Consultingwerk
software architecture and development

# *Mysteries of Temp-Table and ProDataset Parameters*

**Mike Fechner**
**mike.fechner@consultingwerk.de**

# Mike Fechner

- Director, Lead Modernization Architect and Product Manager, Architect of the SmartComponent Library and WinKit
- Specialized on object-oriented design, software architecture, desktop user interfaces and web technologies
- 30 years of Progress experience (V5 … OE12)
- Active member of the OpenEdge community
- Frequent speaker at OpenEdge related conferences around the world


© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Consultingwerk Software Services Ltd.


- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiaries in UK, USA and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Services Portfolio, Progress Software

- OpenEdge (ABL, Developer Tools, Database, PASOE, …)
- Telerik DevCraft (.NET, Kendo UI, Angular, …), Telerik Reporting
- OpenEdge UltraControls (Infragistics .NET)
- Telerik Sitefinity CMS (incl. integration with OpenEdge applications)
- Kinvey Plattform, NativeScript
- Corticon BRMS
- WhatsUp Gold infrastructure-, network- and application monitoring
- Kemp Loadmaster
- …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Services Portfolio, related products

- Protop Database Monitoring
- Combit List & Label
- Web frameworks, e.g. Angular
- .NET
- Java
- ElasticSearch, Lucene
- Amazon AWS, Azure
- DevOps, Docker, Jenkins, ANT, Gradle, JIRA, …
- …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Agenda

- **Introduction**
- Temp-Tables as Parameter Parameters
- BIND with Temp-Tables
- Table-Handle Parameters
- Temp-Table vs. local Database

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Temp-Tables and ProDatasets

- Temp-Tables are the most widely used complex data-structure in the ABL
- Different use cases
  - 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
  - Business Entities / OERA
  - …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# ProDatasets

- ProDatasets provide functional extensions to 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
  - …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# ProDataset parameters

- The samples on the following slides are based on Temp-Tables for simplicity
- But the same rules apply to ProDatasets – as they are basically a collection of Temp-Tables with some extra features

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Management of Temp-Tabels

- Static: DEFINE TEMP-TABLE statement
- Dynamic: CREATE TEMP-TABLE statement
- Static temp-tables always scoped to a compile unit (Procedure, class instance or class static)
- 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Temp-Table related AVM startup parameters

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# -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

- How much –Bt do I need? **As much as needed to avoid DBI file growth**

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# -Bt and –tmpbsize on PASOE

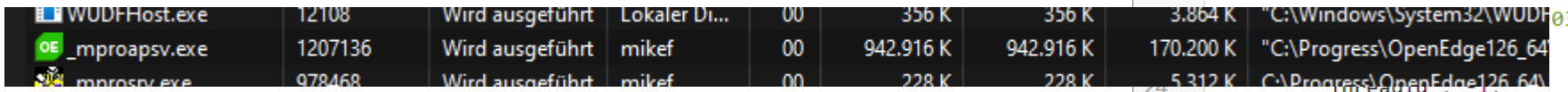
- -Bt and –tmpbsize are allocated per session on PASOE multi-session agent

| Value | -Bt 10<br>-tmpbsize 8 | -Bt 50000<br>-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 |

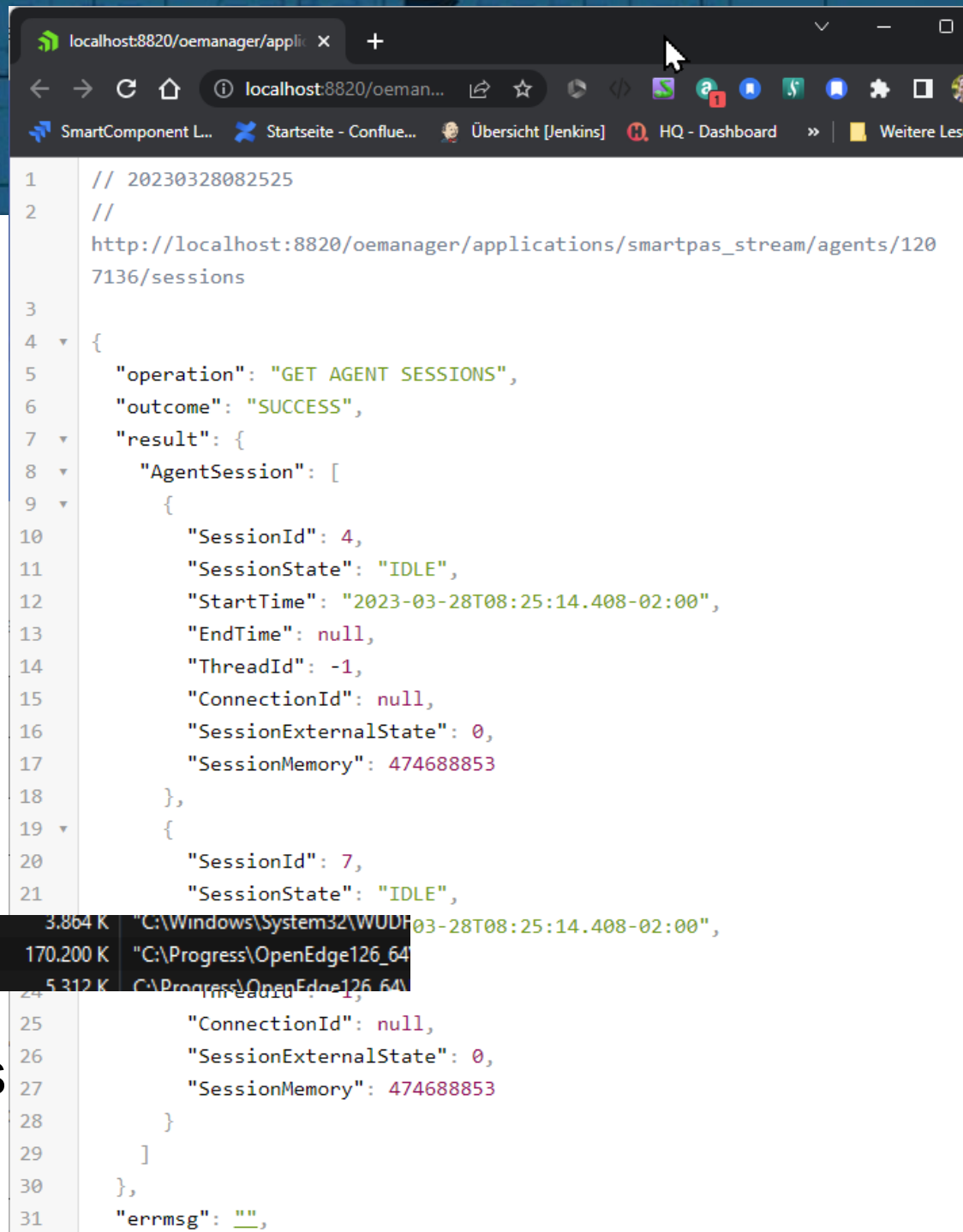
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# -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!



