-- BEGIN PART 5A ORACLE

SET DEFINE OFF;

SET SERVEROUTPUT ON SIZE UNLIMITED;

ALTER SESSION SET NLS\_COMP=LINGUISTIC;

ALTER SESSION SET NLS\_SORT=BINARY\_CI;

ALTER SYSTEM SET OPEN\_CURSORS=3000 SCOPE=MEMORY;

/\*

Add guids to empty guid PK columns.

Merge PK GUIDs into FK columns.

Drop old keys.

Add guid keys.

\*/

BEGIN

DBMS\_OUTPUT.ENABLE(buffer\_size => NULL);

END;

/

-----------------------------------------------------------------------------------------------------------------------------------------

-- ---------------------------------------------------------- ADD GUIDS. -----------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------------

-- WIP Set changes into default data. Leave this code to update custom user tables.

UPDATE DATADICT DD SET

NULL\_ALLOW = 'Y',

UNIQUEKEY = 'N',

KEYATTR1 = 'N'

WHERE EXISTS (

SELECT \*

FROM PON\_DICT D

JOIN PON\_TABLE T ON T.TABLE\_NAME = D.TABLE\_NAME

WHERE T.RANK <> 99

AND D.TABLE\_NAME = UPPER(DD.TABLE\_NAME)

AND D.COL\_NAME = UPPER(DD.COL\_NAME)

AND D.PK = 1

OR EXISTS(SELECT \* FROM PON\_FK WHERE TABLE\_NAME = DD.TABLE\_NAME AND COL\_NAME = DD.COL\_NAME AND PURPOSE = 1)

);

-- Update non join (1 column in the pk) guid PK columns.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Update non join (1 column in the pk) guid PK columns.');

OPEN V\_CUR FOR

SELECT 'UPDATE ' || D.TABLE\_NAME || ' SET ' || D.COL\_NAME || ' = SYS\_GUID() WHERE ' || D.COL\_NAME || ' IS NULL'

FROM (SELECT TABLE\_NAME, COL\_NAME

FROM PON\_DICT

WHERE PK = 2

MINUS

SELECT TABLE\_NAME, COL\_NAME

FROM PON\_FK

WHERE PURPOSE = 2

) D;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Update guid FK columns (including PK columns in join tables).

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Update guid FK columns (including PK columns in join tables).');

OPEN V\_CUR FOR

SELECT

'MERGE INTO ' || F.TABLE\_NAME || ' T ' ||

'USING ' || F.R\_TABLE\_NAME || ' R ' ||

'ON (' || PON\_CONCAT(CURSOR(SELECT 'T.' || COL\_NAME || ' = R.' || R\_COL\_NAME

FROM PON\_FK

WHERE FK\_NAME = F.FK\_NAME AND PURPOSE = 1),

' AND ') ||

') ' ||

'WHEN MATCHED THEN ' ||

'UPDATE SET T.' || F.COL\_NAME || ' = R.' || F.R\_COL\_NAME

FROM (

SELECT FK\_NAME, TABLE\_NAME, COL\_NAME, R\_TABLE\_NAME, R\_COL\_NAME

FROM PON\_FK TF

WHERE PURPOSE = 2

AND EXISTS(SELECT 1 FROM PON\_FK WHERE FK\_NAME = TF.FK\_NAME AND PURPOSE = 1)

) F

JOIN PON\_TABLE T ON T.TABLE\_NAME = F.TABLE\_NAME

ORDER BY T.RANK;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Drop FKs to or from PON\_ELEM\_INSP

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Drop FKs to or from PON\_ELEM\_INSP');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || TABLE\_NAME || ' DROP CONSTRAINT ' || UPPER(CONSTRAINT\_NAME)

FROM USER\_CONSTRAINTS

WHERE R\_CONSTRAINT\_NAME IN (SELECT CONSTRAINT\_NAME FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'PON\_ELEM\_INSP')

OR TABLE\_NAME = 'PON\_ELEM\_INSP';

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Import PON\_ELEM\_INSP into PON\_ELEM\_INVENTORY and PON\_ELEM\_INVENTORY\_INSP.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Import PON\_ELEM\_INSP into PON\_ELEM\_INVENTORY and PON\_ELEM\_INVENTORY\_INSP.');

MERGE INTO PON\_ELEM\_INSP T

USING PON\_ELEM\_INSP S

ON (S.BRKEY = T.BRKEY AND

S.INSPKEY = T.INSPKEY AND

S.ELEM\_KEY = T.ELEM\_PARENT\_KEY AND

S.ENVKEY = T.ENVKEY AND

S.STRUNITKEY = T.STRUNITKEY)

WHEN MATCHED THEN UPDATE SET T.PARENT\_PON\_ELEM\_INSP\_GD = S.PON\_ELEM\_INSP\_GD;

-- Import only the PON\_ELEM\_INSP rows from the latest inspection.

-- Note PON\_ELEM\_INSP\_GD is imported into PON\_ELEM\_INVENTORY\_GD.

INSERT INTO PON\_ELEM\_INVENTORY(PON\_ELEM\_INVENTORY\_GD, PARENT\_PON\_ELEM\_INVENTORY\_GD, STRUCTURE\_UNIT\_GD, PON\_ENVT\_DEFS\_GD, PON\_ELEM\_DEFS\_GD, INVT\_DESC, TOTAL\_QUANTITY, SCALE\_FACTOR, CREATEDATETIME, CREATEUSERKEY, MODTIME, MODUSERKEY)

SELECT EI.PON\_ELEM\_INSP\_GD, EI.PARENT\_PON\_ELEM\_INSP\_GD, EI.STRUCTURE\_UNIT\_GD, EI.PON\_ENVT\_DEFS\_GD, EI.PON\_ELEM\_DEFS\_GD, EI.ELEM\_DESC, EI.ELEM\_QUANTITY, EI.ELEM\_SCALE\_FACTOR, EI.ELEM\_CREATEDATETIME, EI.ELEM\_CREATEUSERKEY, EI.ELEM\_MODTIME, EI.ELEM\_MODUSERKEY

FROM PON\_ELEM\_INSP EI

WHERE EI.INSPKEY IN (SELECT INSPKEY

FROM INSPEVNT INSP

JOIN (SELECT BRKEY, MAX(INSPDATE) INSPDATE

FROM INSPEVNT

GROUP BY BRKEY) LATEST\_INSPECTION ON LATEST\_INSPECTION.BRKEY = INSP.BRKEY

AND LATEST\_INSPECTION.INSPDATE = INSP.INSPDATE);

INSERT INTO PON\_ELEM\_INVENTORY\_INSP(INSPEVNT\_GD, PON\_ELEM\_INVENTORY\_GD, QTYSTATE1, QTYSTATE2, QTYSTATE3, QTYSTATE4, INVT\_INSP\_NOTES, CREATEDATETIME, CREATEUSERKEY, MODTIME, MODUSERKEY)

SELECT EI.INSPEVNT\_GD, EI.PON\_ELEM\_INSP\_GD, EI.ELEM\_QTYSTATE1, EI.ELEM\_QTYSTATE2, EI.ELEM\_QTYSTATE3, EI.ELEM\_QTYSTATE4, EI.ELEM\_NOTES, EI.ELEM\_CREATEDATETIME, EI.ELEM\_CREATEUSERKEY, EI.ELEM\_MODTIME, EI.ELEM\_MODUSERKEY

FROM PON\_ELEM\_INSP EI

WHERE EI.INSPKEY IN (SELECT INSPKEY

FROM INSPEVNT INSP

JOIN (SELECT BRKEY, MAX(INSPDATE) INSPDATE

FROM INSPEVNT

GROUP BY BRKEY) LATEST\_INSPECTION ON LATEST\_INSPECTION.BRKEY = INSP.BRKEY

AND LATEST\_INSPECTION.INSPDATE = INSP.INSPDATE);

END IF;

END;

/

-- Drop old FKs.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Drop old FKs.');

OPEN V\_CUR FOR

SELECT DISTINCT 'ALTER TABLE ' || T.TABLE\_NAME || ' DROP CONSTRAINT ' || T.CONSTRAINT\_NAME

FROM (

SELECT UC.TABLE\_NAME, UC.CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS UC

JOIN USER\_CONS\_COLUMNS UCC ON UCC.CONSTRAINT\_NAME = UC.CONSTRAINT\_NAME

JOIN PON\_FK FK ON FK.PURPOSE = 1 AND FK.TABLE\_NAME = UC.TABLE\_NAME AND FK.COL\_NAME = UCC.COLUMN\_NAME

WHERE UC.CONSTRAINT\_TYPE = 'R'

-- Include standard user tables for those users that have custom user tables.

