-- begin part 5 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;

BEGIN

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

END;

/

-- Users may keep custom names for the user tables ("USERBRDG" etc) in PON\_COPTIONS (or COPTIONS in older databases).

-- Replace "USERBRDG" in PON\_TABLE, PON\_DICT, PON\_FK, PON\_INDEX with the custom name from PON\_COPTIONS.

-- ALWAYS read PON\_COPTIONS.OPTIONAL and update PON\_COPTIONS\_T.

BEGIN

IF (TABLE\_EXISTS('PON\_COPTIONS') = 1) THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Update PON\_COPTIONS\_T with values from PON\_COPTIONS.');

EXECUTE IMMEDIATE 'MERGE INTO PON\_COPTIONS\_T T

USING (SELECT \* FROM PON\_COPTIONS WHERE OPTIONVAL <> ''-1'') S

ON (T.OPTIONNAME = S.OPTIONNAME)

WHEN MATCHED THEN UPDATE SET T.OPTIONVAL = S.OPTIONVAL';

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error updating PON\_COPTIONS\_T with values from PON\_COPTIONS. BrM options will be reset to default. ' || SQLERRM);

END;

ELSIF (TABLE\_EXISTS('COPTIONS') = 1) THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Update PON\_COPTIONS\_T with values from COPTIONS.');

EXECUTE IMMEDIATE 'MERGE INTO PON\_COPTIONS\_T T

USING (SELECT \* FROM COPTIONS WHERE OPTIONVAL <> ''-1'') S

ON (T.OPTIONNAME = S.OPTIONNAME)

WHEN MATCHED THEN UPDATE SET T.OPTIONVAL = S.OPTIONVAL';

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error updating PON\_COPTIONS\_T with values from PON\_COPTIONS. BrM options will be reset to default. ' || SQLERRM);

END;

END IF;

END;

/

ALTER TABLE PON\_DICT DISABLE CONSTRAINT PON\_DICT\_\_PON\_TABLE\_FK;

ALTER TABLE PON\_FK DISABLE CONSTRAINT PON\_FK\_\_PON\_DICT\_FK;

ALTER TABLE PON\_INDEX DISABLE CONSTRAINT PON\_INDEX\_\_PON\_DICT\_FK;

ALTER TABLE PON\_TABLE ADD ORIGINAL\_TABLE\_NAME VARCHAR2(30) NULL;

ALTER TABLE PON\_DICT ADD ORIGINAL\_TABLE\_NAME VARCHAR2(30) NULL;

ALTER TABLE PON\_FK ADD ORIGINAL\_TABLE\_NAME VARCHAR2(30) NULL;

ALTER TABLE PON\_INDEX ADD ORIGINAL\_TABLE\_NAME VARCHAR2(30) NULL;

UPDATE PON\_TABLE SET ORIGINAL\_TABLE\_NAME = TABLE\_NAME WHERE TABLE\_NAME IN ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT');

UPDATE PON\_DICT SET ORIGINAL\_TABLE\_NAME = TABLE\_NAME WHERE TABLE\_NAME IN ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT');

UPDATE PON\_FK SET ORIGINAL\_TABLE\_NAME = TABLE\_NAME WHERE TABLE\_NAME IN ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT');

UPDATE PON\_INDEX SET ORIGINAL\_TABLE\_NAME = TABLE\_NAME WHERE TABLE\_NAME IN ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT');

ALTER TABLE PON\_TABLE ADD CORE\_TABLE\_NAME VARCHAR2(30) NULL;

UPDATE PON\_TABLE

SET CORE\_TABLE\_NAME = CASE TABLE\_NAME

WHEN 'USERBRDG' THEN 'BRIDGE'

WHEN 'USERINSP' THEN 'INSPEVNT'

WHEN 'USERRWAY' THEN 'ROADWAY'

WHEN 'USERSTRUNIT' THEN 'STRUCTURE\_UNIT'

END

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

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

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

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

USING (SELECT

CASE UPPER(OPTIONNAME)

WHEN ''XTRNBRDGTABLE'' THEN ''USERBRDG''

WHEN ''XTRNINSPTABLE'' THEN ''USERINSP''

WHEN ''XTRNRWAYTABLE'' THEN ''USERRWAY''

WHEN ''XTRNSTRUNITTABLE'' THEN ''USERSTRUNIT''