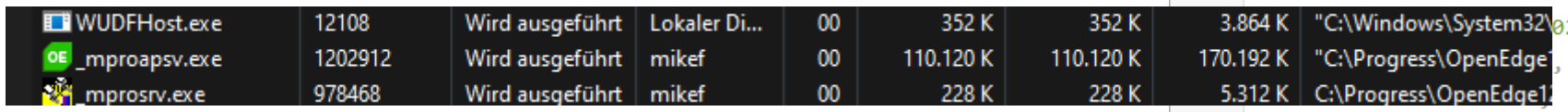

- 50000 * 8192 * 1,1 = 450,560,000 bytes per session




© 2023 Consultingwerk Software Services Ltd. All rights reserved.

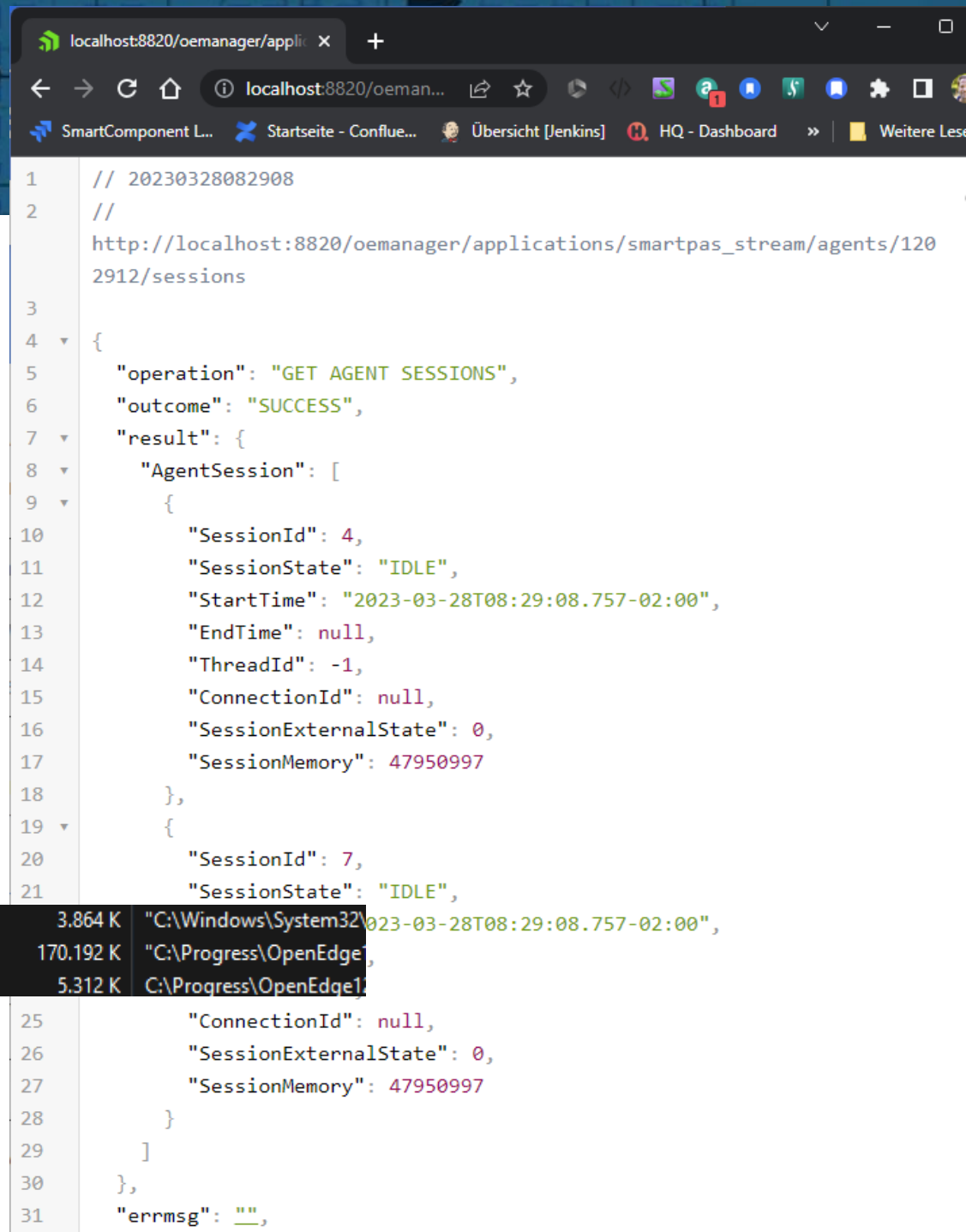
# -Bt and –tmpbsize on PASOE

- -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




© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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 |