UNION

SELECT TABLE\_NAME, CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS

WHERE CONSTRAINT\_TYPE = 'R' AND

TABLE\_NAME IN ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT')

) T

ORDER BY 1;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Drop old PKs.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Drop old PKs.');

OPEN V\_CUR FOR

SELECT DISTINCT 'ALTER TABLE ' || UC.TABLE\_NAME || ' DROP CONSTRAINT ' || UC.CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS UC

JOIN USER\_CONS\_COLUMNS UCC ON UCC.CONSTRAINT\_NAME = UC.CONSTRAINT\_NAME

JOIN PON\_DICT D ON D.TABLE\_NAME = UC.TABLE\_NAME AND D.COL\_NAME = UCC.COLUMN\_NAME

JOIN PON\_TABLE T ON T.TABLE\_NAME = D.TABLE\_NAME

WHERE UC.CONSTRAINT\_TYPE = 'P' AND D.PK = 1 AND T.RANK <> 99

ORDER BY 1;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Make old key columns nullable.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Make old key columns nullable.');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || D.TABLE\_NAME || ' MODIFY(' || D.COL\_NAME || ' NULL)'

FROM PON\_TABLE T

JOIN PON\_DICT D ON D.TABLE\_NAME = T.TABLE\_NAME

JOIN USER\_TAB\_COLS TC ON TC.TABLE\_NAME = D.TABLE\_NAME AND TC.COLUMN\_NAME = D.COL\_NAME

WHERE T.RANK <> 99

AND TC.NULLABLE = 'N'

AND (

D.PK = 1 OR EXISTS(SELECT \* FROM PON\_FK WHERE TABLE\_NAME = D.TABLE\_NAME AND COL\_NAME = D.COL\_NAME AND PURPOSE = 1) OR D.COL\_NAME LIKE '%CAPTION\_ID'

);

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Make PK GUID columns required.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Make PK GUID columns required.');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || D.TABLE\_NAME || ' MODIFY ' || D.COL\_NAME || ' ' || D.DATA\_TYPE ||

CASE WHEN D.LENGTH IS NOT NULL AND D.SCALE IS NOT NULL THEN '(' || CAST(D.LENGTH AS VARCHAR(30)) || ', ' || CAST(D.SCALE AS VARCHAR(30)) || ')'

WHEN D.LENGTH IS NOT NULL AND D.SCALE IS NULL THEN '(' || CAST(D.LENGTH AS VARCHAR(30))|| ')'

ELSE ''

END || ' NOT NULL'

FROM PON\_DICT D

JOIN USER\_TAB\_COLUMNS C ON C.TABLE\_NAME = D.TABLE\_NAME AND C.COLUMN\_NAME = D.COL\_NAME

WHERE D.PK > 1 AND C.NULLABLE = 'Y';

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Add replacement PKs (PON\_DICT.PK = 2).

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Add replacement PKs (PON\_DICT.PK = 2).');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || T.TABLE\_NAME || ' ADD CONSTRAINT ' || SUBSTR(T.TABLE\_NAME, 1, 27) || '\_PK PRIMARY KEY (' ||

PON\_CONCAT(

CURSOR(SELECT COL\_NAME FROM PON\_DICT WHERE TABLE\_NAME = T.TABLE\_NAME AND PK = 2)

) ||

')'

FROM (

SELECT DISTINCT TABLE\_NAME

FROM PON\_DICT

WHERE PK = 2

) T

ORDER BY 1;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Add replacement FKs (PON\_FK.PURPOSE > 1).

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Add replacement FKs (PON\_FK.PURPOSE = 2)');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || F.TABLE\_NAME || ' ADD CONSTRAINT ' || F.FK\_NAME ||

' FOREIGN KEY (' || PON\_CONCAT(CURSOR(SELECT COL\_NAME FROM PON\_FK WHERE FK\_NAME = F.FK\_NAME AND PURPOSE > 1)) || ') REFERENCES ' || F.R\_TABLE\_NAME || ' (' || PON\_CONCAT(CURSOR(SELECT R\_COL\_NAME FROM PON\_FK WHERE FK\_NAME = F.FK\_NAME AND PURPOSE > 1)) || ') ' || CASE F.DELETE\_RULE WHEN 1 THEN ' ON DELETE CASCADE' WHEN 2 THEN ' ON DELETE SET NULL' ELSE '' END

FROM (SELECT DISTINCT FK\_NAME, TABLE\_NAME, R\_TABLE\_NAME, DELETE\_RULE

FROM PON\_FK

WHERE PURPOSE = 2

) F

ORDER BY 1;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- Update guid non key columns.

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Update guid non key columns using old keys');

-- Rank 20 tables

UPDATE INSPEVNT

SET INSPECTOR\_GD = (SELECT PON\_APP\_USERS\_GD FROM PON\_APP\_USERS WHERE USERKEY = INSPEVNT.INSPUSRKEY)

WHERE IS\_INTEGER(INSPUSRKEY) = 1

AND INSPECTOR\_GD IS NULL;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

-- WIP Replace with foreign keys and automated merges.

MERGE INTO PON\_UTIL\_CRIT\_CATEGORY T

USING PON\_UTIL\_CRIT\_CATEGORY S

ON (T.CATKEY\_PARENT = S.CATKEY)

WHEN MATCHED THEN UPDATE SET T.PARENT\_UTIL\_CRIT\_CATEGORY\_GD = S.PON\_UTIL\_CRIT\_CATEGORY\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_UTIL\_CRIT\_CATEGORY T

USING PON\_DEFS\_ASSESSMENT S

ON (T.ASMTDEFKEY = S.ASMTDEFKEY)

WHEN MATCHED THEN UPDATE SET T.PON\_DEFS\_ASSESSMENT\_GD = S.PON\_DEFS\_ASSESSMENT\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_UTIL\_CRIT\_CATEGORY T

USING PON\_ELEM\_DEFS S

ON (T.ELEMKEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_UTIL\_CRIT\_CATEGORY T

USING PON\_FORMULAS S

ON (T.FORMULA\_ID = S.FORMULA\_ID)

WHEN MATCHED THEN UPDATE SET T.PON\_FORMULAS\_GD = S.PON\_FORMULAS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_FORMULA\_PARAMS T

USING PON\_FORMULAS S

ON (T.FORMULA\_ID = S.FORMULA\_ID)

WHEN MATCHED THEN UPDATE SET T.PON\_FORMULAS\_GD = S.PON\_FORMULAS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_FORMULA\_CATEGORIES T

USING PON\_FORMULA\_CATEGORIES S

ON (T.PARENTCATEGORYID = S.CATEGORYID)

WHEN MATCHED THEN UPDATE SET T.PARENT\_FORMULA\_CATEGORIES\_GD = S.PON\_FORMULA\_CATEGORIES\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_BENEFIT\_ELEMENTS T

USING PON\_ELEM\_DEFS S

ON (T.ELEM\_KEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_BENEFIT\_ELEMENTS T

USING PON\_ELEM\_DEFS S

ON (T.ELEM\_PARENT\_KEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.PARENT\_PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_BENEFIT\_ELEMENTS T

USING PON\_ELEM\_DEFS S

ON (T.ELEM\_GRANDPARENT\_KEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.GPARENT\_PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

-- Fill random GUIDs during the 1st run and on the merge - replace random with existing data from BRIDGE table

UPDATE CICOXCPT SET BRIDGE\_GD = SYS\_GUID();

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO CICOXCPT T

USING BRIDGE S

ON (T.BRKEY = S.BRKEY)

WHEN MATCHED THEN UPDATE SET T.BRIDGE\_GD = S.BRIDGE\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_FLEX\_ELEMS T

USING PON\_FLEXACTIONS\_SETS S

ON (T.FLEXKEY = S.FLEX\_ACTION\_KEY)

WHEN MATCHED THEN UPDATE SET T.PON\_FLEXACTIONS\_SETS\_GD = S.PON\_FLEXACTIONS\_SETS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_FLEX\_ELEMS T

USING PON\_ELEM\_DEFS S

ON (T.ELEMKEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_ELEM\_DEFS T

USING METRIC\_ENGLISH S

ON (T.ELEM\_PAIRCODE = S.PAIRCODE)

WHEN MATCHED THEN UPDATE SET T.METRIC\_ENGLISH\_GD = S.METRIC\_ENGLISH\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_ELEM\_DEFS T

USING PON\_ELEM\_CAT\_DEFS S

ON (T.ELEM\_CAT\_KEY = S.ECATKEY)

WHEN MATCHED THEN UPDATE SET T.PON\_ELEM\_CAT\_DEFS\_GD = S.PON\_ELEM\_CAT\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_ELEM\_DEFS T

