp-tables were deleted in a request
- On deactivate, calculate and report on temp-table activity
  - Call into the "delta" method
  - Report on the temp-table statistics for that request

Example output of TableStat and IndexStat VSTs

```
#################################################################    TEMP-TABLE STATISTICS    ##########################################################################
Table Name                                Procedure Name                                                                  Reads          Creates          Updates          Deletes Deleted
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------
ttSerializableProperties                  Consultingwerk.JsonSerializable                                                     1                0                0                0 ?
ttServices                                C.Framework.ServiceContainer                                                       38                0                0                0 ?
ttFactory                                 C.Framework.Factory                                                                 3                0                0                0 ?
ttServices                                C.OERA.ServiceManagerImpl                                                           3                0                3                0 ?
ttSerializeNames                          C.Framework.SerializeNamesProviderService                                           8                0                0                0 ?
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Index Name                                Procedure Name                                                                  Reads          Creates          Updates          Deletes Deleted
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
ttSerializableProperties.ClassName        Consultingwerk.JsonSerializable                                                     1                0                0                0 ?
ttServices.ServiceType                    C.Framework.ServiceContainer                                                       41                0                0                0 ?
ttServices.ServiceName                    C.OERA.ServiceManagerImpl                                                           3                0                0                0 ?
ttServices.ClassNameLastRequest           C.OERA.ServiceManagerImpl                                                           0                3                0                0 ?
ttServices.LastAccessed                   C.OERA.ServiceManagerImpl                                                           0                3                0                0 ?
eSmartBusinessEntity.BusinessEntityName   C.SF.System.SmartBusinessEntityConfigurationProvider                               1                0                0                0 ?
ttSerializeNames.ClassName                C.Framework.SerializeNamesProviderService                                          16                0                0                0 ?
ttFactory.Factory                         C.Framework.Factory                                                                 3                0                0                0 ?
```

Demo

- Temp-table VSTs and service

Agenda

- Introduction
- Startup parameters
- Temp-table VSTs
- Detecting deep copies
- Rowids & holders

Temp-Tables as Parameters

- A Temp-Table can be passed from one program to another as parameter
- Temp-Tables are by default passed “by-value” (deep copy of every record of the source Temp-Table)
- Table-handles and statically-defined temp-tables are interchangable
- Requires schema to be “similar”
- Type-check can be relaxed with –nochkttnames startup parameter

- Log-entry-type TEMP-TABLES shows creation of temp-tables in memory – even for statically-defined ones

Demo: Example of deep copies

BY-REFERENCE to the rescue

- The caller (!!!) can specify the BY-REFERENCE keyword when passing the Temp-Table parameter
- Instead of duplicating the Temp-Table the “handle” is passed
- The caller’s Temp-Table will “overload” the callee’s own temp-table
- DBI file without significant growth

```
RUN tt-param-callee.p (INPUT-OUTPUT TABLE ttCustomer BY-REFERENCE) .
```

```
DBI File Size: 41,4 MB
tt-param-start.p:  Rowid Customer No 1 0x0000000000000120
tt-param-callee.p: Rowid Customer No 1 0x0000000000000120
tt-param-start.p:  Rowid Customer No 1 0x0000000000000120
Runtime for calling program: 3 msecs
DBI File Size: 41,5 MB
```

Counting references

- ABL does not provide a direct way to tell if a callee is working on a caller's Temp-Table or its own
- Temp-Table's Handle changes during execution of procedure invoked with BY-REFERENCE Temp-Table parameter
- e.g. persistent procedure's main-block vs. internal procedure invoked from caller
- Temp-Table object handle provides NUM-REFERENCES attribute indicating how many additional routines hold a reference to the Temp-Table