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Temp-Table related AVM startup parameters

| Parameter Name | Description |
|---|---|
| -ttbaseindex | Temp-Table Base Index |
| -ttbasetable | Temp-Table Base Table |
| -ttindexrangesize | Temp-table Index Range Size |
| -tttablerangesize | Temp-Table Table Range Size |

- See https://community.progress.com/s/article/How-to-get-TEMP-TABLE-table-and-index-statistics for details on using the Temp-Table VST‘s
- Potentially a great subject for a future presentation …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# A „simple" program

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

ETIME (YES).

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
    ASSIGN i = i + 1 .
END.

MESSAGE "Runtime for creating records:" ETIME "msecs" .
MESSAGE "TT records created:" i .

ETIME (YES) .

FOR EACH ttCustomer BY ttCustomer.Name:
END.

MESSAGE "Runtime for iterating by ttCustomer.Name:" ETIME "msecs" .
```

Explicitly defines a single index on a single INTEGER field

©

# 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Alternative temp-table definition

```
DEFINE TEMP-TABLE ttCustomer NO-UNDO
    LIKE Customer
    //INDEX CustNum IS PRIMARY UNIQUE CustNum
    .
ETIME (YES).

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
    ASSIGN i = i + 1 .
END.

MESSAGE "Runtime for creating records:" ETIME "msecs" .
MESSAGE "TT records created:" i .

ETIME (YES) .

FOR EACH ttCustomer BY ttCustomer.Name:
END.

MESSAGE "Runtime for iterating by ttCustomer.Name:" ETIME "msecs" .
```

Inherits all indexes of DB table:

- CustNum (INTEGER)
- Name (CHARACTER)
- Country, PostalCode (CHARACTER, CHARACTER)
- Salesrep (CHARACTER)
- Comments (Word-Index)

©

# A guesstimate game ….

- With –Bt 10 and –tmpbsize 1 … how does the runtime change with 5 indexes in the temp-table
- Expectation is, that the read operation ordered by Name will be faster, as it's indexed
- Expectation is, that the population of the temp-table becomes slower with more indexes – but by which ratio?
  - 10%
  - 50%
  - 2 times
  - 10 times

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
12.6 64 c:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes>_progres -p all-indexes.p -b -db c:\work\SmartComponents4NET\126_64\db\Sports2000\sports2000 -Bt 10 -t -tm
Number of indexes: 5
Runtime for creating records: 25277 msecs
TT records created: 201122
Runtime for iterating by ttCustomer.Name: 1851 msecs
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes\DBI1321056a01232
DBI File Size: 46 MB

12.6 64 c:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes>_progres -p all-indexes.p -b -db c:\work\SmartComponents4NET\126_64\db\Sports2000\sports2000 -Bt 10 -t -tm
Number of indexes: 5
Runtime for creating records: 25042 msecs
TT records created: 201122
Runtime for iterating by ttCustomer.Name: 1753 msecs
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes\DBI1315152a34192
DBI File Size: 46 MB

12.6 64 c:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes>_progres -p all-indexes.p -b -db c:\work\SmartComponents4NET\126_64\db\Sports2000\sports2000 -Bt 10 -t -tm
Number of indexes: 5
Runtime for creating records: 24809 msecs
TT records created: 201122
Runtime for iterating by ttCustomer.Name: 1827 msecs
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TempTableIndexes\DBI1094276a24180
DBI File Size: 46 MB
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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 | 2,0 sec | 0,5 sec | 2,5 sec |
| **5** | **201122** | **10** | **1** | **46 MB** | **25,0 sec** | **1,8 sec** | **26,8 sec** |
| 5 | 201122 | 50000 | 1 | empty | 3,0 sec | 0,3 sec | 3,3 sec |
| 5 | 201122 | 10 | 8 | 38,3 MB | 18 sec | 2,0 sec | 20 sec |
| 5 | 201122 | 50000 | 8 | empty | 3,8 sec | 0,2 sec | 4 sec |

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Conclusion

- Total runtime with 1 index vs. 5 indexes always faster
- Difference with very small Temp-Table buffer significant (up to 8 times slower with 5 indexes)
- Difference with large Temp-Table buffer still noticeable (2 times slower with 5 indexes)
- Indexed read by Name only significantly faster when Temp-Table is fully in temp-table buffer
- As expected, indexes are slowing down creates and updates

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Conclusion

- Be careful which indexes to define on a Temp-Table – unlike a database table they may never be used for read (e.g. Word Index)
- As a rule of thumb – only define PUK index on temp-tables (ProDataset temp-tables should always be defined with a primary unique key)
- Only create additional indexes when you know that they are going to be used – frequently!
- Single unindexed read typically faster than maintaining the index
- Be careful with LIKE definition of Temp-Tables
- Cost of indexes on Temp-Tables increases when Temp-Tables are passed as parameters (by-value)

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Agenda

- Introduction
- **Temp-Tables as Parameters**
- BIND with Temp-Table Parameters
- Table-Handle Parameters
- Temp-Table vs. local Database

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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)
- Requires schema to be “similar”
- Type-check can be relaxed with –nochkttnames startup parameter

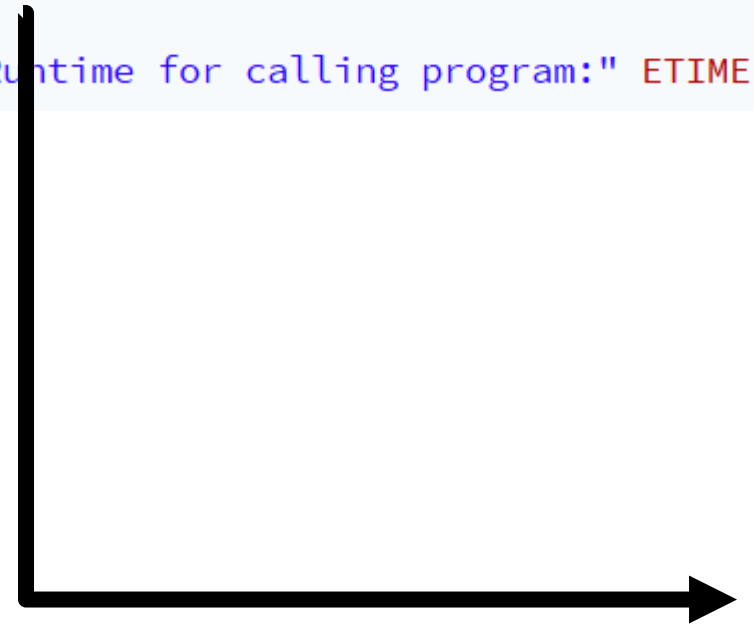
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Let‘s try it out – INPUT PARAMETER TABLE