ELSE OPTIONVAL

END TABLE\_NAME,

UPPER(OPTIONVAL) CUSTOM\_TABLE\_NAME

FROM PON\_COPTIONS\_T

WHERE UPPER(OPTIONNAME) IN (''XTRNBRDGTABLE'', ''XTRNINSPTABLE'', ''XTRNRWAYTABLE'', ''XTRNSTRUNITTABLE'')

) S

ON (T.ORIGINAL\_TABLE\_NAME = S.TABLE\_NAME)

WHEN MATCHED THEN UPDATE SET T.TABLE\_NAME = S.CUSTOM\_TABLE\_NAME'

FROM USER\_TABLES

WHERE TABLE\_NAME IN ('PON\_TABLE', 'PON\_DICT', 'PON\_FK', 'PON\_INDEX');

-- select \* from pon\_table where original\_table\_name in ('USERBRDG', 'USERINSP', 'USERRWAY', 'USERSTRUNIT') order by table\_name;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

ALTER TABLE PON\_DICT ENABLE CONSTRAINT PON\_DICT\_\_PON\_TABLE\_FK;

ALTER TABLE PON\_FK ENABLE CONSTRAINT PON\_FK\_\_PON\_DICT\_FK;

ALTER TABLE PON\_INDEX ENABLE CONSTRAINT PON\_INDEX\_\_PON\_DICT\_FK;

-- Change 'USERBRDG', 'USERINSP', 'USERRWAY', and 'USERSTRUNIT' in DATADICT to the table names in PON\_COPTIONS.

UPDATE DATADICT\_T DD

SET TABLE\_NAME = (SELECT LOWER(TABLE\_NAME) FROM PON\_DICT WHERE ORIGINAL\_TABLE\_NAME = DD.TABLE\_NAME AND COL\_NAME = DD.COL\_NAME)

WHERE (DD.TABLE\_NAME, DD.COL\_NAME) IN (SELECT ORIGINAL\_TABLE\_NAME, COL\_NAME FROM PON\_DICT WHERE TABLE\_NAME <> ORIGINAL\_TABLE\_NAME);

-- -- Set REQUIRED for old key columns to 0. Also done in part 3.

UPDATE PON\_DICT D

SET D.REQUIRED = 0

WHERE GET\_VAR('HAS\_GUIDS') = 1

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'

);

UPDATE PON\_DICT D

SET REPLACEMENT\_KEY = 1

WHERE GET\_VAR('HAS\_GUIDS') = '0'

AND (

PK = 2

OR EXISTS(

SELECT \*

FROM PON\_FK

WHERE PURPOSE = 2

AND (TABLE\_NAME = D.TABLE\_NAME AND COL\_NAME = D.COL\_NAME

OR R\_TABLE\_NAME = D.TABLE\_NAME AND R\_COL\_NAME = D.COL\_NAME

)

)

);

COMMIT;

-- Rename lookup tables ELCATDFS, MATDEFS, ELTYPDFS, ENVTDEFS.

ALTER SESSION SET NLS\_SORT=BINARY\_CI;

/\* Rename

ELCATDFS >> PON\_ELEM\_CAT\_DEFS

MATDEFS >> PON\_ELEM\_MAT\_DEFS

ELTYPDFS >> PON\_ELEM\_TYPE\_DEFS

ENVTDEFS >> PON\_ENVT\_DEFS

\*/

DECLARE

TYPE R\_CURSOR IS REF CURSOR;

CUR R\_CURSOR;

V\_Q VARCHAR2(1000);

V\_TN VARCHAR2(30);

V\_CN VARCHAR2(30);

V\_TYPE VARCHAR2(30);

V\_OLD\_TABLE\_EXISTS PLS\_INTEGER;

V\_NEW\_TABLE\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_OLD\_TABLE\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'ELCATDFS';

SELECT COUNT(\*) INTO V\_NEW\_TABLE\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'PON\_ELEM\_CAT\_DEFS';

IF V\_OLD\_TABLE\_EXISTS = 1 AND V\_NEW\_TABLE\_EXISTS = 0 THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Rename lookup tables ELCATDFS, MATDEFS, ELTYPDFS, ENVTDEFS.');