USING PON\_ELEM\_DEFS S

ON (T.ELEM\_SUBSET\_KEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.SUBSET\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_MRR\_ACTION\_DEFS T

USING PON\_ELEM\_DEFS S

ON (T.ELEM\_KEY = S.ELEM\_KEY)

WHEN MATCHED THEN UPDATE SET T.PON\_ELEM\_DEFS\_GD = S.PON\_ELEM\_DEFS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_CONTEXT T

USING PON\_FILTERS S

ON (T.DEFAULT\_FILTERKEY = S.FILTERKEY)

WHEN MATCHED THEN UPDATE SET T.DEFAULT\_PON\_FILTERS\_GD = S.PON\_FILTERS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_CONTEXT T

USING PON\_LAYOUTS S

ON (T.DEFAULT\_LAYOUTKEY = S.LAYOUTKEY)

WHEN MATCHED THEN UPDATE SET T.DEFAULT\_PON\_LAYOUTS\_GD = S.PON\_LAYOUTS\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_FORMULA\_CATEGORIES T

USING PON\_FORMULA\_CATEGORIES S

ON (T.PARENTCATEGORYID = S.CATEGORYID)

WHEN MATCHED THEN UPDATE SET T.PARENT\_FORMULA\_CATEGORIES\_GD = S.PON\_FORMULA\_CATEGORIES\_GD;

END IF;

END;

/

-- Set Bridge\_GD so export code does not need to be fixed.

-- Not replaceable with automated merges using PON\_FK because bridge\_gd is not the primary key.

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_ELEM\_INSP T

USING INSPEVNT S

ON (T.INSPEVNT\_GD = S.INSPEVNT\_GD)

WHEN MATCHED THEN UPDATE SET T.BRIDGE\_GD = S.BRIDGE\_GD;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_INSP\_WORKCAND T

USING INSPEVNT S

ON (T.INSPEVNT\_GD = S.INSPEVNT\_GD)

WHEN MATCHED THEN UPDATE SET T.BRIDGE\_GD = S.BRIDGE\_GD;

END IF;

END;

/

-- Update Bridge\_gd in user tables.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

OPEN V\_CUR FOR

SELECT 'MERGE INTO ' || TABLE\_NAME || ' T

USING ' || CORE\_TABLE\_NAME || ' S

ON (T.' || CORE\_TABLE\_NAME || '\_GD = S.' || CORE\_TABLE\_NAME || '\_GD)

WHEN MATCHED THEN UPDATE SET T.BRIDGE\_GD = S.BRIDGE\_GD' -- Not Null

FROM PON\_TABLE

WHERE ORIGINAL\_TABLE\_NAME IN ('USERINSP', 'USERRWAY', 'USERSTRUNIT');

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

MERGE INTO PON\_GRPS\_ROAD T

USING ROADWAY S

ON (T.ROADWAY\_GD = S.ROADWAY\_GD)

WHEN MATCHED THEN UPDATE SET T.BRIDGE\_GD = S.BRIDGE\_GD; -- Not Null

END IF;

END;

/

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

-- Hard-coded merge of default data because PKs are 2s but the table is new.

MERGE INTO PON\_NAV\_TASK\_KEY T

USING PON\_NAV\_TASK\_KEY\_T S

ON (T.PON\_NAV\_TASK\_GD = S.PON\_NAV\_TASK\_GD)

WHEN MATCHED THEN UPDATE SET T.TABLE\_NAME = S.TABLE\_NAME, T.FIELD\_NAME = S.FIELD\_NAME, T.PONTIS\_STANDARD\_IND = S.PONTIS\_STANDARD\_IND;

MERGE INTO PON\_NAV\_TASK\_KEY T

USING PON\_NAV\_TASK\_KEY\_T S

ON (T.PON\_NAV\_TASK\_GD = S.PON\_NAV\_TASK\_GD)

WHEN NOT MATCHED THEN INSERT(PON\_NAV\_TASK\_GD, TABLE\_NAME, FIELD\_NAME, PONTIS\_STANDARD\_IND) VALUES (S.PON\_NAV\_TASK\_GD, S.TABLE\_NAME, S.FIELD\_NAME, S.PONTIS\_STANDARD\_IND);

END IF;

END;

/

-- Update LB columns.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

IF GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Update LB columns.');

OPEN V\_CUR FOR

SELECT

'UPDATE ' || T.TABLE\_NAME || ' SET ' || T.COL\_NAME || ' = ' || PON\_CONCAT(

CURSOR(SELECT COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME AND PK = 1

ORDER BY COL\_ORDER),

' || ''|'' || '

)

FROM PON\_DICT T

WHERE T.COL\_NAME = SUBSTR(T.TABLE\_NAME, 1, 27) || '\_LB'

AND EXISTS(SELECT \* FROM PON\_DICT WHERE TABLE\_NAME = T.TABLE\_NAME AND PK = 1);

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-----------------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------- END OF ADD GUIDS. --------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------------

-- Enable disabled NOT NULL constraints.

-- If successful this will prevent "ORA-01442: column to be modified to NOT NULL is already NOT NULL" when making columns required in the next block. 2.

-- Will fail with "ORA-02293: cannot validate" if there is bad data.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Enable disabled NOT NULL constraints.');

-- Drop disabled NOT NULL constraints which make the next code fail.

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || D.TABLE\_NAME || ' ENABLE CONSTRAINT ' || CONS.CONSTRAINT\_NAME

FROM PON\_DICT D

JOIN USER\_TAB\_COLUMNS C ON C.TABLE\_NAME = D.TABLE\_NAME AND C.COLUMN\_NAME = D.COL\_NAME

JOIN USER\_CONSTRAINTS CONS ON CONS.TABLE\_NAME = D.TABLE\_NAME

JOIN USER\_CONS\_COLUMNS CONS\_COLS ON CONS\_COLS.CONSTRAINT\_NAME = CONS.CONSTRAINT\_NAME AND CONS\_COLS.COLUMN\_NAME = D.COL\_NAME

WHERE D.REQUIRED = 1 AND

D.PK <> 1 AND

C.NULLABLE = 'Y' AND

D.COL\_NAME NOT LIKE '%CAPTION\_ID' AND

CONS.STATUS = 'DISABLED' AND

CONS\_COLS.POSITION IS NULL AND

INSTR(F\_READ\_SEARCH\_CONDITION(CONS.CONSTRAINT\_NAME), 'IS NOT NULL') > 0

ORDER BY D.TABLE\_NAME, D.COL\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Make some non PK columns required.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Make non PK GUID columns required.');

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || D.TABLE\_NAME || ' MODIFY ' || D.COL\_NAME || ' ' || ' NOT NULL'

FROM PON\_DICT D

JOIN USER\_TAB\_COLUMNS C ON C.TABLE\_NAME = D.TABLE\_NAME AND C.COLUMN\_NAME = D.COL\_NAME

WHERE D.REQUIRED = 1 AND D.PK <> 1 AND C.NULLABLE = 'Y' AND D.COL\_NAME NOT LIKE '%CAPTION\_ID'

ORDER BY D.TABLE\_NAME, D.COL\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Change 'USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT' in PON\_EXCHANGE\_OPTION\_DETAIL to the table names in PON\_COPTIONS.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'UPDATE PON\_EXCHANGE\_OPTION\_DETAIL SET TABLE\_NAME = ''' || TABLE\_NAME || ''' ' ||

'WHERE TABLE\_NAME = ''' || ORIGINAL\_TABLE\_NAME || ''''

FROM PON\_TABLE

WHERE ORIGINAL\_TABLE\_NAME IN (

'USERBRDG',

'USERINSP',

'USERRWAY',

'USERSTRUNIT'

)

AND TABLE\_NAME <> ORIGINAL\_TABLE\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Replace 'USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT' in PON\_EXCHANGE\_OPTION.OPTION\_NAME with the table names in PON\_COPTIONS.

