2023-02-23_Parse_IDMS_schema_source

February 23, 2023

1 Parsing IDMS schema syntax

- primer
- IDMS schema and subschema syntax

1.1 syntax components

- areas
- records - table
- elements - field
- sets

2 Import libraries

[1]:
```
import pandas as pd
from pathlib import Path
import re
from collections import Counter
```

3 List of all the IDMS subschema files we were provided

[2]:
```
pwd
```

[2]:
```
'/Users/ccoletta/projects/gsa_coe/usda/copybooks'
```

3.1 Do a line count for each file we received

[3]:
```
!wc -l  IDMS_schema_source_ALL/*
```

```
    2375 IDMS_schema_source_ALL/EMPSCHM-V100-SCHEMA-020623.txt
   16234 IDMS_schema_source_ALL/FARMS-V1-SCHEMA-020623.formatted.txt
   16234 IDMS_schema_source_ALL/FARMS-V1-SCHEMA-020623.txt
   16642 IDMS_schema_source_ALL/FARMS-V10-SCHEMA-020623.txt
   25112 IDMS_schema_source_ALL/MCMMF01-SCHEMA-020623.txt
      47 IDMS_schema_source_ALL/RDUSE01-SCHEMA-020623.txt
    1599 IDMS_schema_source_ALL/RMS-SCHEMA-020623.txt
    3387 IDMS_schema_source_ALL/SCHEMA-LISTING-020623.zip
```

```
  1779 IDMS_schema_source_ALL/SCMAC01-SCHEMA-020623.txt
    50 IDMS_schema_source_ALL/SCMACCT-SCHEMA-020623.txt
  1538 IDMS_schema_source_ALL/SCMFA01-SCHEMA-020623.txt
   263 IDMS_schema_source_ALL/SCMHN01-SCHEMA-020623.txt
 26281 IDMS_schema_source_ALL/SCMMF01-SCHEMA-020623.txt
 26039 IDMS_schema_source_ALL/SCMMF01L-SCHEMA-0206023.txt
 26041 IDMS_schema_source_ALL/SCMMF01M-SCHEMA-020623.txt
 26177 IDMS_schema_source_ALL/SCMMF01P-SCHEMA-020623.txt
 25988 IDMS_schema_source_ALL/SCMMF02-SCHEMA-020623.txt
   924 IDMS_schema_source_ALL/SCMQA01-SCHEMA-020623.txt
 23935 IDMS_schema_source_ALL/SCMTEST-SCHEMA-020623.txt
  2020 IDMS_schema_source_ALL/SCMTL01-SCHEMA-020623.txt
  1924 IDMS_schema_source_ALL/SCMUA01-SCHEMA-020623.txt
   196 IDMS_schema_source_ALL/SCMXREF-SCHEMA-020623.txt
 16590 IDMS_schema_source_ALL/XARMS-SCHEMA-020623.txt
 23992 IDMS_schema_source_ALL/XCMMF01-SCHEMA-020623.txt
 26179 IDMS_schema_source_ALL/XSCMMF01-SCHEMA-020623.txt
311546 total
```

4 Examine the contents of one of the subschema files

[4]:
```
!head -300 IDMS_schema_source_ALL/FARMS-V1-SCHEMA-020623.txt | tail -50
```

```
*+           USAGE IS DISPLAY
*+           ELEMENT LENGTH IS 19
*+           POSITION IS 24
*+           .
*+       10 NME-ADRS-OBLR-1
*+           PICTURE IS  X(19)
*+           USAGE IS DISPLAY
*+           ELEMENT LENGTH IS 19
*+           POSITION IS 43
*+           .
*+       10 NME-ADRS-OBLR-2
*+           PICTURE IS  X(19)
*+           USAGE IS DISPLAY
*+           ELEMENT LENGTH IS 19
*+           POSITION IS 62
*+           .
*+   05 NME-OBLR-GRP-2
*+       REDEFINES NME-OBLR-GRP-1
*+       USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 57
*+       POSITION IS 24
*+       .
*+       10 NME-OBLR-KEY
*+           PICTURE IS  X(20)
```

```
*+           USAGE IS DISPLAY
*+           ELEMENT LENGTH IS 20
*+           POSITION IS 24
*+           .
*+       10 FILLER
*+           PICTURE IS  X(0037)
*+           USAGE IS DISPLAY
*+           ELEMENT LENGTH IS 37
*+           POSITION IS 44
*+           .
*+   05 NME-ADRS-OBLR-3
*+       PICTURE IS  X(19)
*+       USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 19
*+       POSITION IS 81
*+       .
*+   05 NME-ADRS-OBLR-4
*+       PICTURE IS  X(19)
*+       USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 19
*+       POSITION IS 100
*+       .
*+   05 ZIP-CDE
*+       PICTURE IS  9(5)
*+       USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 5
```