V\_Q := 'CREATE TABLE PON\_ELEM\_CAT\_DEFS AS SELECT \* FROM ELCATDFS';

EXECUTE IMMEDIATE V\_Q;

V\_Q := 'CREATE TABLE PON\_ELEM\_MAT\_DEFS AS SELECT \* FROM MATDEFS';

EXECUTE IMMEDIATE V\_Q;

V\_Q := 'CREATE TABLE PON\_ELEM\_TYPE\_DEFS AS SELECT \* FROM ELTYPDFS';

EXECUTE IMMEDIATE V\_Q;

V\_Q := 'CREATE TABLE PON\_ENVT\_DEFS AS SELECT \* FROM ENVTDEFS';

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR RENAMING TABLE: ' || V\_Q || '; ' || SQLERRM);

END;

END IF;

END;

/

-- Set force\_def to 1 if a column is added to a table.

UPDATE PON\_DICT D SET

FORCE\_DEF = 1,

NEW\_COL\_OLD\_TABLE = 1

WHERE EXISTS(SELECT \* FROM USER\_TABLES WHERE TABLE\_NAME = D.TABLE\_NAME)

AND NOT EXISTS(SELECT \* FROM USER\_TAB\_COLS WHERE TABLE\_NAME = D.TABLE\_NAME AND COLUMN\_NAME = D.COL\_NAME);

COMMIT;

-- ARCHIVE ITEMS BEFORE ALTERING SCHEMA

/\*

Archive PON\_ELEM\_INSP, PON\_ELEM\_DEFS, PON\_MRR\_ACTION\_DEFS, PON\_STATE\_DEFS as \*\_2011.

Archive if PON\_ELEM\_INSP does not contain ELEM\_GRANDPARENT\_KEY.

Drop PKs and FKs and rename.

\*/

DECLARE

TYPE R\_CURSOR IS REF CURSOR;

CUR R\_CURSOR;

V\_Q VARCHAR2(1000);

V\_TN VARCHAR2(30);

V\_CN VARCHAR2(30);

V\_TYPE VARCHAR2(30);

V\_PON\_ELEM\_INSP\_EXISTS PLS\_INTEGER;

V\_ELEM\_GRANDPARENT\_KEY\_EXISTS PLS\_INTEGER;

V\_ARCHIVE\_PON\_ELEM\_INSP VARCHAR2(20);

V\_ARCHIVED\_TABLE\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_PON\_ELEM\_INSP\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'PON\_ELEM\_INSP';

SELECT COUNT(\*) INTO V\_ELEM\_GRANDPARENT\_KEY\_EXISTS FROM USER\_TAB\_COLUMNS WHERE TABLE\_NAME = 'PON\_ELEM\_INSP' AND COLUMN\_NAME = 'ELEM\_GRANDPARENT\_KEY';

--ARCHIVE\_PON\_ELEM\_INSP will be DEFAULT, ALWAYS or NEVER.

SELECT VAR\_VALUE INTO V\_ARCHIVE\_PON\_ELEM\_INSP FROM PON\_GLOB\_VAR WHERE VARI = 'ARCHIVE\_PON\_ELEM\_INSP';

IF (V\_PON\_ELEM\_INSP\_EXISTS = 1)

AND (

(V\_ARCHIVE\_PON\_ELEM\_INSP = 'ALWAYS')

OR (V\_ARCHIVE\_PON\_ELEM\_INSP = 'DEFAULT' AND V\_ELEM\_GRANDPARENT\_KEY\_EXISTS = 0)

)

THEN

FOR I IN (

SELECT TRIGGER\_NAME

FROM USER\_TRIGGERS

WHERE TABLE\_NAME IN ('PON\_ELEM\_INSP', 'PON\_ELEM\_DEFS', 'PON\_MRR\_ACTION\_DEFS', 'PON\_STATE\_DEFS')

)

LOOP

V\_Q := 'DROP TRIGGER ' || I.TRIGGER\_NAME;

BEGIN

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DROPPING TRIGGER BEFORE ARCHIVING TABLE:' || V\_Q || ';' || SQLERRM);

END;

END LOOP;

OPEN CUR FOR SELECT TABLE\_NAME FROM USER\_TABLES WHERE TABLE\_NAME IN ('PON\_ELEM\_INSP', 'PON\_ELEM\_DEFS', 'PON\_MRR\_ACTION\_DEFS', 'PON\_STATE\_DEFS');

