TECHNOTE – Using Lua in Optim for z/OS to run LOOKUP functions using optim.mask() call

Since the GA version of Optim for z/OS 11.3 did not include support for the LOOKUP Providers, the documentation explaining the syntax of those Providers was not included. They are being included here in this document. The LOOKUP Provider syntax has evolved into two basic forms since its inception, and both forms are shown here since both forms are supported. One form is before the DEST parameter was introduced, and the other form is when syntax includes the DEST parameter.

Here is the description on the syntax on using optim.mask to perform LOOKUP, HASH_LOOKUP and RANDOM_LOOKUP:

LOOKUP privacy providers

The LOOKUP privacy providers mask data by using values that are selected from a "lookup" table. The LOOKUP privacy provider and hash lookup privacy provider select values that are based on the source value. The random lookup privacy provider selects random values without regard to the source value.

Certain type of data such as names, addresses, etc., cannot be generated by using arithmetical logic that is used by many of the privacy providers. When this type of data needs to be masked, a similar set of replacement data, such as a lookup table, is required.

Lookup table data is normally stored as a set of rows in a table with a key column. The lookup table fields with replacement data must have names that match the source table fields that receive the replacement data. The lookup table field and the corresponding source table field must also have similar data types.

The providers can mask data in multiple fields. The providers can also keep NULL, SPACE and ZERO-Length values in selected source tables, instead of using the lookup value.

One restriction that currently exists is that column names in the LOOKUP table must be all uppercase characters, and the specification of these names in the SEARCH and FLDDEFx statements must be in uppercase as well.

Lookup enhancement to support DEST parameter

Starting with IBM InfoSphere Optim Version 11.3.0.4 and beyond, the presence of DEST parameter enables the DEST included behavior and its absence uses original behavior.

Original behavior:

1. Parameters SEARCH, REPLACE, PRExxx, FLDDEFn take lookup table field names.

LOOKUP: REPLACE, SEARCH, PRExxx

HASH_LOOKUP: REPLACE, PRExxx

RANDOM_LOOKUP: REPLACE, PRExxx

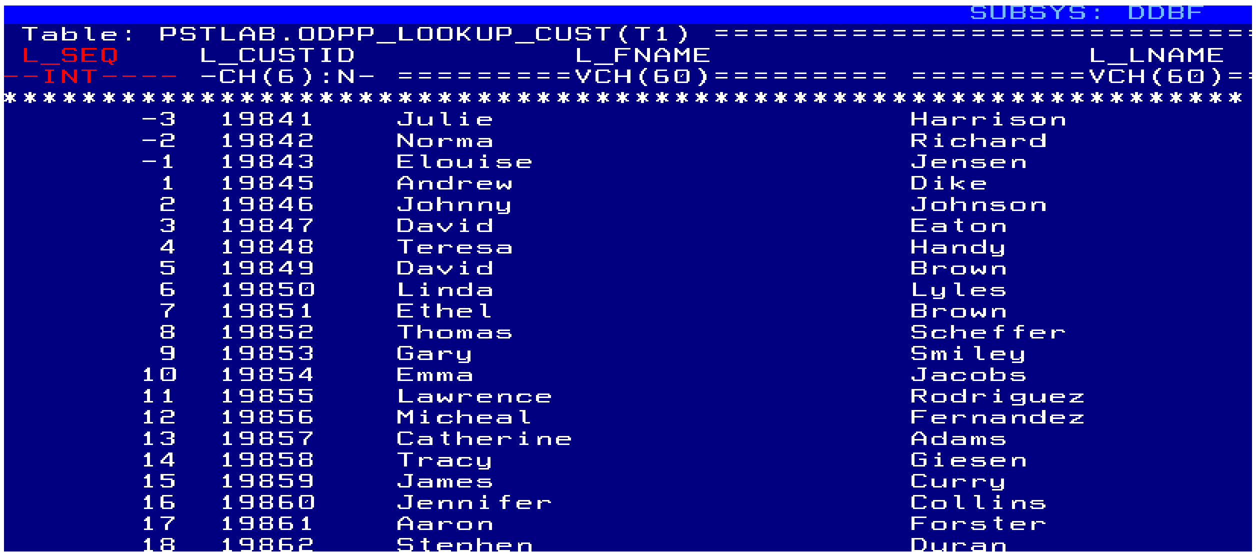
FLDDEFn are required for column names for both source and lookup tables.