Table‘s HANDLE and NUM-REFERENCES

```
DEFINE TEMP-TABLE ttCustomer NO-UNDO
    LIKE Customer
    INDEX CustNum IS PRIMARY UNIQUE CustNum
    .

/* ***************************  Main Block  *************************** */

MESSAGE "Main-Block:" THIS-PROCEDURE:FILE-NAME
        "ttCustomer:" TEMP-TABLE ttCustomer:HANDLE
        "num-references:" TEMP-TABLE ttCustomer:NUM-REFERENCES.

PROCEDURE internal-procedure:
    DEFINE INPUT-OUTPUT PARAMETER TABLE FOR ttCustomer .

    MESSAGE "internal-procedure:" THIS-PROCEDURE:FILE-NAME
            "ttCustomer:" TEMP-TABLE ttCustomer:HANDLE
            "num-references:" TEMP-TABLE ttCustomer:NUM-REFERENCES.

END PROCEDURE .
```

Table‘s HANDLE and NUM-REFERENCES

- First procedure‘s ttCustomer handle 1001
- Callee procedure‘s ttCustomer handle 1081 (in main-block)
- Within internal procedure callee with BY-REFERENCE temp-table ttCustomer’s handle is 1001
- All buffer’s defined in callee procedure (main-block or internal procedure) reference caller’s Temp-Table

```
tt-param-start.p:  Rowid Customer No 1 0x0000000000000120
Procedure tt-param-start-by-reference.p ttCustomer 1001 num-references: 0
Main-Block: tt-param-callee-by-reference.p ttCustomer: 1081 num-references: 0
internal-procedure: tt-param-callee-by-reference.p ttCustomer: 1001 num-references: 1
tt-param-start.p:  Rowid Customer No 1 0x0000000000000120
```

INPUT and OUTPUT and BY-REFERNCE

- Temp-Table passed BY-REFERENCE always passed from caller to callee
- Regardless of INPUT, OUTPUT or INPUT-OUTPUT option
- It's basically always INPUT (!!!) – as a reference (a strong-typed handle) is passed
- Exception are TABLE-HANDLE parameters when the caller provides an invalid Temp-Table Handle (unknown-value, ?)

Demo: By-reference

Agenda

- Introduction
- Startup parameters
- Temp-table VSTs
- Detecting deep copies
- Rowids & holders

Temp-Table ROWID‘s

```
FIND ttCustomer WHERE ttCustomer.CustNum = 1 .
MESSAGE "tt-param-start.p:  Rowid Customer No 1" ROWID (ttCustomer) .
```

```
tt-param-start.p:  Rowid Customer No 1 0x0000000000000120
tt-param-callee.p: Rowid Customer No 1 0x000000000014b920
tt-param-start.p:  Rowid Customer No 1 0x0000000000000220
```

Temp-Table ROWID‘s

- A ROWID determines the physical position of a record (block number and record number in the block)
- As the temp-table is duplicated when passing by-value, the “same” record receives a different ROWID
- After returning from a procedure called with INPUT-OUTPUT Temp-Table the ROWID’s of records have changed too
- In short – Temp-Table ROWID’s cannot be trusted to relocate a record in a Temp-Table – similar to DB ROWID’s after dump and load
- A pledge for primary unique key in every (Temp-)Table!

Wrapping a Temp-Table inside an object

- Default parameter passing for Temp-Tables is by-value
- Parameter passing of object-references is always by-reference
- By-reference is generally more efficient
- Wrapping a Temp-Table definition inside an object can make passing Temp-Table by-reference easier

Temp-Table Holder class definition

```
CLASS BindParameter.CustomerTempTableHolder:

    DEFINE TEMP-TABLE ttCustomer NO-UNDO
        LIKE Customer
        INDEX CustNum IS PRIMARY UNIQUE CustNum .

    METHOD PUBLIC VOID BindTable (OUTPUT TABLE FOR ttCustomer BIND):
    END METHOD.

END CLASS.
```

Questions

Subscribe Now to Consultingwerk's Official Newsletter!

www.consultingwerk.com/newsletter

or marketing@consultingwerk.com


Consultingwerk
software architecture and development