```
ETIME (YES) .

RUN tt-param-callee.p (TABLE ttCustomer) .

MESSAGE "Runtime for calling program:" ETIME "msecs" .
```



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

DEFINE INPUT PARAMETER TABLE FOR ttCustomer .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Runtime statistics – INPUT PARAMETER TABLE

```
12.6 64 c:\Work\Presentations\TempTableAndDatasetParameter\TempTableParameter>_progres -p tt-param-s
Number of indexes: 1
Runtime for creating records: 3781 msecs
TT records created: 201122
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TempTableParameter\DBI1343412a45916
DBI File Size: 41,4 MB
Runtime for calling program: 3208 msecs
DBI File Size: 82,9 MB
```

- Running procedure and passing temp-table (1 index here) takes almost as long as initially populating the temp-table
- DBI file size is doubled

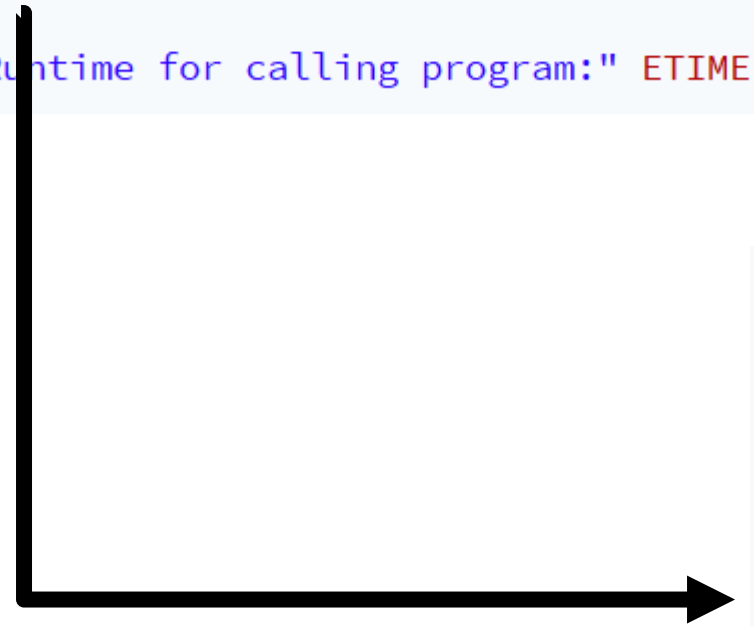
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Let‘s try it out – INPUT-OUTPUT PARAMETER TABLE

```
ETIME (YES) .

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

MESSAGE "Runtime for calling program:" ETIME "msecs" .
```



```
BLOCK-LEVEL ON ERROR UNDO, THROW.

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

DEFINE INPUT-OUTPUT PARAMETER TABLE FOR ttCustomer .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Runtime statistics – INPUT-OUTPUT PARAMETER

```
12.6 64 c:\Work\Presentations\TempTableAndDatasetParameter\TempTableParameter>_progres -p tt-param-s
Number of indexes: 1
Runtime for creating records: 3604 msecs
TT records created: 201122
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TempTableParameter\DBI1139636a56748
DBI File Size: 41,4 MB
Runtime for calling program: 6094 msecs
DBI File Size: 82,9 MB
```

- INPUT-OUTPUT is passing the Temp-Table twice “by-value”
- Runtime for calling the procedure nearly doubled (3.2 to 6.1 seconds)
- DBI file size not affected

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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!

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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 .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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, ?)

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
RUN tt-param-callee-by-reference2.p PERSISTENT SET hProcedure .
RUN internal-procedure IN hProcedure (OUTPUT TABLE ttCustomer BY-REFERENCE) .

DEFINE QUERY qry FOR ttCustomer.
OPEN QUERY qry PRESELECT EACH ttCustomer.
MESSAGE "number-of records:" QUERY qry:NUM-RESULTS .
```

- Records populated in main-block of callee
- Caller receives zero records when calling with BY-REFERENCE
- Caller receives all records when calling by-value - without the BY-REFERENCE keyword on the RUN)

```
FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
END.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# REFERENCE-ONLY definitions

- Temp-Tables defined with REFERENCE-ONLY provide compile-time schema
- But no Temp-Table instance will be created when running the program
- Relies on BY-REFERENCE (or BIND) call from another procedure
- Trying to access the Temp-Table before foreign reference is received will result in an error

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# REFERENCE-ONLY definitions

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

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

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
END.
```

```
Attempt to reference uninitialised temp-table. (12378)
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# REFERENCE-ONLY definitions

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

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

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

    FOR EACH Customer NO-LOCK:
        CREATE ttCustomer .
        BUFFER-COPY Customer TO ttCustomer .
    END.