DEST included behavior:

1. SOURCE, DEST, PRExxx and FLDDEFn take field (column) names from the source table.
2. SEARCH and REPLACE take field (column) names from the lookup table.
3. There is a 1-1 mapping between the field (column) names in DEST and REPLACE.
4. For plain Lookup, there is a 1-1 mapping between the field (column) names in SOURCE and SEARCH. SOURCE must be used along with DEST. If DEST is not specified SOURCE and SEARCH must not be specified together.
5. LUA optim.mask() call arguments count must match the number of FLDDEFn (in other words, this is the count of SOURCE and DEST fields(columns), irrespective of whether PRExxx is specified or not).

Hash Lookup

Lookup Table



```
                                                                      SUBSYS:  DDBF
 Table:  PSTLAB.ODPP_LOOKUP_CUST(T1) ===========================
 L_SEQ     L_CUSTID            L_FNAME                      L_LNAME
--INT----  -CH(6):N-  =========VCH(60)=========  =========VCH(60)==
****************************************************************
       -3   19841     Julie                      Harrison
       -2   19842     Norma                      Richard
       -1   19843     Elouise                    Jensen
        1   19845     Andrew                     Dike
        2   19846     Johnny                     Johnson
        3   19847     David                      Eaton
        4   19848     Teresa                     Handy
        5   19849     David                      Brown
        6   19850     Linda                      Lyles
        7   19851     Ethel                      Brown
        8   19852     Thomas                     Scheffer
        9   19853     Gary                       Smiley
       10   19854     Emma                       Jacobs
       11   19855     Lawrence                   Rodriguez
       12   19856     Micheal                    Fernandez
       13   19857     Catherine                  Adams
       14   19858     Tracy                      Giesen
       15   19859     James                      Curry
       16   19860     Jennifer                   Collins
       17   19861     Aaron                      Forster
       18   19862     Stephen                    Duran
```

Single source field, multiple destination fields

Table 1. Source

EMPNO	FIRSTNME	LASTNAME
000010	CHRISTINE	ADAMSON

Table 1. Source		
EMPNO	**FIRSTNME**	**LASTNAME**
000030	<null>	BOND
000040	SALLY	BROWN
000020	MICHAEL	ALONZO
000050	VERONICA	<6 spaces>

Masked with preserve

Original Syntax:
```
PRO=HASH_LOOKUP,hashfld="L_SEQ",source="EMPNO",
REPLACE="L_FNAME,L_LNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
PRENULL="L_FNAME",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=60),
FLDDEF3=(NAME="L_LNAME",DT=varchar_sz,len=60)
```
DEST included Syntax:
```
PRO=HASH_LOOKUP,hashfld="L_SEQ",SOURCE="EMPNO",
DEST="FIRSTNME,LASTNAME",REPLACE="L_FNAME,L_LNAME",
id="SCHEMA.ODPP_LOOKUP_CUST",PRENULL="FIRSTNME",
lib=DB2ZOSSQL, FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF3=(NAME="LASTNAME",DT=varchar_sz,len=15)
```
Lua:
```
x_empno = source.column.getvalue("EMPNO")
x_fname = source.column.getvalue("FIRSTNME")
x_lname = source.column.getvalue("LASTNAME")
mask_fname_value, mask_lname_value = optim.mask(x_empno, x_fname,
x_lname,<param_string>)
target.column.setvalue("FIRSTNME", mask_fname_value)
target.column.setvalue("LASTNAME", mask_lname_value)
```

Table 2. Masked with Preserve		
EMPNO	**FIRSTNME**	**LASTNAME**
000010	Edward	Guenther
000030	<null>	Mikels
000040	Kristen	Simpson
000050	Janet	Baade
000020	Wesley	Plummer

Masked without preserve

Original Syntax:

```
PRO=HASH_LOOKUP,hashfld="L_SEQ",source="EMPNO",
REPLACE="L_FNAME,L_LNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=60),
FLDDEF3=(NAME="L_LNAME",DT=varchar_sz,len=60)
```