-- WIP Do this to \_T instead. Same problem in SQL.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'UPDATE PON\_EXCHANGE\_OPTION SET OPTION\_NAME = REPLACE(OPTION\_NAME, ''' || ORIGINAL\_TABLE\_NAME || ''', ''' || TABLE\_NAME || ''') ' ||

'WHERE OPTION\_NAME LIKE ''%' || ORIGINAL\_TABLE\_NAME || ''''

FROM PON\_TABLE

WHERE ORIGINAL\_TABLE\_NAME IN (

'USERBRDG',

'USERINSP',

'USERRWAY',

'USERSTRUNIT'

)

AND TABLE\_NAME <> ORIGINAL\_TABLE\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Replace 'USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT' in PON\_FILTERS.SQL\_FILTER with the table names in PON\_COPTIONS.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'UPDATE PON\_FILTERS SET SQL\_FILTER = REPLACE(SQL\_FILTER, ''' || ORIGINAL\_TABLE\_NAME || ''', LOWER(''' || TABLE\_NAME || ''')) ' ||

'WHERE SQL\_FILTER LIKE ''%' || ORIGINAL\_TABLE\_NAME || '%'' AND PONTIS\_STANDARD\_IND = ''T'''

FROM PON\_TABLE

WHERE ORIGINAL\_TABLE\_NAME IN (

'USERBRDG',

'USERINSP',

'USERRWAY',

'USERSTRUNIT'

)

AND TABLE\_NAME <> ORIGINAL\_TABLE\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Add missing permissions.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Add missing permissions, (if any).');

--ADD MISSING TASK SECURITY for top level tasks without tab security.

INSERT INTO PON\_APP\_TASK\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_TASK\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_TASK\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' AND TASK\_NAME = 'VIEW LIST' THEN 'T' ELSE 'F' END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' AND TASK\_NAME = 'VIEW LIST' THEN 'F' ELSE 'T' END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' AND TASK\_NAME = 'VIEW LIST' THEN 'T' ELSE 'F' END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' AND TASK\_NAME = 'VIEW LIST' THEN 'F' ELSE 'T' END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, PON\_NAV\_TASK\_GD, TAB\_NAME, TASK\_NAME

FROM PON\_APP\_ROLES R, (SELECT TASK.PON\_NAV\_TAB\_GD, TASK.PON\_NAV\_TASK\_GD, TAB.NAME TAB\_NAME, TASK.NAME TASK\_NAME

FROM PON\_NAV\_TAB TAB

JOIN PON\_NAV\_TASK TASK ON TASK.PON\_NAV\_TAB\_GD = TAB.PON\_NAV\_TAB\_GD

WHERE PARENT\_PON\_NAV\_TASK\_GD IS NULL

) T

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_TASK\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_TASK\_GD = ROLE\_NAV.PON\_NAV\_TASK\_GD

LEFT OUTER JOIN PON\_APP\_TAB\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_TAB\_GD = ROLE\_NAV.PON\_NAV\_TAB\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL AND PARENT\_SECURITY.PON\_APP\_ROLES\_GD IS NULL

ORDER BY 1, 2, 3;

-- ADD MISSING TAB SECURITY

INSERT INTO PON\_APP\_TAB\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_TAB\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' THEN 'T' ELSE 'F' END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' THEN 'F' ELSE 'T' END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' THEN 'T' ELSE 'F' END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' OR TAB\_NAME = 'BRIDGES' THEN 'F' ELSE 'T' END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, NAME TAB\_NAME

FROM PON\_APP\_ROLES, PON\_NAV\_TAB

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_TAB\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_TAB\_GD = ROLE\_NAV.PON\_NAV\_TAB\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL

ORDER BY 1, 2;

-- ADD MISSING TASK SECURITY 1, cascade tab security to top level tasks.

INSERT INTO PON\_APP\_TASK\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_TASK\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_TASK\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.VISIBLE\_IND END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.READ\_ONLY\_IND END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.MOBILE\_VISIBLE\_IND END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.MOBILE\_READ\_ONLY\_IND END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, PON\_NAV\_TASK\_GD, PARENT\_PON\_NAV\_TASK\_GD

FROM PON\_APP\_ROLES, PON\_NAV\_TASK

WHERE PARENT\_PON\_NAV\_TASK\_GD IS NULL

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_TASK\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_TASK\_GD = ROLE\_NAV.PON\_NAV\_TASK\_GD

LEFT OUTER JOIN PON\_APP\_TAB\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_TAB\_GD = ROLE\_NAV.PON\_NAV\_TAB\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL

ORDER BY 1, 2, 3;

-- ADD second level task security.

INSERT INTO PON\_APP\_TASK\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_TASK\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_TASK\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.VISIBLE\_IND END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.READ\_ONLY\_IND END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.MOBILE\_VISIBLE\_IND END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.MOBILE\_READ\_ONLY\_IND END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, PON\_NAV\_TASK\_GD, PARENT\_PON\_NAV\_TASK\_GD

FROM

(SELECT PON\_APP\_ROLES\_GD FROM PON\_APP\_ROLES) A,

(

SELECT CHILD.PON\_NAV\_TAB\_GD, CHILD.PON\_NAV\_TASK\_GD, CHILD.PARENT\_PON\_NAV\_TASK\_GD

FROM PON\_NAV\_TASK PARENT

JOIN PON\_NAV\_TASK CHILD ON CHILD.PON\_NAV\_TAB\_GD = PARENT.PON\_NAV\_TAB\_GD

AND CHILD.PARENT\_PON\_NAV\_TASK\_GD = PARENT.PON\_NAV\_TASK\_GD

AND PARENT.PARENT\_PON\_NAV\_TASK\_GD IS NULL

) B

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_TASK\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_TASK\_GD = ROLE\_NAV.PON\_NAV\_TASK\_GD

LEFT OUTER JOIN PON\_APP\_TAB\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_TAB\_GD = ROLE\_NAV.PON\_NAV\_TAB\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL

ORDER BY 1, 2, 3;

-- ADD third level task security

INSERT INTO PON\_APP\_TASK\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_TASK\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_TASK\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.VISIBLE\_IND END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.READ\_ONLY\_IND END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.MOBILE\_VISIBLE\_IND END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.MOBILE\_READ\_ONLY\_IND END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TAB\_GD, PON\_NAV\_TASK\_GD, PARENT\_PON\_NAV\_TASK\_GD

FROM

(SELECT PON\_APP\_ROLES\_GD FROM PON\_APP\_ROLES) A,

(

SELECT CHILD2.PON\_NAV\_TAB\_GD, CHILD2.PON\_NAV\_TASK\_GD, CHILD2.PARENT\_PON\_NAV\_TASK\_GD

FROM PON\_NAV\_TASK PARENT

JOIN PON\_NAV\_TASK CHILD ON CHILD.PON\_NAV\_TAB\_GD = PARENT.PON\_NAV\_TAB\_GD

AND CHILD.PARENT\_PON\_NAV\_TASK\_GD = PARENT.PON\_NAV\_TASK\_GD

AND PARENT.PARENT\_PON\_NAV\_TASK\_GD IS NULL

JOIN PON\_NAV\_TASK CHILD2 ON CHILD2.PON\_NAV\_TAB\_GD = CHILD.PON\_NAV\_TAB\_GD

AND CHILD2.PARENT\_PON\_NAV\_TASK\_GD = CHILD.PON\_NAV\_TASK\_GD

) B

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_TASK\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_TASK\_GD = ROLE\_NAV.PON\_NAV\_TASK\_GD

LEFT OUTER JOIN PON\_APP\_TAB\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_TAB\_GD = ROLE\_NAV.PON\_NAV\_TAB\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL

ORDER BY 1, 2, 3;

-- ADD CONTROL GROUP SECURITY.

INSERT INTO PON\_APP\_CONTROL\_GROUP\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_CONTROL\_GROUP\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_CONTROL\_GROUP\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.VISIBLE\_IND END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.READ\_ONLY\_IND END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.MOBILE\_VISIBLE\_IND END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.MOBILE\_READ\_ONLY\_IND END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_TASK\_GD, PON\_NAV\_CONTROL\_GROUP\_GD

FROM PON\_APP\_ROLES, PON\_NAV\_CONTROL\_GROUP

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_CONTROL\_GROUP\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_CONTROL\_GROUP\_GD = ROLE\_NAV.PON\_NAV\_CONTROL\_GROUP\_GD

LEFT OUTER JOIN PON\_APP\_TASK\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_TASK\_GD = ROLE\_NAV.PON\_NAV\_TASK\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL;

-- ADD CONTROL SECURITY.