5 Create some regular expressions to help us parse schema text

[5]:
```
#print( record_components[0] )
```

[6]:
```
redefines_pattern = re.compile( r'REDEFINES' )
occurs_pattern = re.compile( r' OCCURS ' )
```

[7]:
```
column_names = [ 'indent', 'data_level', 'element_name',␣
 ↪'raw_element_descriptors' ]
element_search_pat = re.compile( r'^(\s+)?(\d\d) (\S+)\n\s+(.*?)\n\s+\.',␣
 ↪flags=re.MULTILINE | re.DOTALL )
```

[8]:
```
record_name_pat = re.compile( r'RECORD NAME IS (\S+)' )
```

[9]:
```
IS_pat = re.compile( r' IS ' )
```

6 Define parsing functions

[10]:
```
def DescriptorsSplitter( raw_descriptor_string : str ) -> pd.Series:
    """This function parses the descriptors after one single data element,
    including PICTURE, USAGE, ELEMENT LENGTH, POSITION, etc. Returns a single
    row's worth of element descriptors."""

    descriptors = [ _.strip() for _ in raw_descriptor_string.split( '\n' ) ]

    key_value_pairs = [ IS_pat.split( _ ) for _ in descriptors ]
    try:
        index, values = zip( *key_value_pairs )
    except:
        # empty series
        print( "\t\tproblem splitting these characteristics" )
        print( key_value_pairs )
        retval = pd.Series( dtype='object')
    else:
        retval = pd.Series( values, index=index )
    #print( retval )
    return retval
```

[11]:
```
def FormatRecord( component_text : str, debug=False ) -> pd.DataFrame:
    """Takes the raw text of one record's worth of schema definition
    and parses all data elements from it."""

    if debug:
        print( "*" * 50 )
    record_name = record_name_pat.match( component_text ).group(1)
    data_elements = element_search_pat.findall( component_text )
    if debug:
        print( "record", record_name, "has", len( data_elements ), "elements." )
    data_elements = pd.DataFrame( data_elements, columns = column_names )
    data_elements['record'] = record_name
    data_elements['data_step'] = [ (1+int(_)) * 100 for _ in data_elements.
 ↪index ]
    data_elements['indent'] = data_elements['indent'].apply( len )

    # Here, we add in an "IS" to REDEFINES and OCCURS descriptors to allow for
    # DescriptorSplitter() to work in a uniform way across all descriptors that
    # already use the "IS", e.g., "PICTURE", "ELEMENT LENGTH" etc.
    data_elements['raw_element_descriptors'] = \
        data_elements['raw_element_descriptors'].str.replace(␣
 ↪redefines_pattern, 'REDEFINES IS' )

    data_elements['raw_element_descriptors'] = \
```

```
        data_elements['raw_element_descriptors'].str.replace( occurs_pattern, '␣
↪OCCURS IS ' )

    modifiers_df = data_elements['raw_element_descriptors'].apply(␣
↪DescriptorsSplitter )

    data_elements = pd.concat( (data_elements, modifiers_df), axis=1 )

    data_elements = data_elements.set_index( 'record', append=True )
    return data_elements
```