DEST included Syntax:

```
PRO=HASH_LOOKUP,hashfld="L_SEQ",source="EMPNO",
dest="FIRSTNME,LASTNAME",replace="L_FNAME,L_LNAME",
id="SCHEMA.ODPP_LOOKUP_CUST",lib=DB2ZOSSQL,
FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF3=(NAME="LASTNAME",DT=varchar_sz,len=15)
```

Lua:

```
custid_val = source.column.getvalue(“EMPNO”)
firstname_val = source.column.getvalue(“FIRSTNME”)
lastname_val = source.column.getvalue(“LASTNAME”)
mask_fname_value, mask_lname_value = optim.mask(custid_val,firstname_val,
lastname_val,<param_string>)
target.column.setvalue(“FIRSTNME”, mask_fname_value)
target.column.setvalue(“LASTNAME”, mask_lname_value)
```

Table 3. Masked without preserve

EMPNO	FIRSTNME	LASTNAME
10005	Robert	Olson
10003	Rene	Dunn
10002	Michael	York
10004	Allen	Perl
10001	Joe	Jusino

Hash Source and destination field is the same (Example: FIRSTNME)

Masked without preserve

Original Syntax (2 separate FLDDEFn are required):

```
PRO=HASH_LOOKUP,HASHFLD="L_SEQ",SOURCE="FIRSTNME",
REPLACE="L_FNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=60)
```

DEST included Syntax:

```
PRO=HASH_LOOKUP,HASHFLD="L_SEQ",SOURCE="FIRSTNME",
DEST="FIRSTNME",REPLACE="L_FNAME",
id="SCHEMA.ODPP_LOOKUP_CUST",lib=DB2ZOSSQL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz,len=12)
```

Lua:

```
firstname_val = source.column.getvalue(“FIRSTNME”)
```

```
mask_fname_value = optim.mask(firstname_val, <param_string>)
target.column.setvalue(“FIRSTNME”, mask_fname_value)
```

Table 4. Source

FIRSTNME
Tim
<null>
Jenny
Sarah
Veronica

Table 5. Masked

FIRSTNME
Joe
r_null
Rene
Joe
Allen

Masked with preserve

Original Syntax (2 separate FLDDEFn are required):

```
PRO=HASH_LOOKUP,hashfld="L_SEQ",SOURCE="FIRSTNME",replace="L_FNAME",
prenull="L_FNAME",id="SCHEMA.ODPP_LOOKUP_CUST",lib=DB2ZOSSQL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=15)
```

DEST included Syntax:

```
PRO=HASH_LOOKUP,hashfld="L_SEQ",source="FIRSTNME",dest="FIRSTNME",
REPLACE="L_FNAME",prenull="FIRSTNME",id="SCHEMA.ODPP_LOOKUP_CUST",lib=DB2ZOSS
QL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz,len=12)
```

Lua:

```
firstname_val = source.column.getvalue(“FIRSTNME”)
mask_fname_value = optim.mask(firstname_val, <param_string>)
target.column.setvalue(“FIRSTNME”, mask_fname_value)
```