INSERT INTO PON\_APP\_CONTROL\_SECURITY (PON\_APP\_ROLES\_GD, PON\_NAV\_CONTROL\_GD, VISIBLE\_IND, READ\_ONLY\_IND, MOBILE\_VISIBLE\_IND, MOBILE\_READ\_ONLY\_IND)

SELECT

ROLE\_NAV.PON\_APP\_ROLES\_GD,

ROLE\_NAV.PON\_NAV\_CONTROL\_GD,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.VISIBLE\_IND END VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.READ\_ONLY\_IND END READ\_ONLY\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'T' ELSE PARENT\_SECURITY.MOBILE\_VISIBLE\_IND END MOBILE\_VISIBLE\_IND,

CASE WHEN ROLE\_NAV.PON\_APP\_ROLES\_GD = '3B6F530FDFF44FF0811700BECA1FC33B' THEN 'F' ELSE PARENT\_SECURITY.MOBILE\_READ\_ONLY\_IND END MOBILE\_READ\_ONLY\_IND

FROM (

SELECT PON\_APP\_ROLES\_GD, PON\_NAV\_CONTROL\_GROUP\_GD, PON\_NAV\_CONTROL\_GD

FROM PON\_APP\_ROLES, PON\_NAV\_CONTROL

) ROLE\_NAV

LEFT OUTER JOIN PON\_APP\_CONTROL\_SECURITY NAV\_SECURITY ON NAV\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND NAV\_SECURITY.PON\_NAV\_CONTROL\_GD = ROLE\_NAV.PON\_NAV\_CONTROL\_GD

LEFT OUTER JOIN PON\_APP\_CONTROL\_GROUP\_SECURITY PARENT\_SECURITY ON PARENT\_SECURITY.PON\_APP\_ROLES\_GD = ROLE\_NAV.PON\_APP\_ROLES\_GD

AND PARENT\_SECURITY.PON\_NAV\_CONTROL\_GROUP\_GD = ROLE\_NAV.PON\_NAV\_CONTROL\_GROUP\_GD

WHERE NAV\_SECURITY.PON\_APP\_ROLES\_GD IS NULL;

END;

/

-- Add indexes.

DECLARE V\_CUR SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Add indexes.');

OPEN V\_CUR FOR

SELECT DISTINCT 'CREATE ' || CASE WHEN PI.IS\_UNIQUE = 1 THEN 'UNIQUE ' ELSE '' END || 'INDEX ' || PI.INDEX\_NAME || ' ON ' || PI.TABLE\_NAME || '(' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME

FROM PON\_INDEX

WHERE INDEX\_NAME = PI.INDEX\_NAME

ORDER BY COL\_ORDER

),

','

) ||

')'

FROM (

SELECT PI\_T.\*, COUNT(\*) OVER (PARTITION BY PI\_T.INDEX\_NAME) COL\_COUNT

FROM PON\_INDEX PI\_T

) PI

LEFT OUTER JOIN (

SELECT

UI.TABLE\_NAME,

UI.INDEX\_NAME,

CASE WHEN UI.UNIQUENESS = 'UNIQUE' THEN 1 ELSE 0 END IS\_UNIQUE,

UIC.COLUMN\_NAME,

UIC.COLUMN\_POSITION,

COUNT(\*) OVER (PARTITION BY UIC.INDEX\_NAME) COL\_COUNT

FROM USER\_INDEXES UI

JOIN USER\_IND\_COLUMNS UIC ON UIC.INDEX\_NAME = UI.INDEX\_NAME

) I ON I.TABLE\_NAME = PI.TABLE\_NAME AND

I.IS\_UNIQUE = PI.IS\_UNIQUE AND

I.COLUMN\_NAME = PI.COL\_NAME AND

I.COLUMN\_POSITION = PI.COL\_ORDER AND

I.COL\_COUNT = PI.COL\_COUNT

WHERE I.TABLE\_NAME IS NULL;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Views.

BEGIN

IF GET\_VAR('SVW') = 1 THEN

BEGIN

-- 1

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE VIEW V\_PON\_APP\_USER\_PERMISSIONS AS

SELECT DISTINCT

U.USERKEY,

U.PON\_APP\_USERS\_GD,

G.GROUPKEY,

G.PON\_APP\_GROUPS\_GD,

U.USERID,

U.FIRST\_NAME,

U.LAST\_NAME,

U.AGENCY,

U.PHONE,

U.EMAIL,

U.STATUS,

R.ROLEKEY,

R.PON\_APP\_ROLES\_GD,

R.ROLENAME,

P.PERMISSIONKEY,

P.PON\_APP\_PERMISSIONS\_GD,

P.PERMISSIONNAME,

RP.GRANTED

FROM

PON\_APP\_USERS U

LEFT JOIN PON\_APP\_USERS\_ROLES UR ON U.PON\_APP\_USERS\_GD = UR.PON\_APP\_USERS\_GD

LEFT JOIN PON\_APP\_ROLES R ON UR.PON\_APP\_ROLES\_GD = R.PON\_APP\_ROLES\_GD

LEFT JOIN PON\_APP\_USERS\_GROUPS UG ON U.PON\_APP\_USERS\_GD = UG.PON\_APP\_USERS\_GD

LEFT JOIN PON\_APP\_GROUPS G ON G.PON\_APP\_GROUPS\_GD = UG.PON\_APP\_GROUPS\_GD

LEFT JOIN PON\_APP\_ROLES\_PERMISSIONS RP ON RP.PON\_APP\_ROLES\_GD = R.PON\_APP\_ROLES\_GD

LEFT JOIN PON\_APP\_PERMISSIONS P ON P.PON\_APP\_PERMISSIONS\_GD = RP.PON\_APP\_PERMISSIONS\_GD';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON V\_PON\_APP\_USER\_PERMISSIONS TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 2

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE VIEW V\_ELEMVALUE AS

SELECT

PON\_ELEM\_DEFS.ELEM\_KEY AS ELEMKEY,

PON\_ELEM\_DEFS.PON\_ELEM\_DEFS\_GD,

PON\_ELEM\_DEFS.ELEM\_REL\_WEIGHT AS ELEMVALUE,

(2.0/3.0) AS S2FACTOR,

(1.0/3.0) AS S3FACTOR,

0.0 AS S4FACTOR

FROM PON\_ELEM\_DEFS

WHERE PON\_ELEM\_DEFS.ELEM\_SMART\_FLAG = ''N''';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON V\_ELEMVALUE TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 3

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE VIEW PON\_ELEM\_SMFLG\_VIEW as

SELECT

INSP.BRKEY,

INSP.BRIDGE\_GD,

INSP.INSPKEY,

INVT\_INSP.INSPEVNT\_GD,

PARENT\_ELEM\_DEFS.ELEM\_KEY PARENT\_ELEM\_KEY,

PARENT\_INVT.PON\_ELEM\_DEFS\_GD PARENT\_PON\_ELEM\_DEFS\_GD,

INVT.PARENT\_PON\_ELEM\_INVENTORY\_GD,

ELEM\_DEFS.ELEM\_KEY,

ELEM\_DEFS.PON\_ELEM\_DEFS\_GD,

ENVT\_DEFS.ENVKEY,

INVT.PON\_ENVT\_DEFS\_GD,

STRU.STRUNITKEY,

INVT.STRUCTURE\_UNIT\_GD,

INVT.TOTAL\_QUANTITY,

INVT.SCALE\_FACTOR,

(INVT\_INSP.QTYSTATE1/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE1,

(INVT\_INSP.QTYSTATE2/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE2,

(INVT\_INSP.QTYSTATE3/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE3,

(INVT\_INSP.QTYSTATE4/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE4,

INVT\_INSP.QTYSTATE1,

INVT\_INSP.QTYSTATE2,

INVT\_INSP.QTYSTATE3,

INVT\_INSP.QTYSTATE4,

INVT.INVT\_DESC,

INVT\_INSP.CREATEDATETIME,

INVT\_INSP.CREATEUSERKEY,

INVT\_INSP.MODTIME,

INVT\_INSP.MODUSERKEY,

INVT\_INSP.DOCREFKEY,

INVT\_INSP.INVT\_INSP\_NOTES

FROM PON\_ELEM\_INVENTORY INVT

JOIN PON\_ELEM\_INVENTORY\_INSP INVT\_INSP ON INVT\_INSP.PON\_ELEM\_INVENTORY\_GD = INVT.PON\_ELEM\_INVENTORY\_GD

JOIN PON\_ELEM\_DEFS ELEM\_DEFS ON ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN PON\_ENVT\_DEFS ENVT\_DEFS ON ENVT\_DEFS.PON\_ENVT\_DEFS\_GD = INVT.PON\_ENVT\_DEFS\_GD

JOIN INSPEVNT INSP ON INSP.INSPEVNT\_GD = INSP.INSPEVNT\_GD

LEFT JOIN PON\_ELEM\_INVENTORY PARENT\_INVT ON PARENT\_INVT.PON\_ELEM\_INVENTORY\_GD = INVT.PARENT\_PON\_ELEM\_INVENTORY\_GD

LEFT JOIN PON\_ELEM\_DEFS PARENT\_ELEM\_DEFS ON PARENT\_ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = PARENT\_INVT.PON\_ELEM\_DEFS\_GD