END PROCEDURE .
```

```
internal-procedure tt-param-callee-by-reference3.p BIND or BY-REFERENCE modifier required on
RUN, FUNCTION or METHOD invocation parameter when called TABLE or DATASET parameter ttCustome
r is REFERENCE-ONLY and is still not bound. (13012)
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Agenda

- Introduction
- Temp-Tables as Parameters
- **BIND with Temp-Table Parameters**
- Table-Handle Parameters
- Temp-Table vs. local Database

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# BIND with Temp-Table Parameters

- Similar to BY-REFERENCE allows two (or more) procedure files or object instances to share the same Temp-Table instance
- BIND provides a permanent coupling between multiple procedures and the Temp-Table
- BY-REFERENCE is temporary during the execution of a single procedure or object method
- BIND can be INPUT or OUTPUT
- BIND requires receiving Temp-Table definition to be REFERENCE-ONLY

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# BIND with Temp-Table Parameters

- Attempt to BIND non REFERENCE-ONLY Temp-Table definition:

```
bind-table tt-param-callee-bind2.p BIND modifier not allowed for cases where neither the caller nor the called
 TABLE or DATASET parameter ttCustomer has been defined REFERENCE-ONLY and neither caller nor called parameter
 is a TABLE-HANDLE or DATASET-HANDLE. (13009)
```

- Attempt to BIND with BIND keyword missing on PARAMETER definition:

```
bind-table tt-param-callee-bind.p If a caller or called REFERENCE-ONLY parameter ttCustomer will become BOUND
to its opposite as a result of the call, then the BIND keyword must be supplied on both the caller invocation
and on the called parameter definition. (13161)
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# OUTPUT BIND

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

RUN tt-param-callee-bind.p PERSISTENT SET hProcedure.
RUN bind-table IN hProcedure (OUTPUT TABLE ttCustomer BIND).

DEFINE QUERY qry FOR ttCustomer.
OPEN QUERY qry PRESELECT EACH ttCustomer.
MESSAGE "number-of records:" QUERY qry:NUM-RESULTS .
```

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

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

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
END.

PROCEDURE bind-table:
    DEFINE OUTPUT PARAMETER TABLE FOR ttCustomer BIND .
END PROCEDURE.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# INPUT BIND

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

```
RUN tt-param-callee-bind2.p PERSISTENT SET hProcedure .
RUN bind-table IN hProcedure (TABLE ttCustomer BIND) .

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
END.

RUN count-records IN hProcedure .
```

```
BLOCK-LEVEL ON ERROR UNDO, THROW.

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

PROCEDURE bind-table:
    DEFINE INPUT PARAMETER TABLE FOR ttCustomer BIND .
END PROCEDURE.

PROCEDURE count-records:
    DEFINE QUERY qry FOR ttCustomer.
    OPEN QUERY qry PRESELECT EACH ttCustomer.
    MESSAGE "number-of records:" QUERY qry:NUM-RESULTS .
END PROCEDURE .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# BIND is permanent

- Once a REFERENCE-ONLY Temp-Table definition is bound to an instance at runtime, it cannot be bound to another instance

```
RUN tt-param-callee-bind3.p PERSISTENT SET hProcedure.
RUN bind-table IN hProcedure (OUTPUT TABLE ttCustomer BIND).

RUN tt-param-callee-bind3.p PERSISTENT SET hProcedure2.
RUN bind-table IN hProcedure2 (OUTPUT TABLE ttCustomer BIND).
```

```
bind-table tt-param-callee-bind3.p If BIND used when caller TABLE or DATASET parameter is bound and called
 parameter ttCustomer is also bound, the caller and called parameters must be the same instance. (13011)
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# BIND survives delete of instantiating procedure

- Prior to running bind-table in hProcedure the temp-table ttCustomer is not usable. `Cannot access the HANDLE attribute because the widget does not exist. (3140)`
- After DELETE OBJECT of the persistent procedure, the temp-table remains accessible

```
RUN tt-param-callee-bind3.p PERSISTENT SET hProcedure.
RUN bind-table IN hProcedure (OUTPUT TABLE ttCustomer BIND).

DELETE OBJECT hProcedure .

MESSAGE VALID-OBJECT (hProcedure) .

MESSAGE TEMP-TABLE ttCustomer:INSTANTIATING-PROCEDURE:FILE-NAME .

FIND FIRST ttCustomer .
MESSAGE ttCustomer.Name .
```

© 2023 Consultingwerk Software Services Ltd. All rights re

# 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# 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.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
DEFINE VARIABLE oCustomerTempTableHolder AS CustomerTempTableHolder NO-UNDO.

oCustomerTempTableHolder = NEW CustomerTempTableHolder() .
oCustomerTempTableHolder:BindTable(OUTPUT TABLE ttCustomer BIND) .

FOR EACH Customer NO-LOCK:
    CREATE ttCustomer .
    BUFFER-COPY Customer TO ttCustomer .
END.