Table 6. Source

FIRSTNME
Tim
<null>
Jenny
Sarah

Table 6. Source

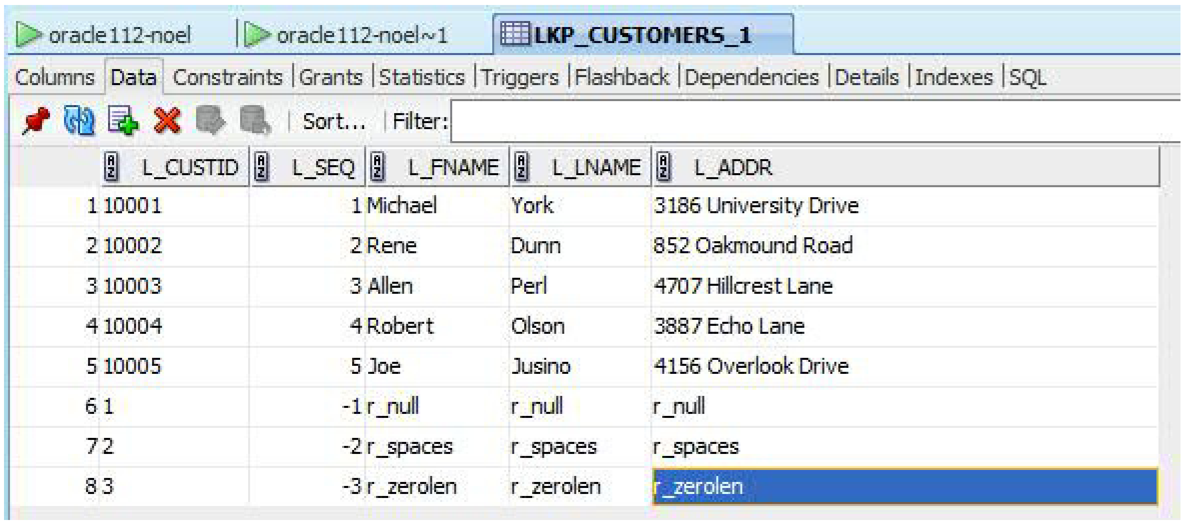
FIRSTNME
Veronica

Table 7. Masked

FIRSTNME
Joe
<null>
Rene
Joe
Allen

Plain Lookup

Lookup Table



	L_CUSTID	L_SEQ	L_FNAME	L_LNAME	L_ADDR
1	10001	1	Michael	York	3186 University Drive
2	10002	2	Rene	Dunn	852 Oakmound Road
3	10003	3	Allen	Perl	4707 Hillcrest Lane
4	10004	4	Robert	Olson	3887 Echo Lane
5	10005	5	Joe	Jusino	4156 Overlook Drive
6	1	-1	r_null	r_null	r_null
7	2	-2	r_spaces	r_spaces	r_spaces
8	3	-3	r_zerolen	r_zerolen	r_zerolen

Multiple search fields, single destination field

Source

Table 8. Source

EMPNO	FIRSTNME	LASTNAME
10005	Joe	Richards
10003	Allen	Bond
10002	Rene	<10 spaces >

Table 8. Source

EMPNO	FIRSTNME	LASTNAME
10004	Robert	Reeves
10001	Michael	Parker

Masked with preserve

Original Syntax:

```
PRO=LOOKUP,SEARCH="L_CUSTID,L_FNAME",REPLACE="L_LNAME",
OPERATOR=AND,PRESPACES="L_LNAME",
id="SCHEMA.ODPP_LOOKUP_CUST",lib=DB2ZOSSQL,
FLDDEF1=(NAME="L_CUSTID",DT=char,len=6),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=60),
FLDDEF3=(NAME="L_LNAME",DT=varchar_sz,len=60)
```

DEST included Syntax:

```
PRO=LOOKUP,SOURCE="EMPNO,FIRSTNME",SEARCH="L_CUSTID,L_FNAME",
DEST="LASTNAME",REPLACE="L_LNAME",
OPERATOR=AND,PRESPACES="LASTNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF3=(NAME="LASTNAME",DT=varchar_sz,len=15)
```

Lua:

```
custid_val = source.column.getvalue(“EMPNO”)
firstname_val = source.column.getvalue(“FIRSTNME”)
lastname_val = source.column.getvalue(“LASTNAME”)
mask_lname_value = optim.mask(custid_val,firstname_val,
lastname_val,<param_string>)
target.column.setvalue(“LASTNAME”, mask_lname_value)
```

Table 9. Masked with Preserve

EMPNO	FIRSTNME	LASTNAME
10005	Joe	Jusino
10003	Allen	Perl
10002	Rene	<10 spaces>
10004	Robert	Olson
10001	Michael	York

Masked without preserve

Original Syntax:

```
PRO=LOOKUP,SEARCH="L_CUSTID,L_FNAME",REPLACE="L_LNAME",
OPERATOR=AND, id="SCHEMA.ODPP_LOOKUP_CUST",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="L_CUSTID",DT=char,len=6),
FLDDEF2=(NAME="L_FNAME",DT=varchar_sz,len=60),
FLDDEF3=(NAME="L_LNAME",DT=varchar_sz,len=60)
```