[12]:
```
def ScrapeRecordsAndElements( schema_source_path : Path ) -> pd.DataFrame:
    """Takes the path of one schema source file, parses out its component parts
    including SCHEMA, AREA, RECORD, and SET, and parses each record. Output is
    a PANDAS dataframe containing parsed info."""

    print( "=" * 50 )
    lines_df = pd.read_csv( schema_source_path, header=None )
    lines_df.columns = [ 'raw_line' ]
    lines_df[ 'stripped' ] = lines_df[ 'raw_line' ].str.slice( start = 5 )
    lines_df[ 'stripped' ] = lines_df[ 'stripped' ].str.rstrip()

    stripped_whitespace_file = Path( schema_source_path ).with_suffix( ".
↪formatted.txt" )
    print( "writing", f'"{ str(stripped_whitespace_file) }"' )
    lines_df[ 'stripped' ].to_csv( stripped_whitespace_file, header=False,␣
↪index=False )
    raw_text = stripped_whitespace_file.read_text()

    add_pat = re.compile( r'\nADD\n' )
    components = add_pat.split( raw_text )

    #len( components )
    # remove the initial ADD
    components[0] = components[0][4:]

    schema_name_ver_pat = re.compile( r'SCHEMA NAME IS (\S+) VERSION IS (\d+)' )
    schema_info = schema_name_ver_pat.search( components[0] ).groups()
    schema_name, schema_version = schema_info

    first_word_pat = re.compile( r'^(\S+)' )
    component_categories = [ first_word_pat.match( _ ).group(1) for _ in␣
↪components ]
    c = Counter( component_categories )

    print( schema_info, "\n", c.most_common() )
```

```
    # Analyze Record components

    record_components = [ c for t, c in zip( component_categories, components )␣
↪if t == "RECORD" ]

    pivoted_record_data = pd.concat( [ FormatRecord(_) for _ in␣
↪record_components ] )
    pivoted_record_data = pivoted_record_data.swaplevel().sort_index()
    return pivoted_record_data
```

7 INPUT THE SCHEMA FILE YOU WANT TO PARSE HERE:

[13]:
```
retval = ScrapeRecordsAndElements( 'IDMS_schema_source_ALL/
↪FARMS-V1-SCHEMA-020623.txt' )
```

```
==================================================
writing "IDMS_schema_source_ALL/FARMS-V1-SCHEMA-020623.formatted.txt"
('FARMS', '1')
 [('RECORD', 113), ('SET', 85), ('AREA', 20), ('SCHEMA', 1)]
```

[14]:
```
retval.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 2218 entries, ('ACCT-DATA', 0) to ('USERS', 12)
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   indent                  2218 non-null   int64
 1   data_level              2218 non-null   object
 2   element_name            2218 non-null   object
 3   raw_element_descriptors 2218 non-null   object
 4   data_step               2218 non-null   int64
 5   USAGE                   2218 non-null   object
 6   ELEMENT LENGTH          2154 non-null   object
 7   POSITION                2218 non-null   object
 8   PICTURE                 1961 non-null   object
 9   REDEFINES               23 non-null     object
 10  VALUE                   66 non-null     object
 11  OCCURS                  31 non-null     object
dtypes: int64(2), object(10)
memory usage: 229.4+ KB
```

[15]:
```
retval.head()
```

[15]:

	indent	data_level	element_name	\
record				
ACCT-DATA 0	0	05	LN-NBR	

```
          1       0         05       FD-CDE
          2       4         10     FD-CDE-3
          3       8         15     FD-CDE-2
          4       8         15   FD-CDE-3RD

                                        raw_element_descriptors  data_step  \
record
ACCT-DATA 0  PICTURE IS  9(2)\n    USAGE IS DISPLAY\n    EL…        100
          1  USAGE IS DISPLAY\n    ELEMENT LENGTH IS 4\n   …        200
          2  USAGE IS DISPLAY\n        ELEMENT LENGTH IS 3\…        300
          3  PICTURE IS  9(2)\n            USAGE IS DISPLAY…        400
          4  PICTURE IS  9(1)\n            USAGE IS DISPLAY…        500

                USAGE ELEMENT LENGTH POSITION PICTURE REDEFINES VALUE OCCURS
record
ACCT-DATA 0  DISPLAY             2        1    9(2)       NaN   NaN    NaN
          1  DISPLAY             4        3     NaN       NaN   NaN    NaN
          2  DISPLAY             3        3     NaN       NaN   NaN    NaN
          3  DISPLAY             2        3    9(2)       NaN   NaN    NaN
          4  DISPLAY             1        5    9(1)       NaN   NaN    NaN
```