RUN tt-param-callee-bind4.p PERSISTENT SET hProcedure (oCustomerTempTableHolder).
```

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

DEFINE INPUT PARAMETER oCustomer AS CustomerTempTableHolder NO-UNDO .

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

oCustomer:BindTable(OUTPUT TABLE ttCustomer BIND) .

DEFINE QUERY qry FOR ttCustomer.
OPEN QUERY qry PRESELECT EACH ttCustomer.
MESSAGE "number-of records:" QUERY qry:NUM-RESULTS .
```

© 2023 Consultingwerk Software Services Ltd. All

# Temp-Table Definition in Include Files

- As Temp-Tables can be shared between multiple programs/objects their definition should be stored in an Include File
- Use compile time parameter for REFERENCE-ONLY

```
/* ttCustomer.i */

DEFINE TEMP-TABLE ttCustomer NO-UNDO {&REFERENCE-ONLY}
    LIKE Customer
    INDEX CustNum IS PRIMARY UNIQUE CustNum
    .
```

```
{BindParameter/ttCustomer.i}
```

```
{BindParameter/ttCustomer.i &REFERENCE-ONLY=REFERENCE-ONLY}
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Agenda

- Introduction
- Temp-Tables as Parameters
- BIND with Temp-Table Parameters
- **Table-Handle Parameters**
- Temp-Table vs. local Database

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# TABLE-HANDLE parameters

- Aren‘t TABLE-HANDLE parameters much simpler as they are like passing around handles?
- Not at all – a TABLE-HANDLE parameter does not just pass around a handle of a Temp-Table
- A TABLE-HANDLE parameter **by default provides also a “by-value”** deep-copy (record by record) of a Temp-Table to/from the callee
- The same rules apply regarding BY-REFERENCE and BIND passing

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# TABLE-HANDLE parameters

- TABLE-HANDLE parameters are only adding support for a Temp-Table schema not yet known at compile-time
- TABLE-HANDLE parameters have rare use-cases in business logic
- No developer should be forced to use hTempTable::CustNum in Business Logic
- Typically used in interface code or API's
- TABLE and TABLE-HANDLE parameters compatible with each other as long as the Temp-Table schema matches (in both directions)

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Using TABLE-HANDLE Parameters

```
DO i = 1 TO 5:
    MESSAGE "---------------" .

    ETIME (YES).
    RUN tth-param-callee.p (TABLE ttCustomer) .

    MESSAGE "Iteration:" i "Runtime:" ETIME .
END.
```

```
DEFINE INPUT PARAMETER TABLE-HANDLE hTable .

DEFINE VARIABLE hQuery AS HANDLE NO-UNDO.
DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.

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

ASSIGN hBuffer = hTable:DEFAULT-BUFFER-HANDLE .

CREATE QUERY hQuery.
hQuery:ADD-BUFFER (hBuffer) .
hQuery:QUERY-PREPARE (SUBSTITUTE ("preselect each &1", hBuffer:NAME))
hQuery:QUERY-OPEN () .

MESSAGE "count of records:" hQuery:NUM-RESULTS .

FINALLY:
    IF VALID-HANDLE (hQuery) THEN
        DELETE OBJECT hQuery .
END FINALLY.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved

```
Number of indexes: 1
Runtime for creating records: 2114 msecs
TT records created: 201122
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TableHandleParameter\DBI140488a50464
DBI File Size: 35,2 MB
----------------
count of records: 201122
Iteration: 1 Runtime: 1412
Iteration: 1 DBI File Size: 70,4 MB
----------------
count of records: 201122
Iteration: 2 Runtime: 1379
Iteration: 2 DBI File Size: 105,5 MB
----------------
count of records: 201122
Iteration: 3 Runtime: 1370
Iteration: 3 DBI File Size: 140,6 MB
----------------
count of records: 201122
Iteration: 4 Runtime: 1414
Iteration: 4 DBI File Size: 175,7 MB
----------------
count of records: 201122
Iteration: 5 Runtime: 1440
Iteration: 5 DBI File Size: 210,8 MB
```

© 2018 Consultingwerk Software Services Ltd. All rights reserved.

# Houston, we have a problem!

- **Apparently the DBI file grows constantly with every iteration!**
- Let's look closer …
- Enable DynObjects.DB and 4GLTrace client log entries

```
LOG-MANAGER:LOGFILE-NAME = "client.log".
LOG-MANAGER:CLEAR-LOG () .
LOG-MANAGER:LOG-ENTRY-TYPES = "DynObjects.DB:4,4GLTrace" .
```

- Close client-logfile before existing procedure

```
LOG-MANAGER:CLOSE-LOG () .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# DynObjects.DB client logfile entries

```
4GLTRACE      Run tth-param-callee.p "ttCustomer" [Main Block - tth-param-start.p @ 130]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1002 (tth-param-callee.p @ 0)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1004 (tth-param-callee.p @ 0) IMPLICIT Table:ttCustomer Pool:<Session Pool>
DYNOBJECTS    Created        QUERY                   Handle:1005 (tth-param-callee.p @ 27)  Pool:<Session Pool>
DYNOBJECTS    Deleted        QUERY                   Handle:1005 (tth-param-callee.p @ 36)
4GLTRACE      Run tth-param-callee.p "ttCustomer" [Main Block - tth-param-start.p @ 130]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1007 (tth-param-callee.p @ 0)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1008 (tth-param-callee.p @ 0) IMPLICIT Table:ttCustomer Pool:<Session Pool>
DYNOBJECTS    Created        QUERY                   Handle:1009 (tth-param-callee.p @ 27)  Pool:<Session Pool>
DYNOBJECTS    Deleted        QUERY                   Handle:1009 (tth-param-callee.p @ 36)
4GLTRACE      Run tth-param-callee.p "ttCustomer" [Main Block - tth-param-start.p @ 130]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1011 (tth-param-callee.p @ 0)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1012 (tth-param-callee.p @ 0) IMPLICIT Table:ttCustomer Pool:<Session Pool>
DYNOBJECTS    Created        QUERY                   Handle:1013 (tth-param-callee.p @ 27)  Pool:<Session Pool>
DYNOBJECTS    Deleted        QUERY                   Handle:1013 (tth-param-callee.p @ 36)
4GLTRACE      Run tth-param-callee.p "ttCustomer" [Main Block - tth-param-start.p @ 130]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1015 (tth-param-callee.p @ 0)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1016 (tth-param-callee.p @ 0) IMPLICIT Table:ttCustomer Pool:<Session Pool>
DYNOBJECTS    Created        QUERY                   Handle:1017 (tth-param-callee.p @ 27)  Pool:<Session Pool>
DYNOBJECTS    Deleted        QUERY                   Handle:1017 (tth-param-callee.p @ 36)
4GLTRACE      Run tth-param-callee.p "ttCustomer" [Main Block - tth-param-start.p @ 130]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1019 (tth-param-callee.p @ 0)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1020 (tth-param-callee.p @ 0) IMPLICIT Table:ttCustomer Pool:<Session Pool>
DYNOBJECTS    Created        QUERY                   Handle:1021 (tth-param-callee.p @ 27)  Pool:<Session Pool>
DYNOBJECTS    Deleted        QUERY                   Handle:1021 (tth-param-callee.p @ 36)
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Reading DynObject client-logfile entries