DEST included Syntax:

```
PRO=LOOKUP,SOURCE="EMPNO,FIRSTNME",SEARCH="L_CUSTID,L_FNAME",
DEST="LASTNAME",
REPLACE="L_LNAME",OPERATOR=AND,id="SCHEMA.ODPP_LOOKUP_CUST",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="EMPNO",DT=char,len=6),
FLDDEF2=(NAME="FIRSTNME",DT=varchar_sz,len=12),
FLDDEF3=(NAME="LASTNAME",DT=varchar_sz,len=15)
```

Lua:

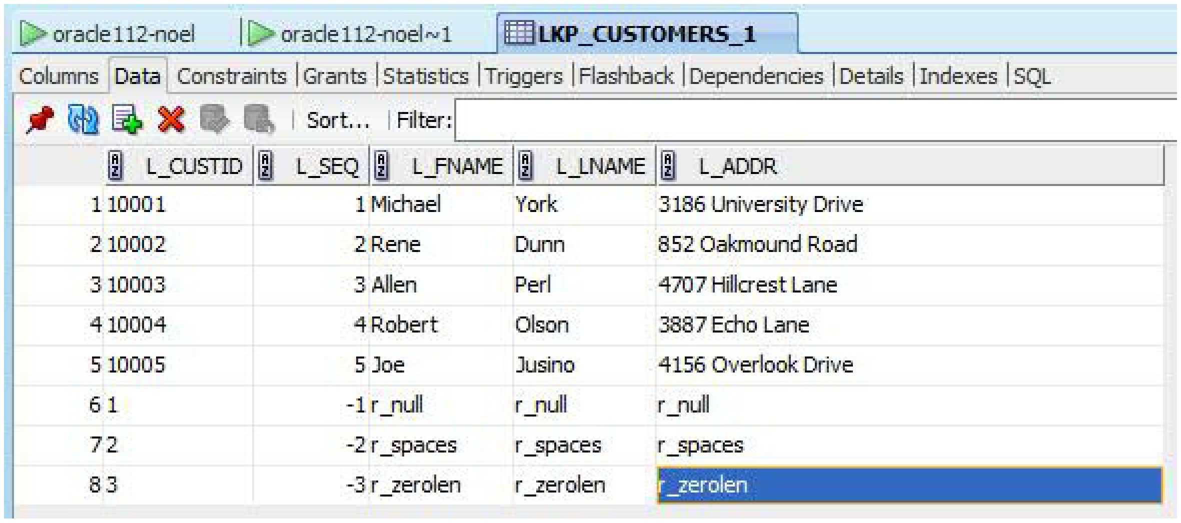
```
custid_val = source.column.getvalue("EMPNO")
firstname_val = source.column.getvalue("FIRSTNME")
lastname_val = source.column.getvalue("LASTNAME")
mask_lname_value = optim.mask(custid_val,firstname_val,
lastname_val,<param_string>)
target.column.setvalue("LASTNAME", mask_lname_value)
```

Table 10. Masked without Preserve

EMPNO	FIRSTNME	LASTNAME
1005	Joe	Jusino
1003	Allen	Perl
1002	Rene	Dunn
1004	Robert	Olson
10001	Michael	York

Random Lookup

Lookup Table



	L_CUSTID	L_SEQ	L_FNAME	L_LNAME	L_ADDR
1	10001	1	Michael	York	3186 University Drive
2	10002	2	Rene	Dunn	852 Oakmound Road
3	10003	3	Allen	Perl	4707 Hillcrest Lane
4	10004	4	Robert	Olson	3887 Echo Lane
5	10005	5	Joe	Jusino	4156 Overlook Drive
6	1	-1	r_null	r_null	r_null
7	2	-2	r_spaces	r_spaces	r_spaces
8	3	-3	r_zerolen	r_zerolen	r_zerolen

Multiple destination fields

Source

Table 11. Source

FIRSTNME	LASTNAME
Tim	Richards
<null>	Bond
Jenny	<10 spaces>
Sarah	Reeves
Veronica	Parker

Original Syntax:

```
PRO=RANDOM_LOOKUP,REPLACE="L_FNAME,
L_LNAME",id="SCHEMA.ODPP_LOOKUP_CUST",PRENULL="L_FNAME",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="L_FNAME",DT=varchar_sz,len=60),
FLDDEF2=(NAME="L_LNAME",DT=varchar_sz,len=60)
```