8 Cleanup

8.1 Cleanup item 1: remove parentheses from valid values

[16]:
```
retval['VALUE'].value_counts()
```

[16]:
```
( 'S' )      10
( 'D' )      10
( 'C' )      10
( 'T' )       8
( 'U' )       8
( 'V' )       8
( '0' )       4
( '1' )       4
( SPACE )     2
( 'R' )       1
( 'A' )       1
Name: VALUE, dtype: int64
```

[17]:
```
retval['VALUE'] = retval['VALUE'].str.extract( r' (\S+) ' )
```

[18]:
```
retval['VALUE'].value_counts()
```

[18]:
```
'S'      10
'D'      10
'C'      10
```

```
'T'       8
'U'       8
'V'       8
'0'       4
'1'       4
SPACE     2
'R'       1
'A'       1
Name: VALUE, dtype: int64
```

8.2 Cleanup item 2: Reformat indents so copybooks look nice

[19]:
```
retval['data_level'].value_counts()
```

[19]:
```
05    1295
10     626
15     220
88      64
20      13
Name: data_level, dtype: int64
```

[20]:
```
retval['indent'].value_counts()
```

[20]:
```
0     1295
4      626
8      220
16      64
12      13
Name: indent, dtype: int64
```

[21]:
```
retval['indent'] = (retval['data_level'].astype(int) // 5 )
```

[22]:
```
retval.loc[ retval['indent'] >= 5, 'indent' ] = 5
```

[23]:
```
retval['indent'] = retval['indent'] * 2
```

[24]:
```
retval['indent'].value_counts()
```

[24]:
```
2     1295
4      626
6      220
10      64
8       13
Name: indent, dtype: int64
```

8.3 Cleanup item 3: remove leading spaces from PICTURE clause

[25]:
```
retval['PICTURE'].values
```

[25]:
```
array([' 9(2)', nan, nan, …, ' 9(1)', ' XX', ' X'], dtype=object)
```

[26]:
```
retval.head()
```

[26]:

record		indent	data_level	element_name	raw_element_descriptors	data_step	USAGE	ELEMENT LENGTH	POSITION	PICTURE	REDEFINES	VALUE	OCCURS
ACCT-DATA	0	2	05	LN-NBR	PICTURE IS 9(2)\n	100	DISPLAY	2	1	9(2)	NaN	NaN	NaN
	1	2	05	FD-CDE	USAGE IS DISPLAY\n	200	DISPLAY	4	3	NaN	NaN	NaN	NaN
	2	4	10	FD-CDE-3	ELEMENT LENGTH IS 4\n	300	DISPLAY	3	3	NaN	NaN	NaN	NaN
	3	6	15	FD-CDE-2	USAGE IS DISPLAY…	400	DISPLAY	2	3	9(2)	NaN	NaN	NaN
	4	6	15	FD-CDE-3RD	PICTURE IS 9(1)\n	500	DISPLAY	1	5	9(1)	NaN	NaN	NaN

[27]:
```
retval['PICTURE'] = retval['PICTURE'].str.strip()
```

9 Add data formatting to the csv output to get it ready for ingestion by CreateCopyBooks notebook/script

Here we are just changing the tablenames of the output spreadsheet and adding dummy columns so the next script can run unedited.