- Read this K-Base article for more details
- https://community.progress.com/s/article/P133306

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Fixing the memory leaks

- Option A: Deleting the Temp-Table

```
IF VALID-HANDLE (hTable) AND hTable:NUM-REFERENCES = 0 THEN
    DELETE OBJECT hTable .
```

- We will see the DBI file grow once
- This is because the caller has his instance of the temp-table and there is always a second instance for the callee
- But no memory leak anymore

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
Number of indexes: 1
Runtime for creating records: 2093 msecs
TT records created: 201122
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TableHandleParameter\DBI122212a01628
DBI File Size: 35,2 MB
----------------
count of records: 201122
Iteration: 1 Runtime: 1374
Iteration: 1 DBI File Size: 70,4 MB
----------------
count of records: 201122
Iteration: 2 Runtime: 1389
Iteration: 2 DBI File Size: 70,4 MB
----------------
count of records: 201122
Iteration: 3 Runtime: 1378
Iteration: 3 DBI File Size: 70,4 MB
----------------
count of records: 201122
Iteration: 4 Runtime: 1378
Iteration: 4 DBI File Size: 70,4 MB
----------------
count of records: 201122
Iteration: 5 Runtime: 1403
Iteration: 5 DBI File Size: 70,4 MB
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Fixing the memory leaks

- Option B: BY-REFERENCE passing to the TABLE-HANDLE parameter

```
RUN tth-param-callee.p (TABLE ttCustomer BY-REFERENCE) .
```

- Temp-Table will be passed BY-REFERENCE, no deep-copy happening, so less memory and less runtime (win-win)
- No change in the callee, still checking for NUM-REFERENCES before deleting Temp-Table, as **callee cannot enforce BY-REFERENCE** ☹

```
FINALLY:
    IF VALID-HANDLE (hTable) AND hTable:NUM-REFERENCES = 0 THEN
        DELETE OBJECT hTable .

    IF VALID-HANDLE (hQuery) THEN
```

```
Number of indexes: 1
Runtime for creating records: 2133 msecs
TT records created: 201122
DBI File: C:\Work\Presentations\TempTableAndDatasetParameter\TableHandleParameter\DBI65940a74724
DBI File Size: 35,2 MB
----------------
count of records: 201122
Iteration: 1 Runtime: 218
Iteration: 1 DBI File Size: 35,3 MB
----------------
count of records: 201122
Iteration: 2 Runtime: 222
Iteration: 2 DBI File Size: 35,3 MB
----------------
count of records: 201122
Iteration: 3 Runtime: 235
Iteration: 3 DBI File Size: 35,3 MB
----------------
count of records: 201122
Iteration: 4 Runtime: 222
Iteration: 4 DBI File Size: 35,3 MB
----------------
count of records: 201122
Iteration: 5 Runtime: 221
Iteration: 5 DBI File Size: 35,3 MB
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# OUTPUT TABLE-HANDLE Parameters

- Typical use-case would be a “dispatcher” procedure accepting calls on the AppServer

```
DEFINE INPUT  PARAMETER pcProcName AS CHARACTER NO-UNDO.
DEFINE OUTPUT PARAMETER TABLE-HANDLE hTable .

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

IF pcProcName = "customer" THEN
    ASSIGN pcProcName = "tth-param-callee2.p" .

RUN VALUE (pcProcName) (OUTPUT TABLE-HANDLE hTable) .
```

© 2

# DynObjects.DB client logfile entries