DEST included Syntax:

```
PRO=RANDOM_LOOKUP,DEST="FIRSTNME,LASTNAME",
REPLACE="L_FNAME,L_LNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
PRENULL="FIRSTNME",lib=DB2ZOSSQL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz),
FLDDEF2=(NAME="LASTNAME",DT=varchar_sz)
```

Lua:

```
firstname_val = source.column.getvalue(“FIRSTNME”)
lastname_val = source.column.getvalue(“LASTNAME”)
mask_fname_value, mask_lname_value = optim.mask(firstname_val,
lastname_val,<param_string>)
target.column.setvalue(“FIRSTNME”, mask_fname_value)
target.column.setvalue(“LASTNAME”, mask_lname_value)
```

The output changes on each run:

Source:

Table 12. Source

FIRSTNME	LASTNAME
Tim	Richards
<null>	Bond
Jenny	<10 Spaces>
Sarah	Reeves
Veronica	Parker

Table 13. Masked

FIRSTNME	LASTNAME
Joe	Jusino
<null>	York
Rene	Dunn
r_zerolen	r_zerolen
Allen	Perl

Masked without preserve

Original Syntax:
```
PRO=RANDOM_LOOKUP,REPLACE="L_FNAME,L_LNAME",
id="SCHEMA.ODPP_LOOKUP_CUST",PRENULL="L_FNAME",
lib=DB2ZOSSQL,
FLDDEF1=(NAME="L_FNAME",DT=varchar_sz),
FLDDEF2=(NAME="L_LNAME",DT=varchar_sz)
```
DEST included Syntax:
```
PRO=RANDOM_LOOKUP,DEST="FIRSTNME,LASTNAME",
REPLACE="L_FNAME,L_LNAME",id="SCHEMA.ODPP_LOOKUP_CUST",
PRENULL="FIRSTNME",lib=DB2ZOSSQL,
FLDDEF1=(NAME="FIRSTNME",DT=varchar_sz),
FLDDEF2=(NAME="LASTNAME",DT=varchar_sz)
```

Lua:

```
firstname_val = source.column.getvalue("FIRSTNME")
laststname_val = source.column.getvalue("LASTNAME")
mask_fname_value, mask_lname_value =
optim.mask(firstname_val,lastname_val,<param_string>)
target.column.setvalue("FIRSTNME", mask_fname_value)
target.column.setvalue("LASTSTNAME", mask_lname_value)
```

The output changes on each run:

Table 14. Source

FIRSTNME	LASTNAME
Tim	Richards
<null>	Bond
Jenny	<10 Spaces>
Sarah	Reeves
Veronica	Parker

Table 15. Masked

FIRSTNME	LASTNAME
Michael	York
Rene	Dunn
Robert	Olson
Rene	Dunn
r_null	r_null

Lua proc example

When ODPP is used in Lua, PRESERVE options are easily implemented by Lua script even with the original syntax. See the following sample:

```
function is_spaces(str)
    str1 = string.gsub(str, " ", "")
    if #str1 == 0 then
        return true
    else
        return false
    end
end

function cm_transform()
    -- optim.mask() returns mask values without PRExxxx options.
    value1 = source.column.getvalue("COL_CH1")
    mask2, mask3 = optim.mask(value1,
           'PRO=HASH_LOOKUP, FLDDEF1=(NAME="COL_CH1",DT=WCHAR),
                                      SRC="COL_CH1",
                                      HASHFLD="SEQ",
                                      REPLACE="COL_CH2, COL_VC3",
                                      ...')
    -- If COL_CH2 is NULL or SPACES, copy it to mask2.
    if value2 == nil then
        mask2 = value2
    elseif is_spaces(value2) == true then
        mask2 = value2
    end

    -- If COL_VC3 is NULL or ZEROLENGTH, copy it to mask3.
    if value3 == nil then
        mask3 = value3
    elseif #value3 == 0 then
        mask3 = value3
    end

    target.column.setvalue("COL_CH2", mask2)
    target.column.setvalue("COL_CH3", mask3)
end
```

