Blaise 5 Manipula

Selected Topics

Performance Improvements

Processing ASCII/Blaise Files, C# engine, Lazy Fields

Performance Improvements

- 2 approaches for speed improvement:
  - Stream-based storage of ASCII or BLAISE record: instead of zipped XML, just copy the record stream into the .bdbx file
  - Conversion of manipula script into compiled C# code, instead of executing the compiled .msux script
- Speedup is turned on by global setting ENGINE=CSHARP
- Speedup already works in 5.13 for ASCII files, in 5.14 it also works for BLAISE files
- Not all manipula constructs are supported, e.g. reflection; in that case, execution reverts back to normal.

Performance Improvements

Conversion of the ‘old’ format to the stream-based format:

```
settings
    engine=csharp
    autoread=no

  USES
      vm2021u  '..\vm2021u'

  INPUTFILE   INF1    : vm2021u      ('..\Stap07_Respons_Na_Leegpoetsen_LON', blaise)
OUTPUTFILE   OUTF1    : vm2021u      ('..\Stap07_Respons_Na_Leegpoetsen_LON_SMALL_STR', blaise)

manipulate

  while inf1.recordnumber < 100 do
      inf1.READNEXT
      outf1.write
   endwhile

```

Performance Improvements

Script name	M4	M5	M5++	M4	M5++
# records	10.000	10.000	10.000	100.000	100.000
Conversie oud->nieuw	NA	NA	85	NA	683
Stap07a_Afleiden_AFL_01tm08	39	265	15	249	166
Stap07b_Afleiden_AFL_09tm99	31	304	16		180
Stap07c_Afleiden_PUB_01tm19	61	275	25		
Stap07d_Afleiden_PUB_20tm99	32	349	21		
Stap09_Respons_Afbakenen	19	300	6		
Stap13a_BLAISE2ASCII	17	176	45		
Totaal	199	1669	213		
Ratio M5/M4		8,39	1,07		

Timing is in seconds, NA = Not Applicable
M4: manipula 4
M5: current manipula 5
M5++: current manipula 5 with setting ENGINE=CSHARP

Performance Improvements

- Setting LazyFields=Yes
  - Delayed creation of fields
  - Can speed up your scripts
    - Large datamodels and a relative small number of fields being processed
    - Number of 'auto-copy' fields between input and output is large

- Working on:
  - SORT improvement
  - Usage of multiple threads in combination with previous methods
  - Know when to validate ASCII data

Query File

Execute SQL Queries directly in Manipula

Query File

- F.Open(<query>[, <isNative>]) where F is query file
- Query must start with Select, GetTables, GetColumns, GetIndexes etc.
- Not allowed to use Drop, Delete, Insert, Create etc.

- Related:
  - F.ExecuteQuery(<query>, G [, <isNative>]) where F is input file and G query file
  - F.ExecuteNonQuery(<command> [, <isNative>])

- Sample ..\Specific Features\Custom Reports\CallResultOverview: query on CATI.db

- Demo

Survey Data File

Flexible way of accessing installed surveys

Survey Data File

- Two use cases:
  - In Interceptor and Action Setups to access survey data
  - More general: replacement for update files with relative Bdix (installed surveys):

```
USES
   surveymeta (VAR)

SURVEYDATAFILE
   sdf: surveymeta

AUXFIELDS
   ConnectionStr: STRING
   Res: INTEGER

MANIPULATE
  surveymeta.LOADDATAMODEL('MyModel.bmix')

  ConnectionStr := 'InstrumentID=383fd6b8-779d-4db7-9385-49fcc4e368e;Serverpark=LocalDevelopment;'
    +'Host=LocalHost;Port=8033;UserName=Root;Password=Root'
  sdf.SETCONNECTIONSTRING(ConnectionStr)
  IF sdf.ResultOK THEN
     res := sdf.OPEN
  ENDIF
```

Reflection using Server Manager API

Users, Roles etc.