JOIN STRUCTURE\_UNIT STRU ON STRU.STRUCTURE\_UNIT\_GD = INVT.STRUCTURE\_UNIT\_GD

WHERE ELEM\_DEFS.ELEM\_PROTECT\_SYS = ''N''

AND ELEM\_DEFS.ELEM\_SMART\_FLAG = ''Y''

AND INVT.DEACTIVATED\_ON IS NULL';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON PON\_ELEM\_SMFLG\_VIEW TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 4

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE VIEW PON\_ELEM\_PROTECT\_VIEW as

SELECT

INSP.BRKEY,

INSP.BRIDGE\_GD,

INSP.INSPKEY,

INVT\_INSP.INSPEVNT\_GD,

PARENT\_ELEM\_DEFS.ELEM\_KEY PARENT\_ELEM\_KEY,

PARENT\_INVT.PON\_ELEM\_DEFS\_GD PARENT\_PON\_ELEM\_DEFS\_GD,

INVT.PARENT\_PON\_ELEM\_INVENTORY\_GD,

ELEM\_DEFS.ELEM\_KEY,

ELEM\_DEFS.PON\_ELEM\_DEFS\_GD,

ENVT\_DEFS.ENVKEY,

INVT.PON\_ENVT\_DEFS\_GD,

STRU.STRUNITKEY,

INVT.STRUCTURE\_UNIT\_GD,

INVT.TOTAL\_QUANTITY,

INVT.SCALE\_FACTOR,

(INVT\_INSP.QTYSTATE1/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE1,

(INVT\_INSP.QTYSTATE2/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE2,

(INVT\_INSP.QTYSTATE3/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE3,

(INVT\_INSP.QTYSTATE4/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE4,

INVT\_INSP.QTYSTATE1,

INVT\_INSP.QTYSTATE2,

INVT\_INSP.QTYSTATE3,

INVT\_INSP.QTYSTATE4,

INVT.INVT\_DESC,

INVT\_INSP.CREATEDATETIME,

INVT\_INSP.CREATEUSERKEY,

INVT\_INSP.MODTIME,

INVT\_INSP.MODUSERKEY,

INVT\_INSP.DOCREFKEY,

INVT\_INSP.INVT\_INSP\_NOTES

FROM PON\_ELEM\_INVENTORY INVT

JOIN PON\_ELEM\_INVENTORY\_INSP INVT\_INSP ON INVT\_INSP.PON\_ELEM\_INVENTORY\_GD = INVT.PON\_ELEM\_INVENTORY\_GD

JOIN PON\_ELEM\_DEFS ELEM\_DEFS ON ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN PON\_ENVT\_DEFS ENVT\_DEFS ON ENVT\_DEFS.PON\_ENVT\_DEFS\_GD = INVT.PON\_ENVT\_DEFS\_GD

JOIN INSPEVNT INSP ON INSP.INSPEVNT\_GD = INSP.INSPEVNT\_GD

LEFT JOIN PON\_ELEM\_INVENTORY PARENT\_INVT ON PARENT\_INVT.PON\_ELEM\_INVENTORY\_GD = INVT.PARENT\_PON\_ELEM\_INVENTORY\_GD

LEFT JOIN PON\_ELEM\_DEFS PARENT\_ELEM\_DEFS ON PARENT\_ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = PARENT\_INVT.PON\_ELEM\_DEFS\_GD

JOIN STRUCTURE\_UNIT STRU ON STRU.STRUCTURE\_UNIT\_GD = INVT.STRUCTURE\_UNIT\_GD

WHERE ELEM\_DEFS.ELEM\_PROTECT\_SYS = ''Y''

AND ELEM\_DEFS.ELEM\_SMART\_FLAG = ''N''

AND INVT.DEACTIVATED\_ON IS NULL';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON PON\_ELEM\_PROTECT\_VIEW TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 5

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE VIEW PON\_ELEM\_PARENT\_VIEW as

SELECT

INSP.BRKEY,

INSP.BRIDGE\_GD,

INSP.INSPKEY,

INVT\_INSP.INSPEVNT\_GD,

ELEM\_DEFS.ELEM\_KEY,

ELEM\_DEFS.PON\_ELEM\_DEFS\_GD,

ENVT\_DEFS.ENVKEY,

INVT.PON\_ENVT\_DEFS\_GD,

STRU.STRUNITKEY,

INVT.STRUCTURE\_UNIT\_GD,

INVT.TOTAL\_QUANTITY,

INVT.SCALE\_FACTOR,

(INVT\_INSP.QTYSTATE1/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE1,

(INVT\_INSP.QTYSTATE2/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE2,

(INVT\_INSP.QTYSTATE3/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE3,

(INVT\_INSP.QTYSTATE4/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END) \* 100 PCTSTATE4,

INVT\_INSP.QTYSTATE1,

INVT\_INSP.QTYSTATE2,

INVT\_INSP.QTYSTATE3,

INVT\_INSP.QTYSTATE4,

INVT.INVT\_DESC,

INVT\_INSP.CREATEDATETIME,

INVT\_INSP.CREATEUSERKEY,

INVT\_INSP.MODTIME,

INVT\_INSP.MODUSERKEY,

INVT\_INSP.DOCREFKEY,

INVT\_INSP.INVT\_INSP\_NOTES

FROM PON\_ELEM\_INVENTORY INVT

JOIN PON\_ELEM\_INVENTORY\_INSP INVT\_INSP ON INVT\_INSP.PON\_ELEM\_INVENTORY\_GD = INVT.PON\_ELEM\_INVENTORY\_GD

JOIN PON\_ELEM\_DEFS ELEM\_DEFS ON ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN PON\_ENVT\_DEFS ENVT\_DEFS ON ENVT\_DEFS.PON\_ENVT\_DEFS\_GD = INVT.PON\_ENVT\_DEFS\_GD

JOIN INSPEVNT INSP ON INSP.INSPEVNT\_GD = INSP.INSPEVNT\_GD

JOIN STRUCTURE\_UNIT STRU ON STRU.STRUCTURE\_UNIT\_GD = INVT.STRUCTURE\_UNIT\_GD

WHERE INVT.DEACTIVATED\_ON IS NULL

AND INVT.PARENT\_PON\_ELEM\_INVENTORY\_GD IS NULL';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON PON\_ELEM\_PARENT\_VIEW TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 6

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE FORCE VIEW V\_HIX AS

SELECT

INSP.BRKEY,

INSP.BRIDGE\_GD,

INSP.INSPKEY,

INSP.INSPEVNT\_GD,

TYPE\_DEFS.ECATKEY,

TYPE\_DEFS.PON\_ELEM\_CAT\_DEFS\_GD,

ROUND(

(CASE

WHEN SUM(INVT.TOTAL\_QUANTITY\*EV.ELEMVALUE) > 0 THEN

(

SUM(

INVT.TOTAL\_QUANTITY\*EV.ELEMVALUE

\*(

(INVT\_INSP.QTYSTATE1/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S2FACTOR\*(INVT\_INSP.QTYSTATE2/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S3FACTOR\*(INVT\_INSP.QTYSTATE3/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S4FACTOR\*(INVT\_INSP.QTYSTATE4/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

)

)/SUM(INVT.TOTAL\_QUANTITY\*ELEMVALUE)

)

ELSE -1

END

),

2

) AS HIX

FROM PON\_ELEM\_INVENTORY\_INSP INVT\_INSP

JOIN PON\_ELEM\_INVENTORY INVT ON INVT.PON\_ELEM\_INVENTORY\_GD = INVT\_INSP.PON\_ELEM\_INVENTORY\_GD

JOIN V\_ELEMVALUE EV ON EV.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN INSPEVNT INSP ON INSP.INSPEVNT\_GD = INVT\_INSP.INSPEVNT\_GD

JOIN PON\_ELEM\_DEFS ELEM\_DEFS ON ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN PON\_ELEM\_TYPE\_DEFS TYPE\_DEFS ON TYPE\_DEFS.PON\_ELEM\_TYPE\_DEFS\_GD = ELEM\_DEFS.PON\_ELEM\_TYPE\_DEFS\_GD

WHERE INVT.DEACTIVATED\_ON IS NULL

GROUP BY INSP.BRIDGE\_GD, INSP.BRKEY, INSP.INSPEVNT\_GD, INSP.INSPKEY, TYPE\_DEFS.ECATKEY, TYPE\_DEFS.PON\_ELEM\_CAT\_DEFS\_GD';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON PON\_ELEM\_PARENT\_VIEW TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

-- 7

DECLARE

v\_q VARCHAR2(32767);

V\_PONTISUSER\_EXISTS PLS\_INTEGER := 0;