LOOP

FETCH CUR INTO V\_TN;

EXIT WHEN CUR%NOTFOUND;

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM USER\_TABLES WHERE TABLE\_NAME = ''' || V\_TN || '\_2011''' INTO V\_ARCHIVED\_TABLE\_EXISTS;

IF V\_ARCHIVED\_TABLE\_EXISTS = 1 THEN

BEGIN

V\_Q := 'DROP TABLE ' || V\_TN || '\_2011';

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DROPPING ARCHIVED TABLE: ' || V\_Q || '; ' || SQLERRM);

END;

END IF;

V\_Q := 'CREATE TABLE ' || V\_TN || '\_2011 AS SELECT \* FROM ' || V\_TN;

BEGIN

EXECUTE IMMEDIATE V\_Q;

DBMS\_OUTPUT.PUT\_LINE('Table ' || V\_TN || ' archived as ' || V\_TN || '\_2011.');

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR ARCHIVING TABLE: ' || V\_Q || '; ' || SQLERRM);

END;

V\_Q := 'DROP TABLE ' || V\_TN || ' CASCADE CONSTRAINTS PURGE';

BEGIN

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DROPPING TABLE AFTER ARCHIVING:' || V\_Q || ';' || SQLERRM);

END;

END LOOP;

CLOSE CUR;

END IF;

END;

/

-- END OF ARCHIVE ITEMS BEFORE ALTERING SCHEMA

-- Rename mixed case constraints.

ALTER SESSION SET NLS\_SORT=BINARY;

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

FOR X IN (

SELECT

'ALTER TABLE ' || TABLE\_NAME || ' RENAME CONSTRAINT "' || CONSTRAINT\_NAME ||'" TO ' || UPPER(CONSTRAINT\_NAME) Q

FROM USER\_CONSTRAINTS

WHERE CONSTRAINT\_NAME <> UPPER(CONSTRAINT\_NAME)

AND TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_TABLE)

)

LOOP

BEGIN

EXECUTE IMMEDIATE X.Q;

END;

END LOOP;

END;

/

ALTER SESSION SET NLS\_SORT=BINARY\_CI;

-- Drop FKs that are not as defined.

DECLARE

V\_TABLE\_NAME VARCHAR2(30);

V\_CONSTRAINT\_NAME VARCHAR2(30);

V\_CONSTRAINT\_TYPE VARCHAR2(30);

V\_Q VARCHAR2(300);

--Drop FKs.

CURSOR CONSTRAINTS\_TO\_DROP IS

SELECT DISTINCT TABLE\_NAME, CONSTRAINT\_NAME

FROM (

SELECT C.TABLE\_NAME, C.CONSTRAINT\_NAME, C.DELETE\_RULE, CC.COLUMN\_NAME COL\_NAME, CC.POSITION COL\_ORDER, COUNT(\*) OVER (PARTITION BY CC.CONSTRAINT\_NAME) COL\_COUNT

FROM USER\_CONSTRAINTS C

JOIN USER\_CONS\_COLUMNS CC ON C.CONSTRAINT\_NAME = CC.CONSTRAINT\_NAME

WHERE C.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_TABLE)

AND C.CONSTRAINT\_TYPE = 'R'

MINUS

SELECT TABLE\_NAME, FK\_NAME, CASE DELETE\_RULE WHEN 1 THEN 'CASCADE' ELSE 'NO ACTION' END, COL\_NAME, COL\_ORDER, COUNT(\*) OVER (PARTITION BY FK\_NAME)

FROM PON\_FK

WHERE GET\_VAR('HAS\_GUIDS') = 0 AND MOD(PURPOSE, 2) = 1 OR GET\_VAR('HAS\_GUIDS') = 1 AND PURPOSE > 1

);

BEGIN

OPEN CONSTRAINTS\_TO\_DROP;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME;

WHILE CONSTRAINTS\_TO\_DROP%FOUND

LOOP

BEGIN

-- Quote the constraint name because some are mixed case.