```
4GLTRACE      Run tth-dispatcher.p "customer" [Main Block - tth-param-start2.p @ 108]
4GLTRACE      Run tth-param-callee2.p [Main Block - tth-dispatcher.p @ 26]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1003 (tth-param-callee2.p @ 30)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1005 (tth-param-callee2.p @ 30) IMPLICIT Table:ttCustomer Pool:<Session Pool>
4GLTRACE      Run tth-dispatcher.p "customer" [Main Block - tth-param-start2.p @ 108]
4GLTRACE      Run tth-param-callee2.p [Main Block - tth-dispatcher.p @ 26]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1008 (tth-param-callee2.p @ 30)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1010 (tth-param-callee2.p @ 30) IMPLICIT Table:ttCustomer Pool:<Session Pool>
4GLTRACE      Run tth-dispatcher.p "customer" [Main Block - tth-param-start2.p @ 108]
4GLTRACE      Run tth-param-callee2.p [Main Block - tth-dispatcher.p @ 26]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1013 (tth-param-callee2.p @ 30)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1015 (tth-param-callee2.p @ 30) IMPLICIT Table:ttCustomer Pool:<Session Pool>
4GLTRACE      Run tth-dispatcher.p "customer" [Main Block - tth-param-start2.p @ 108]
4GLTRACE      Run tth-param-callee2.p [Main Block - tth-dispatcher.p @ 26]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1018 (tth-param-callee2.p @ 30)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1020 (tth-param-callee2.p @ 30) IMPLICIT Table:ttCustomer Pool:<Session Pool>
4GLTRACE      Run tth-dispatcher.p "customer" [Main Block - tth-param-start2.p @ 108]
4GLTRACE      Run tth-param-callee2.p [Main Block - tth-dispatcher.p @ 26]
DYNOBJECTS    Created        TEMP-TABLE              Handle:1023 (tth-param-callee2.p @ 30)  Pool:<Session Pool>
DYNOBJECTS    Created        BUFFER                  Handle:1025 (tth-param-callee2.p @ 30) IMPLICIT Table:ttCustomer Pool:<Session Pool>
----------    Log file closed at user's request
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Fixing the memory leaks

- Option A: Deleting the Temp-Table

```
IF VALID-HANDLE (hTable) AND hTable:NUM-REFERENCES = 0 THEN
    DELETE OBJECT hTable .
```

- Looks weird (hurts my eyes every time) that we DELETE OBJECT the Temp-Table before it's returned to the caller
- But the AVM defers deletion of the Temp-Table object handle until the program is finished

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
----------------
Iteration: 1 Runtime: 4377
Iteration: 1 DBI File Size: 70,4 MB
----------------
Iteration: 2 Runtime: 4326
Iteration: 2 DBI File Size: 105,5 MB
----------------
Iteration: 3 Runtime: 4290
Iteration: 3 DBI File Size: 105,5 MB
----------------
Iteration: 4 Runtime: 4315
Iteration: 4 DBI File Size: 105,5 MB
----------------
Iteration: 5 Runtime: 4495
Iteration: 5 DBI File Size: 105,5 MB
201122 records.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Fixing the memory leaks

- Option B: BY-REFERENCE passing to/from the TABLE-HANDLE parameter

```
RUN tth-param-callee.p (TABLE ttCustomer BY-REFERENCE) .
```

- Temp-Table will be passed BY-REFERENCE, no deep-copy happening, so less memory and less runtime (win-win)
- No change in the callee, still checking for NUM-REFERENCES before deleting Temp-Table, as **callee cannot enforce BY-REFERENCE** ☹

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
----------------
Iteration: 1 Runtime: 2097
Iteration: 1 DBI File Size: 35,3 MB
----------------
Iteration: 2 Runtime: 2012
Iteration: 2 DBI File Size: 35,3 MB
----------------
Iteration: 3 Runtime: 2059
Iteration: 3 DBI File Size: 35,3 MB
----------------
Iteration: 4 Runtime: 2032
Iteration: 4 DBI File Size: 35,3 MB
----------------
Iteration: 5 Runtime: 2027
Iteration: 5 DBI File Size: 35,3 MB
201122 records.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Call details

- (1) tth-param-start2.p calls tth-dispatcher.p with OUTPUT TABLE BY-REFERENCE ← TABLE-HANDLE
- (2) tth-dispatcher.p calls tth-param-callee.p with OUTPUT TABLE-HANDLE BY-REFERENCE ← TABLE
- (1) passes Temp-Table from caller to callee despite OUTPUT nature
- (2) passes that same Temp-Table from caller to callee
- Schema compatibility will be verified on the RUN (2) – prior this was when returning from (1) - so much later
- (1) passes Temp-Table also with records – so we need to empty the TempTable when executing multiple times

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Agenda

- Introduction
- Temp-Tables as Parameters
- BIND with Temp-Table Parameters
- Table-Handle Parameters
- **Temp-Table vs. local Database**

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Temp-Table vs. local Database

- Database Tables and Temp-Tables are organized in a very similar manner
- Records are stored in blocks; blocks are stored on disk and are read into a buffer cache when required
- Similar options for block size and number of blocks
- Read performance of temp-tables is not generally faster than a local database connected via shared-memory
- Performance of both types of “tables” benefits in similar way from data in buffer (-B, -Bt) and suffer when read from disc

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Temp-Table vs. local Database

- Temp-Table is of temporary nature
- Database tables are persisted in the database
- **No mechanism to share** Temp-Tables between AppServer agents or PASOE sessions
- **Database tables can be accessed from multiple** AppServer agents or PASOE sessions
- Management of schema easier with Temp-Tables
- More options for performance of database, including private buffers, storage areas, APW, BIW

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Caching of data

- Some applications cache data (on demand or at AppServer startup) in Temp-Tables
- Useful only when at least one of the following is true
  - Database not on same machine as PASOE (TCP/IP connect)
  - Cached data with large degree of aggregation, e.g.
    - User security derived from multiple user groups flattened
    - Localized messages
- Nature of data is typically not temporary – as it may live for hours, days, …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Caching in database vs. Temp-Tables

- A local **cache Database** can be shared by multiple AppServer agents or PASOE sessions
- Temp-Tables only accessible by a single AppServer agent (classic AppServer) or PASOE session
- -B on the local cache Database provides value to all AppServer agents or PASOE sessions
- -Bt on PASOE should be reserved for “real” temporary temp-tables

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Caching in database vs. Temp-Tables

- Focus here on AppServer … as a GUI client is typically remote from the database (unless on Terminal Server)

- In short: A database table may be a better solution for certain temp-tables …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

# Questions

© 2023 Consultingwerk Software Services Ltd. All rights reserved.


Consultingwerk
software architecture and development