BEGIN

v\_q := 'CREATE OR REPLACE FORCE VIEW V\_HIX\_BREAKOUT AS

SELECT

INSP.BRKEY,

INSP.BRIDGE\_GD,

INSP.INSPKEY,

INSP.INSPEVNT\_GD,

TYPE\_DEFS.ECATKEY,

TYPE\_DEFS.PON\_ELEM\_CAT\_DEFS\_GD,

ROUND(

(CASE

WHEN SUM(INVT.TOTAL\_QUANTITY\*EV.ELEMVALUE) > 0 THEN

(

SUM(

INVT.TOTAL\_QUANTITY\*EV.ELEMVALUE

\*(

(INVT\_INSP.QTYSTATE1/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S2FACTOR\*(INVT\_INSP.QTYSTATE2/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S3FACTOR\*(INVT\_INSP.QTYSTATE3/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

+ EV.S4FACTOR\*(INVT\_INSP.QTYSTATE4/CASE WHEN INVT.TOTAL\_QUANTITY = 0 THEN NULL ELSE INVT.TOTAL\_QUANTITY END)\*100

)

)/SUM(INVT.TOTAL\_QUANTITY\*ELEMVALUE)

)

ELSE -1

END

),

2

) AS HIX

FROM PON\_ELEM\_INVENTORY\_INSP INVT\_INSP

JOIN PON\_ELEM\_INVENTORY INVT ON INVT.PON\_ELEM\_INVENTORY\_GD = INVT\_INSP.PON\_ELEM\_INVENTORY\_GD

JOIN V\_ELEMVALUE EV ON EV.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN INSPEVNT INSP ON INSP.INSPEVNT\_GD = INVT\_INSP.INSPEVNT\_GD

JOIN PON\_ELEM\_DEFS ELEM\_DEFS ON ELEM\_DEFS.PON\_ELEM\_DEFS\_GD = INVT.PON\_ELEM\_DEFS\_GD

JOIN PON\_ELEM\_TYPE\_DEFS TYPE\_DEFS ON TYPE\_DEFS.PON\_ELEM\_TYPE\_DEFS\_GD = ELEM\_DEFS.PON\_ELEM\_TYPE\_DEFS\_GD

WHERE INVT.DEACTIVATED\_ON IS NULL

GROUP BY INSP.BRIDGE\_GD, INSP.BRKEY, INSP.INSPEVNT\_GD, INSP.INSPKEY, TYPE\_DEFS.ECATKEY, TYPE\_DEFS.PON\_ELEM\_CAT\_DEFS\_GD';

EXECUTE IMMEDIATE(v\_q);

SELECT COUNT(\*) INTO V\_PONTISUSER\_EXISTS FROM SESSION\_ROLES WHERE ROLE = 'PONTISUSER';

IF V\_PONTISUSER\_EXISTS = 1 THEN

V\_Q := 'GRANT SELECT ON PON\_ELEM\_PARENT\_VIEW TO PONTISUSER';

EXECUTE IMMEDIATE(V\_Q);

END IF;

END;

END;

END IF;

END;

/

-- Triggers.

DECLARE v\_temp number :=0;

v\_q VARCHAR2(32767);

BEGIN

IF GET\_VAR('STG') = 1 THEN

-- 1

EXECUTE IMMEDIATE('CREATE OR REPLACE TRIGGER STRUCTURE\_UNIT\_DELTRG

AFTER DELETE ON STRUCTURE\_UNIT

FOR EACH ROW

BEGIN

-- Delete of PON\_ELEM\_INSP is done by ON DELETE CASCADE in new FK.

DELETE PON\_INSP\_WORKCAND

WHERE STRUCTURE\_UNIT\_GD = :OLD.STRUCTURE\_UNIT\_GD;

END;');

-- 2

EXECUTE IMMEDIATE('CREATE OR REPLACE

TRIGGER TRG\_PON\_ELEM\_INSP

AFTER INSERT ON PON\_ELEM\_INSP

FOR EACH ROW

DECLARE

isConverted number;

SRCBRKEY PON\_ELEM\_INSP.BRKEY%TYPE;

SRCINSPKEY PON\_ELEM\_INSP.INSPKEY%TYPE;

BEGIN

SRCBRKEY := :new.BRKEY;

SRCINSPKEY := :new.INSPKEY;

SELECT COUNT(SRCBRKEY) into isConverted

FROM ELEMINSP

WHERE BRKEY = SRCBRKEY

AND INSPKEY= SRCINSPKEY

AND ROWNUM = 1;

IF isConverted > 0 THEN

UPDATE INSPEVNT

SET ELEMCONVERT = ''Y''

WHERE BRKEY = SRCBRKEY

AND INSPKEY = SRCINSPKEY;

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END TRG\_PON\_ELEM\_INSP;');

END IF;

END;

/

-- F.3.0 | Insert Procedures.

DECLARE v\_temp number :=0;

BEGIN

SELECT VAR\_VALUE INTO v\_temp FROM PON\_GLOB\_VAR WHERE VARI = 'PDR';

IF v\_temp = 1 THEN

EXECUTE IMMEDIATE(

'CREATE OR REPLACE PROCEDURE CreateDataDictEntries\_SP

(iv\_TableName IN VARCHAR2 DEFAULT NULL)

AS

v\_TableName VARCHAR2(30) := iv\_TableName;

BEGIN

v\_TableName := UPPER(RTRIM(v\_TableName)) ;

INSERT INTO DATADICT

( TABLE\_NAME, COL\_NAME, COL\_ALIAS, V2CONVERT, DATATYPE, WIDTH, DEC\_PLCS, NULL\_ALLOW, UNIQUEKEY, POSITION, NBI\_CD, VALTYPE, VALATTR1, VALATTR2, SYSFIELD, SYSDEFAULT, KEYATTR1, UNIQUE\_FLD, HELPID, PAIRCODE, CONVERSIONRULES, SNOTES, NOTES, CUSTOM\_FORM\_INCLUDE)

SELECT LOWER(col.TABLE\_NAME),

LOWER(col.COLUMN\_NAME),

LOWER(col.COLUMN\_NAME),

''-1'', --V2CONVERT

LOWER(col.DATA\_TYPE),

--WIDTH Width for storage and display.

CASE WHEN col.DATA\_TYPE IN (''CHAR'', ''VARCHAR2'', ''VARCHAR'', ''NCHAR'', ''NVARCHAR2'', ''NVARCHAR'')

THEN col.CHAR\_LENGTH

ELSE COL.DATA\_LENGTH

END,

col.DATA\_SCALE, --DEC\_PLCS

CASE WHEN col.NULLABLE IS NOT NULL --NULL\_ALLOW

THEN COL.NULLABLE

ELSE ''Y''

END,

-- UNIQUEKEY, Flag indicating whether field value must be unique.

-- Find columns that are the only column in a primary or unique constraint.

CASE

WHEN col.COLUMN\_NAME IN (SELECT MAX(COLS.COLUMN\_NAME)

FROM USER\_CONSTRAINTS C

INNER JOIN USER\_CONS\_COLUMNS COLS ON C.CONSTRAINT\_NAME = COLS.CONSTRAINT\_NAME

WHERE C.TABLE\_NAME = v\_TableName

AND C.CONSTRAINT\_TYPE IN (''P'', ''U'')

GROUP BY COLS.CONSTRAINT\_NAME

HAVING COUNT(\*) = 1)

THEN ''Y''

ELSE ''N''

END,

col.COLUMN\_ID , --POSITION

''-1'', --NBI\_CD

''-1'', --VALTYPE

''-1'', --VALATTR1

''-1'', --VALATTR2

''\_'', --SYSFIELD

''-1'', --SYSDEFAULT

CASE WHEN col.COLUMN\_NAME IN (SELECT COLS.COLUMN\_NAME

FROM USER\_CONSTRAINTS C

JOIN USER\_CONS\_COLUMNS COLS

ON C.CONSTRAINT\_NAME = COLS.CONSTRAINT\_NAME

WHERE C.TABLE\_NAME = v\_TableName

AND C.CONSTRAINT\_TYPE = ''P'')

THEN ''Y''

ELSE ''N''

END, --KEYATTR1

LOWER(col.COLUMN\_NAME), --UNIQUE\_FLD

-1 , --HELPID

-1 , --PAIRCODE

''-1'' , --CONVERSIONRULES

''This is table '' || col.TABLE\_NAME || '' column'' || col.COLUMN\_NAME , --SNOTES

''This is table '' || col.TABLE\_NAME || '' column'' || col.COLUMN\_NAME, --NOTES

''-1'' --CUSTOM\_FORM\_INCLUDE

FROM USER\_TAB\_COLUMNS col

WHERE col.TABLE\_NAME = V\_TABLENAME

AND col.COLUMN\_NAME NOT IN ( SELECT UPPER(RTRIM(DD.COL\_NAME))