V\_Q := 'ALTER TABLE ' || V\_TABLE\_NAME || ' DROP CONSTRAINT "' || V\_CONSTRAINT\_NAME || '"';

--DBMS\_OUTPUT.PUT\_LINE(V\_Q);

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error dropping foreign key before adding foreign keys: ' || V\_Q || ': ' || SQLERRM);

END;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME;

END LOOP;

CLOSE CONSTRAINTS\_TO\_DROP;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error dropping foreign key before adding foreign keys.' || SQLERRM);

CLOSE CONSTRAINTS\_TO\_DROP;

END;

/

-- Drop PKs to be changed and the FKs that relate to them.

DECLARE

V\_TABLE\_NAME VARCHAR2(30);

V\_CONSTRAINT\_NAME VARCHAR2(30);

V\_CONSTRAINT\_TYPE VARCHAR2(30);

V\_Q VARCHAR2(300);

-- Drop PKs and FKs that relate to them.

-- Existing PK columns - design PK columns = Unwanted PK columns to be dropped.

CURSOR CONSTRAINTS\_TO\_DROP IS

WITH PKS\_TO\_DROP AS (

SELECT DISTINCT TABLE\_NAME, CONSTRAINT\_NAME

FROM (

-- Existing PK columns.

SELECT C.TABLE\_NAME, C.CONSTRAINT\_NAME, C.CONSTRAINT\_TYPE, CC.COLUMN\_NAME COL\_NAME, CC.POSITION COL\_ORDER, COUNT(\*) OVER (PARTITION BY CC.TABLE\_NAME) COL\_COUNT

FROM USER\_CONSTRAINTS C

JOIN USER\_CONS\_COLUMNS CC ON C.CONSTRAINT\_NAME = CC.CONSTRAINT\_NAME

WHERE CC.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_DICT)

AND C.CONSTRAINT\_TYPE = 'P'

MINUS

SELECT TABLE\_NAME, 'PK\_' || SUBSTR(TABLE\_NAME, 1, 27), 'P', COL\_NAME, COL\_ORDER, COUNT(\*) OVER (PARTITION BY TABLE\_NAME)

FROM PON\_DICT

WHERE ACTIVE\_PK = 1

)

)

SELECT TABLE\_NAME, CONSTRAINT\_NAME, 'P' CONSTRAINT\_TYPE

FROM PKS\_TO\_DROP

UNION

SELECT TABLE\_NAME, CONSTRAINT\_NAME, 'F'

FROM USER\_CONSTRAINTS

WHERE R\_CONSTRAINT\_NAME IN (SELECT CONSTRAINT\_NAME FROM PKS\_TO\_DROP)

ORDER BY CONSTRAINT\_TYPE;

BEGIN

OPEN CONSTRAINTS\_TO\_DROP;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME, V\_CONSTRAINT\_TYPE;

WHILE CONSTRAINTS\_TO\_DROP%FOUND

LOOP

BEGIN

V\_Q := 'ALTER TABLE ' || V\_TABLE\_NAME || ' DROP CONSTRAINT "' || V\_CONSTRAINT\_NAME || '"';

--DBMS\_OUTPUT.PUT\_LINE(V\_Q); --

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error dropping PKs to be changed and the FKs that relate to them: ' || V\_Q || ': ' || SQLERRM);

END;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME, V\_CONSTRAINT\_TYPE;

END LOOP;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Outer error dropping PKs to be changed and the FKs that relate to them' || SQLERRM);

END;

/

-- Drop indexes that are unwanted or changed.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Drop indexes that are unwanted or changed.');

-- Indexes to drop = Actual indexes - design indexes.

-- Indexes used by primary or unique constraint can not be dropped.

OPEN V\_CUR FOR

SELECT DISTINCT 'DROP INDEX ' || UI.INDEX\_NAME

FROM (

SELECT

UI.INDEX\_NAME, UI.TABLE\_NAME, UICOLS.COLUMN\_NAME, UICOLS.COLUMN\_POSITION,

UI.UNIQUENESS, SUM(1) OVER (PARTITION BY UICOLS.INDEX\_NAME) COLUMN\_COUNT

FROM USER\_INDEXES UI

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

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

LEFT OUTER JOIN USER\_CONSTRAINTS UCONS ON UCONS.CONSTRAINT\_TYPE IN ('P', 'U')