Reflection using Server Manager API

- strRes := [[*servermanager*.Connect()]]
- strRes := [[*servermanager*.Connect('localhost', 8031, 'Root', 'Root')]]
- strRes := [[*servermanager*.Connect(SurveyList)]]

```
for j:= 0 to VAL([[servermanager.Users.Count]])-1 do
   sUserName:= [[servermanager.Users[j].Name]]
   sUserRole:= [[servermanager.Users[j].Role]]
   Display('user= ' + sUserName + ', ' + sUserRole, wait)
enddo
```

Servermanager ⇔ IConnectedServerX interfaces

Reflection using Server Manager API

Demo

More on reflection: ..\Samples\Specific Features\Manipula\ :

- ListAllFields
- ListAllTexts
- ProcessAllChecks

Reading/Writing of Session Data

GetSessionRecord, UpdateSession

Reading/Writing of Session Data

- F is input/update file of relative bdix or survey data file:
  - F.GetSessionData(T)
  - F.GetSessionRecord(<keyValue>)

- In 5.13: no access to creation date, active field, language etc.
- In 5.14: only read access:
  - F.GetDatetimeSessionVar('Creation’)
  - F.GetDatetimeSessionVar('LastModification’)
  - F.GetStringSessionVar(‘ActiveFieldName’)
  - F.GetStringSessionVar(‘LanguageName’)

- F.UpdateSession
  - F is update file of relative bdix or survey data file

Reading/Writing of Session Data

- Scenario 1: Modification of session records

Example

```
SETUP TS
 SETIINGS
   AUTOREAD=NO

 USES
   DM 'SyncModel.bmix'
 UPDATEFILE
   uf1: DM ('RelativeSyncModel.bdix', Bdix)

 TEMPORARYFILE
   G: DM
 AUXFIELDS
   i: INTEGER
 MANIPULATE
   uf1.GETSESSIONDATA(G)
   FOR i := 1 TO G.RECORDCOUNT DO
     G.READNEXT

     uf1.GETSESSIONRECORD(G.KeyUnformatted)
     IF uf1.RESULTOK THEN
    // ... modify field values, field properties etc. of current record of uf1
       uf1.UPDATESESSION
     ENDIF
   ENDDO
```

Reading/Writing of Session Data

- Scenario 2: Copying specific session records to separate file

Example

```
PROCESS ExtractSessionData

 USES InMeta 'QBS.bmix'

 INPUTFILE in1:InMeta('QBS', BDIX)

 TEMPORARYFILE F: InMeta

 OUTPUTFILE out1: InMeta('QBSSession', BLAISE)

 AUXFIELDS res: INTEGER

 MANIPULATE
  res:= in1.GETSESSIONDATA(F)
  IF res=0 AND F.FORMCOUNT>0 THEN
    REPEAT
    F.READNEXT
      IF F.GETSTRINGSESSIONVAR('LanguageName') = 'ENG' THEN
        out1.WRITE
      ENDIF
    UNTIL F.LASTRECORD
  ENDIF
```

Action Setups and Server Variables

SurveyRecord, Set*ServerVar and Get*ServerVar

Action Setups and Server Variables

- The setup must be defined as Actions setup in the Project Properties
- Uses SurveyRecord to access current interview
- Uses procedures/functions to be called on certain events

- Define several types of variables like string, number, date, time …
- Makes use of Set and Get methods
- Examples:
  - F.SetStringServerVar(<varName>, <stringValue>)
  - F.SetNumberServerVar(<varName>, <integerValue>)
  - ID := F.GetStringServerVar('InstrumentID')
  - MaxResponses := F.GetNumberServerVar('MaxResp')