FROM DATADICT DD

WHERE UPPER(RTRIM(DD.TABLE\_NAME)) = V\_TABLENAME);

END;');

END IF;

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('TRIGGERS AND PROCEDURES', CURRENT\_TIMESTAMP);

END;

/

-- = G | Drop \_T tables =========================================================== --

DECLARE

v\_temp NUMBER(3, 0) := 0;

v\_TN1 VARCHAR2(30);

v\_q VARCHAR2(1000);

CURSOR curColData IS

SELECT UT.TABLE\_NAME

FROM USER\_TABLES UT

JOIN PON\_TABLE T ON SUBSTR(T.TABLE\_NAME, 1, 28) || '\_T' = UT.TABLE\_NAME;

BEGIN

IF GET\_VAR('PCA') = '1' THEN

/\*\* G.1.1 | Start a loop through the cursor to create a dynamic drop process \*\*/

OPEN curColData;

FETCH curColData INTO v\_TN1;

WHILE (curColData%FOUND)

LOOP

BEGIN

v\_q := 'DROP TABLE ' || V\_TN1 ;

execute immediate v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F1: Dropping Table [' || v\_q || ']; ' || SQLERRM);

END;

FETCH curColData INTO v\_TN1;

END LOOP;

CLOSE curColData;

END IF;

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('DROP TEMP TABLES', CURRENT\_TIMESTAMP);

END;

/

DECLARE

v\_temp number :=0;

v\_q VARCHAR2(1000);

BEGIN

IF GET\_VAR('PCA') = '1' THEN

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_FK' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_FK';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F2: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_INDEX' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_INDEX';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F3: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_DICT' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_DICT';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F2: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_TABLE' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_TABLE';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F4: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_RETIRED\_TABLE' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_RETIRED\_TABLE';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F4: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(table\_name) = 'PON\_GLOB\_VAR' AND ROWNUM=1;

IF v\_temp > 0 THEN

v\_q := 'DROP TABLE PON\_GLOB\_VAR';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F5: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from USER\_PROCEDURES where PROCEDURE\_NAME = 'F\_READ\_DATA\_DEFAULT';

IF v\_temp > 0 THEN

v\_q := 'DROP PROCEDURE F\_READ\_DATA\_DEFAULT';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F5: Failed Dropping function. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

select COUNT(1) into v\_temp from USER\_PROCEDURES where PROCEDURE\_NAME = 'F\_READ\_SEARCH\_CONDITION';

IF v\_temp > 0 THEN

v\_q := 'DROP PROCEDURE F\_READ\_SEARCH\_CONDITION';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR F5: Failed Dropping function. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

END IF;

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('DROP PON TABLES', CURRENT\_TIMESTAMP);

END;

/

DECLARE

V\_N PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_N FROM PON\_APP\_USERS WHERE USERKEY = 1 AND PASSWORD = '98-39-E1-73-F6-E2-C0-0E-66-7E-E2-1E-BB-C1-28-36';

IF V\_N = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');

DBMS\_OUTPUT.PUT\_LINE('A DEFAULT Pontis USER HAS BEEN ADDED WITH A PASSWORD OF ''pontis''.');

DBMS\_OUTPUT.PUT\_LINE('CHANGE THIS PASSWORD!');

DBMS\_OUTPUT.PUT\_LINE('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');

END IF;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Script finished on ' || SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') || '.');

END;

/

commit;

/

/\* --Find unwanted rows in DataDict that do not have corresponding rows in PON\_DICT.

SELECT DD.TABLE\_NAME, DD.COL\_NAME,

'INSERT TEMP\_DATADICT (TABLE\_NAME, COL\_NAME, COL\_ALIAS, V2CONVERT,

DATATYPE, WIDTH, DEC\_PLCS, NULL\_ALLOW,

UNIQUEKEY, POSITION, NBI\_CD, VALTYPE,

VALATTR1, VALATTR2, SYSFIELD, SYSDEFAULT,

KEYATTR1, UNIQUE\_FLD, HELPID, PAIRCODE,

CONVERSIONRULES, SNOTES, NOTES) VALUES (''' ||

DD.TABLE\_NAME || ''', ''' || DD.COL\_NAME || ''''

FROM DATADICT DD

LEFT OUTER JOIN PON\_DICT PD ON UPPER(PD.TABLE\_NAME) = UPPER(DD.TABLE\_NAME) AND UPPER(PD.COL\_NAME) = UPPER(DD.COL\_NAME)

WHERE PD.TABLE\_NAME IS NULL

ORDER BY DD.TABLE\_NAME, DD.COL\_NAME;

\*/

/\* Make inserts for DATADICT from PON\_DICT.

SELECT PD.TABLE\_NAME, PD.COL\_NAME, PD.COL\_ORDER, CASE WHEN EXISTS (SELECT 1

FROM DATADICT

WHERE UPPER(TABLE\_NAME) = UPPER(PD.TABLE\_NAME)

AND POSITION = PD.COL\_ORDER ) THEN 'Y' END AS CONFLICT,

'INSERT TEMP\_DATADICT (TABLE\_NAME, COL\_NAME, COL\_ALIAS, V2CONVERT, DATATYPE, WIDTH, DEC\_PLCS, NULL\_ALLOW, UNIQUEKEY, POSITION, NBI\_CD, VALTYPE, VALATTR1, VALATTR2, SYSFIELD, SYSDEFAULT, KEYATTR1, UNIQUE\_FLD, HELPID, PAIRCODE, CONVERSIONRULES, SNOTES, NOTES) VALUES (''' ||

PD.TABLE\_NAME || ''', ''' || PD.COL\_NAME || ''', ''' || PD.COL\_NAME || ''', ''-1'', ''' ||

PD.DATA\_TYPE || ''', ' ||

NVL(CAST(PD.LENGTH AS VARCHAR2(10)), 'NULL') || ', ' ||

NVL(CAST(PD.SCALE AS VARCHAR2(10)), 'NULL') || ', ' ||

CASE WHEN PD.REQUIRED = 1 THEN '''N'', ' ELSE '''Y'', ' END ||

CASE WHEN UPPER(PD.COL\_NAME) IN

(SELECT MAX(COLS.COLUMN\_NAME)

FROM USER\_CONSTRAINTS C

INNER JOIN USER\_CONS\_COLUMNS COLS

ON C.CONSTRAINT\_NAME = COLS.CONSTRAINT\_NAME

WHERE C.TABLE\_NAME = PD.TABLE\_NAME

AND C.CONSTRAINT\_TYPE IN ('P', 'U')

GROUP BY COLS.CONSTRAINT\_NAME

HAVING COUNT(\*) = 1)

THEN '''Y'', ' ELSE '''N'', ' END ||

PD.COL\_ORDER || ', ' ||

'''-1'', ''-1'', ' ||

'''-1'', ''-1'', ''1'', ''-1'', ' ||

'''N'', ''-1'', ''-1'', ''-1'', ' ||

'''-1'', ''-1'', ''-1'');'

FROM PON\_DICT PD

LEFT OUTER JOIN DATADICT DD ON UPPER(PD.TABLE\_NAME) = UPPER(DD.TABLE\_NAME) AND UPPER(PD.COL\_NAME) = UPPER(DD.COL\_NAME)

WHERE DD.TABLE\_NAME IS NULL

ORDER BY PD.TABLE\_NAME, PD.COL\_NAME;

\*/

/\*

Profiling

=========

--DROP TABLE PROFILING;

--DROP SEQUENCE S\_PROFILING;

CREATE TABLE PROFILING (

ID NUMBER(10) NOT NULL,

PLACE VARCHAR2(100) NOT NULL,

END\_TIME TIMESTAMP NOT NULL);

CREATE SEQUENCE S\_PROFILING;

CREATE OR REPLACE TRIGGER TRG\_PROFILING

BEFORE INSERT ON PROFILING

FOR EACH ROW

BEGIN

SELECT S\_PROFILING.NEXTVAL INTO :NEW.ID FROM DUAL;

END;

COLUMN PLACE FORMAT A30

COLUMN DURATION FORMAT A18

SELECT B.PLACE, TO\_CHAR(CAST((B.END\_TIME - A.END\_TIME) AS INTERVAL DAY TO SECOND(0)), 'HH:MM:SS') AS DURATION

FROM PROFILING A

INNER JOIN PROFILING B ON A.ID = B.ID -1

ORDER BY B.END\_TIME - A.END\_TIME DESC;

DELETE PROFILING;

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('SEED', CURRENT\_TIMESTAMP);

\*/

-- end part 5a Oracle