[28]:
```
table_index_dict = { table_name: i for i, table_name in enumerate( retval.index.
↪levels[0] ) }
```

[29]:
```
retval = retval.reset_index( 'record', drop=False )
```

[30]:
```
retval['table_index'] = \
    [ table_index_dict[n] for n in retval['record'].values ]
```

[31]:
```
retval['table_vers'] = 1
```

[32]:
```
retval.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2218 entries, 0 to 12
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   record                  2218 non-null   object
 1   indent                  2218 non-null   int64
 2   data_level              2218 non-null   object
 3   element_name            2218 non-null   object
 4   raw_element_descriptors 2218 non-null   object
 5   data_step               2218 non-null   int64
 6   USAGE                   2218 non-null   object
 7   ELEMENT LENGTH          2154 non-null   object
 8   POSITION                2218 non-null   object
 9   PICTURE                 1961 non-null   object
 10  REDEFINES               23 non-null     object
 11  VALUE                   66 non-null     object
 12  OCCURS                  31 non-null     object
 13  table_index             2218 non-null   int64
 14  table_vers              2218 non-null   int64
dtypes: int64(4), object(11)
memory usage: 277.2+ KB
```

[33]:
```
retval.head()
```

[33]:

	record	indent	data_level	element_name	raw_element_descriptors	data_step	USAGE	ELEMENT LENGTH	POSITION	PICTURE	REDEFINES	VALUE	OCCURS	table_index	table_vers
0	ACCT-DATA	2	05	LN-NBR	PICTURE IS 9(2)\n	100	DISPLAY	2	1	9(2)	NaN	NaN	NaN	0	1
1	ACCT-DATA	2	05	FD-CDE	USAGE IS DISPLAY\n	200	DISPLAY	4	3	NaN	NaN	NaN	NaN	0	1
2	ACCT-DATA	4	10	FD-CDE-3	ELEMENT LENGTH IS 4\n	300	DISPLAY	3	3	NaN	NaN	NaN	NaN	0	1
3	ACCT-DATA	6	15	FD-CDE-2	USAGE IS DISPLAY\n	400	DISPLAY	2	3	9(2)	NaN	NaN	NaN	0	1
4	ACCT-DATA	6	15	FD-CDE-3RD	PICTURE IS 9(1)\n	500	DISPLAY	1	5	9(1)	NaN	NaN	NaN	0	1

```
4              1        5    9(1)       NaN   NaN    NaN            0

   table_vers
0           1
1           1
2           1
3           1
4           1
```

[34]:
```
reformatted_retval = retval.rename(
    columns={
        'record': 'table_name',
        'element_name' : 'field_name',
        'USAGE' : 'end',
        'PICTURE' : 'data_type',
        'indent' : 'indent_space_count',
        'data_step' : 'declaration_step'
    } )
```

[35]:
```
reformatted_retval['BLANK ON'] = ''
reformatted_retval['INDEXED BY'] = ''
reformatted_retval['OLQ'] = ''
```

[36]:
```
reformatted_retval = reformatted_retval.drop(␣
 ↪columns=['raw_element_descriptors'] )
```

[37]:
```
reformatted_retval.to_csv( '2023-02-14_FSA_FARMS_schema_from_source.csv' )
```

[38]:
```
#pd.set_option( 'display.max_rows', 100 )
```

[39]:
```
# Two Python/PANDAS syntactical ways to select one whole table for inspection,␣
 ↪if you want
#retval.loc[ 'ACCT-DATA' ]
#retval.loc[ ('ACCT-DATA', slice(None)), : ]
```