AND UCONS.INDEX\_NAME = UI.INDEX\_NAME

WHERE UCONS.CONSTRAINT\_NAME IS NULL

) UI

LEFT OUTER JOIN (

SELECT TABLE\_NAME,

COL\_NAME COLUMN\_NAME,

ROW\_NUMBER() OVER (PARTITION BY INDEX\_NAME ORDER BY COL\_ORDER) COLUMN\_POSITION,

CASE IS\_UNIQUE WHEN 1 THEN 'UNIQUE' ELSE 'NONUNIQUE' END UNIQUENESS,

COUNT(\*) OVER (PARTITION BY INDEX\_NAME) COLUMN\_COUNT

FROM PON\_INDEX

) I ON I.TABLE\_NAME = UI.TABLE\_NAME

AND I.UNIQUENESS = UI.UNIQUENESS

AND I.COLUMN\_NAME = UI.COLUMN\_NAME

AND I.COLUMN\_POSITION = UI.COLUMN\_POSITION

AND I.COLUMN\_COUNT = UI.COLUMN\_COUNT

WHERE I.TABLE\_NAME IS NULL;

REV(V\_CUR);

CLOSE V\_CUR;

END;

/

-- Reduce length or scale or change data type.

DECLARE

v\_FKC0 VARCHAR2(100);

v\_TN1 VARCHAR2(30);

v\_TN0 VARCHAR2(30);

v\_CN1 VARCHAR2(30);

v\_CN0 VARCHAR2(30);

v\_DT1 VARCHAR2(30);

v\_DT0 VARCHAR2(30);

v\_L1 VARCHAR2(10);

v\_L0 VARCHAR2(10);

v\_S1 VARCHAR2(2);

v\_S0 VARCHAR2(2);

v\_PK1 PLS\_INTEGER;

v\_FK1 VARCHAR2(150);

v\_FKT1 VARCHAR2(100);

v\_FKC1 VARCHAR2(100);

V\_REQUIRED0 PLS\_INTEGER;

V\_REQUIRED1 PLS\_INTEGER;

v\_DV0 VARCHAR2(30);

v\_DV1 VARCHAR2(30);

v\_CO VARCHAR2(3);

v\_q VARCHAR2(2000);

v\_temp number :=0;

v\_temp2 VARCHAR2(100);

V\_LENGTHINESS1 VARCHAR2(20);

V\_LENGTHINESS0 VARCHAR2(20);

V\_DIFFERENCE VARCHAR2(20);

V\_COL\_ORDER PLS\_INTEGER;

V\_X PLS\_INTEGER;

V\_STEPS VARCHAR2(999);

BEGIN

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

DECLARE

CURSOR curColData IS

SELECT \*

FROM (

SELECT

D.TABLE\_NAME,

D.COL\_NAME,

D.DATA\_TYPE,

UCOLS.DATA\_TYPE DATA\_TYPE0,

D.LENGTH,

UCOLS.DATA\_LENGTH LENGTH0,

D.SCALE,

UCOLS.DATA\_SCALE SCALE0,

CASE

WHEN D.REPLACEMENT\_KEY = 1 THEN 0

ELSE D.REQUIRED

END REQUIRED,

CASE UCOLS.NULLABLE

WHEN 'Y' THEN 0

WHEN 'N' THEN 1

ELSE NULL

END REQUIRED0,

D.DEF\_VALUE,

UCOLS.DATA\_DEFAULT DEV\_VALUE0,

CASE

WHEN D.SCALE IS NOT NULL THEN '(' || D.LENGTH || ',' || D.SCALE || ')'

WHEN D.LENGTH IS NOT NULL THEN '(' || D.LENGTH || ')'

END AS LENGTHINESS,

CASE

WHEN UCOLS.DATA\_TYPE IN ('CHAR', 'NCHAR', 'VARCHAR2', 'NVARCHAR2') THEN '(' || UCOLS.DATA\_LENGTH || ')'

WHEN UCOLS.DATA\_SCALE IS NOT NULL THEN '(' || UCOLS.DATA\_PRECISION || ',' || UCOLS.DATA\_SCALE || ')'

WHEN UCOLS.DATA\_LENGTH IS NOT NULL THEN '(' || UCOLS.DATA\_PRECISION || ')'

END AS LENGTHINESS0,

CASE

-- UCOLS.DATA\_TYPE may be TIMESTAMP(6)