Action Setups and Server Variables

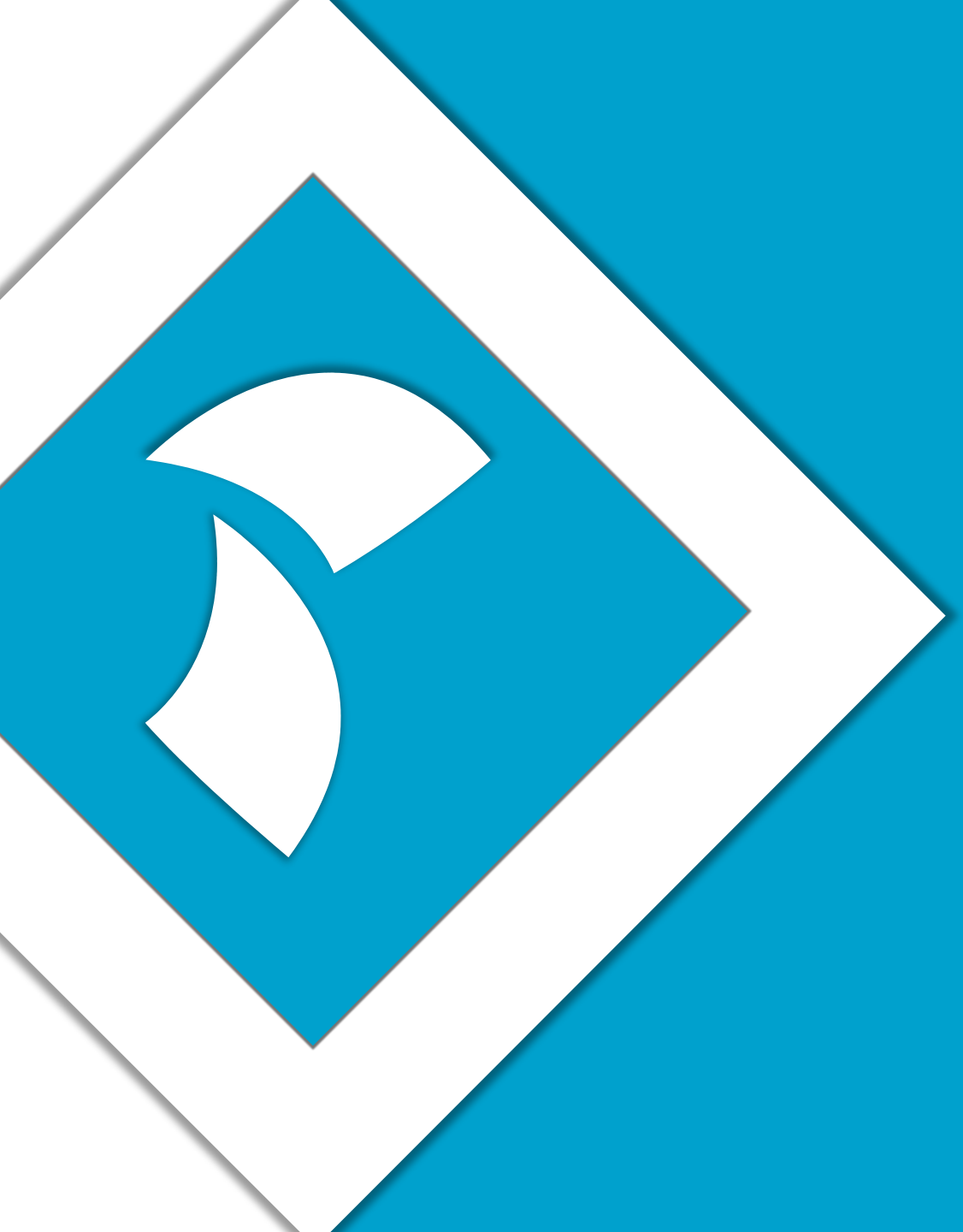
- Can be defined and used in Action setups called from the same data entry session
- Works also in the Apps

History File

Accessing historical records in Manipula

History File

- Make sure to add history table to Bdix
- File definition HistoryFile H defines file for accessing record history
- F.GetHistory(H [, <recordFilter>)
  - H resembles temporary file
  - H is read-only
  - Most recent ones first (*order by HistoryID desc*)
  - History related columns only accessible through reflection
- Bdix can not be relative (yet): F is inputfile with direct .bdix
- Demo



Thank you for your time

Gaining **deeper** understanding

www.blaise.com

blaise@cbs.nl

@blaisecbs

@Blaise5