[40]:
```
!head -30 2023-02-14_FSA_FARMS_schema_from_source.csv
```

```
,table_name,indent_space_count,data_level,field_name,declaration_step,end,ELEMEN
T LENGTH,POSITION,data_type,REDEFINES,VALUE,OCCURS,table_index,table_vers,BLANK
ON,INDEXED BY,OLQ
0,ACCT-DATA,2,05,LN-NBR,100,DISPLAY,2,1,9(2),,,,0,1,,,
1,ACCT-DATA,2,05,FD-CDE,200,DISPLAY,4,3,,,,,0,1,,,
2,ACCT-DATA,4,10,FD-CDE-3,300,DISPLAY,3,3,,,,,0,1,,,
3,ACCT-DATA,6,15,FD-CDE-2,400,DISPLAY,2,3,9(2),,,,0,1,,,
4,ACCT-DATA,6,15,FD-CDE-3RD,500,DISPLAY,1,5,9(1),,,,0,1,,,
5,ACCT-DATA,4,10,FD-CDE-4TH,600,DISPLAY,1,6,9(1),,,,0,1,,,
6,ACCT-DATA,2,05,KIND-CDE-LN,700,DISPLAY,2,7,9(2),,,,0,1,,,
7,ACCT-DATA,2,05,INT-RATE-NOTE,800,DISPLAY,6,9,9(2)V9(4),,,,0,1,,,
```

```
8,ACCT-DATA,2,05,INT-RATE-NOTE-1ST,900,DISPLAY,6,9,V9(6),INT-RATE-NOTE,,,0,1,,,
9,ACCT-DATA,2,05,PYMT-TYP-CDE,1000,DISPLAY,1,15,9(1),,,,0,1,,,
10,ACCT-DATA,2,05,DIR-PYMT-CDE,1100,DISPLAY,1,16,9(1),,,,0,1,,,
11,ACCT-DATA,2,05,DTE-AMORTN-EFCTV,1200,DISPLAY,6,17,9(06),,,,0,1,,,
12,ACCT-DATA,2,05,DSTR-DCLRD-CDE,1300,DISPLAY,5,23,,,,,0,1,,,
13,ACCT-DATA,4,10,DSTR-TYP-CDE,1400,DISPLAY,1,23,9(1),,,,0,1,,,
14,ACCT-DATA,4,10,FY-DSTR-DCLRD,1500,DISPLAY,1,24,9(1),,,,0,1,,,
15,ACCT-DATA,4,10,DSTR-DCLRD-NBR,1600,DISPLAY,3,25,9(3),,,,0,1,,,
16,ACCT-DATA,2,05,MRG-CNTRL,1700,DISPLAY,2,28,9(2),,,,0,1,,,
17,ACCT-DATA,2,05,DOCMT-TYP-CDE,1800,DISPLAY,1,30,X(1),,,,0,1,,,
18,ACCT-DATA,2,05,DTE-OBLGN-LN,1900,DISPLAY,6,31,9(06),,,,0,1,,,
19,ACCT-DATA,2,05,ASSTNC-TYP-CDE,2000,DISPLAY,3,37,9(3),,,,0,1,,,
20,ACCT-DATA,2,05,LN-AMT-OBLGN,2100,COMP-3,6,40,S9(8)V99,,,,0,1,,,
21,ACCT-DATA,2,05,BEGNG-FRMR-RNCHR-CDE,2200,DISPLAY,1,46,X(01),,,,0,1,,,
22,ACCT-DATA,2,05,COLLTL-CDE,2300,DISPLAY,1,47,9(1),,,,0,1,,,
23,ACCT-DATA,2,05,CPN-PROCG-DTE,2400,DISPLAY,6,48,9(6),,,,0,1,,,
24,ACCT-DATA,2,05,CASE-NBR-CHNG-CDE,2500,DISPLAY,1,54,9(01),,,,0,1,,,
25,ACCT-DATA,2,05,PYMT-ASSTNC-METH-CDE,2600,DISPLAY,1,55,9(1),,,,0,1,,,
26,ACCT-DATA,2,05,INT-RATE-PREV,2700,DISPLAY,6,56,9(2)V9(4),,,,0,1,,,
27,ACCT-DATA,2,05,INT-RATE-PREV-REDFND,2800,DISPLAY,6,56,V9(6),INT-RATE-
PREV,,,0,1,,,
28,ACCT-DATA,2,05,FILLER,2900,DISPLAY,19,62,X(0019),,,,0,1,,,
```