WHEN D.DATA\_TYPE = 'DATE' AND UCOLS.DATA\_TYPE LIKE 'TIMESTAMP%' THEN 'NO\_DIFFERENCE'

WHEN D.DATA\_TYPE IN ('VARCHAR2', 'CHAR') AND UCOLS.DATA\_TYPE IN ('VARCHAR2', 'CHAR') THEN 'NO\_DIFFERENCE'

WHEN D.DATA\_TYPE <> UCOLS.DATA\_TYPE THEN 'D0'

WHEN D.DATA\_TYPE = 'FLOAT' AND COALESCE(D.LENGTH, 126) < COALESCE(UCOLS.DATA\_PRECISION, 126) THEN 'D1'

WHEN D.DATA\_TYPE = 'NUMBER' AND (

COALESCE(D.LENGTH, 38) < COALESCE(UCOLS.DATA\_PRECISION, 38) OR

COALESCE(D.SCALE, 0) < COALESCE(UCOLS.DATA\_SCALE, 0)

) THEN 'D3'

ELSE 'NO\_DIFFERENCE'

END DIFFERENCE,

D.COL\_ORDER

FROM PON\_DICT D

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

) T

WHERE T.DIFFERENCE <> 'NO\_DIFFERENCE'

ORDER BY T.TABLE\_NAME, T.COL\_ORDER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Reduce length or scale or change data type.');

OPEN curColData;

FETCH curColData INTO v\_TN1, v\_CN1, v\_DT1, v\_DT0, v\_L1, v\_L0, v\_S1, v\_S0, V\_REQUIRED1, V\_REQUIRED0, v\_DV1, v\_DV0, V\_LENGTHINESS1, V\_LENGTHINESS0, V\_DIFFERENCE, V\_COL\_ORDER;

WHILE (curColData%FOUND)

LOOP

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Old: ' || V\_TN1 || ' ' || V\_CN1 || ' ' || V\_DT0 || V\_LENGTHINESS0 || ' -- ' || V\_DIFFERENCE);

DBMS\_OUTPUT.PUT\_LINE('New: ' || V\_TN1 || ' ' || V\_CN1 || ' ' || V\_DT1 || V\_LENGTHINESS1);

V\_STEPS := '';

-- WIP Remove output from lines below.

--DBMS\_OUTPUT.PUT\_LINE('v\_TN1: ' || v\_TN1 || ', v\_CN1: ' || V\_CN1 ||

-- ', v\_DT1: ' || V\_DT1 || ', v\_DT0: ' || V\_DT0 ||

-- ', v\_L1: ' || V\_L1 || ', v\_L0: ' || V\_L0 ||

-- ', v\_S1: ' || V\_S1 || ', V\_S0: ' || v\_S0 ||

-- ', V\_REQUIRED1: ' || V\_REQUIRED1 || ', V\_REQUIRED0: ' || V\_REQUIRED0 ||

-- ', v\_DV1: ' || V\_DV1 || ', v\_DV0: ' || V\_DV0 ||

-- ', V\_LENGTHINESS1: ' || V\_LENGTHINESS1);

--Drop TEMP\_COL if it exists.

V\_X := 1;

select COUNT(1) into v\_temp

from user\_tab\_cols

where TABLE\_NAME = UPPER(v\_TN1) AND COLUMN\_NAME = 'TEMP\_COL';

IF V\_TEMP = 1 THEN

V\_TEMP2 := 'ALTER TABLE ' || V\_TN1 || ' DROP COLUMN TEMP\_COL';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

BEGIN

EXECUTE IMMEDIATE V\_TEMP2;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error 1 Reduce length or scale or change data type: ' || V\_TEMP2 || ' ' || SQLERRM);

END;

END IF;

V\_X := 2;

V\_TEMP2 := 'ALTER TABLE ' || V\_TN1 || ' ADD TEMP\_COL VARCHAR2(300)';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

EXECUTE IMMEDIATE V\_TEMP2;

V\_X := 3;

V\_TEMP2 := 'UPDATE ' || V\_TN1 || ' SET TEMP\_COL = ' || V\_CN1;

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

EXECUTE IMMEDIATE V\_TEMP2;

-- The column to be altered must be empty, so it must be nullable.