- Lookup privacy provider
  Use the lookup privacy provider to obtain replacement values from one or more lookup table fields, according to the value in one or more source table fields.
- Hash lookup privacy provider
  Use the hash lookup privacy provider to obtain the value for a destination field from a lookup table, according to a hashed value derived from a source field.
- Random lookup privacy provider
  Use the random lookup privacy provider to select values at random from a lookup table and insert them into a destination field. The provider generates a random number between 1 and the limit or number of rows in the lookup table to use as a subscript into the table. The field value or values from the row that correspond to the subscript are inserted in the destination field. The value that is selected from the lookup table is not based on the source value.

Preparing Optim for z/OS to run optim.mask() in Lua using LOOKUP Providers

It will be necessary to perform a DB2 BIND to take advantage of the new function this change includes. This will add the capability to perform optim.mask function within Lua procedure that invokes LOOKUP Providers.
It is a two step process. First a bind of the packages with two new members that will allow Optim Data Privacy LOOKUP functions to work because it will use the same interface that Optim for z/OS uses, which is embedded SQL. The second step is adding this collection to a BIND of the PLAN being used to invoke Optim for z/OS.

A new SFOPSAMP member IOQBIND will be available that has the sample JCL needed to perform the bind, but it is duplicated here as well. Please substitute the appropriate values for plan name, subsystem name, QUAL, etc.:

```
//BINDLKP JOB (),CLASS=A,MSGCLASS=H,
//   MSGLEVEL=(1,1),NOTIFY=PSTLAB,REGION=0M
//*****************************************************************
//*       BIND for LOOKUP support in Optim for z/OS
//*         Please note that Package name is added to BIND of PLAN.
//*         Change OWNER and QUALIFIER as needed.
//*         Adjust PLAN name to match client installation.
//*******************************************************************
//BIND     EXEC PGM=IKJEFT01,DYNAMNBR=100
//STEPLIB  DD DSN=DSN.<db2-ver>.<subsys>.RUNLIB.LOAD,DISP=SHR <===
//        DD DSN=DSN.<db2-ver>.<subsys>.SDSNEXIT,DISP=SHR <===
//        DD DSN=DSN.<db2-ver>.SDSNLOAD,DISP=SHR  <===
//DBRMLIB  DD DISP=SHR,DSN=<hlq>.SIOQSAMP              <===
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(<subsys>)             /* VERIFY DB2 SYSTEM <===     */

  BIND PACKAGE (IOQPKGB30LKP) MEMBER(IOQLKO) OWNER(OPZB30) -
     QUALIFIER(OPZB30) LIBRARY('PSTRAND.ODPP.DBRM')  -
     VALIDATE(BIND) ISOLATION(CS)  -
     FLAG(I) RELEASE(COMMIT) EXPLAIN(NO) CURRENTDATA(YES)  -
     ENCODING(EBCDIC)

  BIND PACKAGE (IOQPKGB30LKP) MEMBER(IOQLKB) OWNER(OPZB30) -
     QUALIFIER(OPZB30) LIBRARY('PSTRAND.ODPP.DBRM')  -
     VALIDATE(BIND) ISOLATION(CS)  -
     FLAG(I) RELEASE(COMMIT) EXPLAIN(NO) CURRENTDATA(YES)  -
     ENCODING(EBCDIC)

  BIND PLAN (FOP1130) OWNER(OPZB30) QUALIFIER(OPZB30)  -
     ACTION(REPLACE) -
     PKLIST(FOPPACKAGEMB.*,FOPPACKAGEME.* IOQPKGB30LKP.*)  -
     CURRENTDATA(YES) VALIDATE(BIND) ISOLATION(CS) FLAG(I)  -
     ACQUIRE(USE) RELEASE(COMMIT) EXPLAIN(NO) ENCODING(EBCDIC)  -
     SQLRULES(STD)  -
     ;

 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA11) /* <=== VERIFY PLAN NAME */
END
//SYSIN   DD  *  FOR DSNTIAD
  GRANT EXECUTE ON PACKAGE IOQPKGB30LKP.* TO PUBLIC;
  GRANT EXECUTE ON PLAN    FOP1130        TO PUBLIC;
```