V\_X := 4;

IF V\_REQUIRED0 = 1 THEN

V\_TEMP2 := 'ALTER TABLE "' || v\_TN1 || '" MODIFY ("' || v\_CN1 || '" NULL)';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

execute immediate V\_TEMP2;

END IF;

V\_X := 5;

V\_TEMP2 := 'UPDATE ' || V\_TN1 || ' SET ' || V\_CN1 || ' = NULL';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

EXECUTE IMMEDIATE V\_TEMP2;

V\_X := 6;

V\_TEMP2 := 'ALTER TABLE "' || v\_TN1 || '" MODIFY ("' || v\_CN1 || '" ' || V\_DT1 || V\_LENGTHINESS1 || ')';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

execute immediate V\_TEMP2;

V\_X := 7;

V\_TEMP2 := 'UPDATE ' || V\_TN1 || ' SET ' || V\_CN1 || ' = CAST(TEMP\_COL AS ' || V\_DT1 || V\_LENGTHINESS1 || ')';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

EXECUTE IMMEDIATE V\_TEMP2;

V\_X := 8;

V\_TEMP2 := 'ALTER TABLE ' || V\_TN1 || ' DROP COLUMN TEMP\_COL';

V\_STEPS := V\_STEPS || CHR(13) || CHR(10) || V\_X || ': ' || V\_TEMP2;

EXECUTE IMMEDIATE V\_TEMP2;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D3.2: Reduce length or scale or change data type: ');

DBMS\_OUTPUT.PUT\_LINE(V\_STEPS);

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

FETCH curColData INTO v\_TN1, v\_CN1, v\_DT1, v\_DT0, v\_L1, v\_L0, v\_S1, v\_S0, V\_REQUIRED1, V\_REQUIRED0, v\_DV1, v\_DV0, V\_LENGTHINESS1, V\_LENGTHINESS0, V\_DIFFERENCE, V\_COL\_ORDER;

END LOOP;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR at end of reduce length or scale or change data type. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

END;

/

-- Alter columns.

-- This code can make existing columns nullable. Code to make columns not nullable is in part 5a "Make some non PK columns required.".

DECLARE

v\_FKC0 VARCHAR2(100);

v\_TN1 VARCHAR2(30);

v\_TN0 VARCHAR2(30);

v\_CN1 VARCHAR2(30);

v\_CN0 VARCHAR2(30);

v\_DT1 VARCHAR2(30);

v\_DT0 VARCHAR2(30);

v\_L1 VARCHAR2(10);

v\_L0 VARCHAR2(10);

v\_S1 VARCHAR2(2);

v\_S0 VARCHAR2(2);

v\_PK1 PLS\_INTEGER;

v\_FK1 VARCHAR2(150);

v\_FKT1 VARCHAR2(100);

v\_FKC1 VARCHAR2(100);

V\_REQUIRED0 PLS\_INTEGER;

V\_REQUIRED1 PLS\_INTEGER;

v\_DV0 VARCHAR2(30);

v\_DV1 VARCHAR2(30);

V\_COL\_ORDER PLS\_INTEGER;

V\_DIFFERENCE VARCHAR2(20);

v\_CO VARCHAR2(3);

v\_q VARCHAR2(2000);

v\_temp number :=0;

v\_temp2 VARCHAR2(100);

V\_LENGTHINESS1 VARCHAR2(20);

BEGIN

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

DECLARE

CURSOR curColData IS

SELECT \*

FROM (

SELECT

D.TABLE\_NAME,

D.COL\_NAME,

D.DATA\_TYPE,

C.DATA\_TYPE DATA\_TYPE0,

D.LENGTH,

C.DATA\_LENGTH LENGTH0,

D.SCALE,

C.DATA\_SCALE SCALE0,

CASE

WHEN D.REPLACEMENT\_KEY = 1

THEN 0

ELSE D.REQUIRED

END REQUIRED,

CASE C.NULLABLE

WHEN 'Y' THEN 0

WHEN 'N' THEN 1

ELSE NULL

END REQUIRED0,

D.DEF\_VALUE,

TRIM(F\_READ\_DATA\_DEFAULT(D.TABLE\_NAME, D.COL\_NAME)) DEF\_VALUE0,

CASE WHEN D.SCALE IS NOT NULL THEN D.LENGTH || ',' || D.SCALE ELSE CAST(D.LENGTH AS VARCHAR2(20)) END AS LENGTHINESS,

D.COL\_ORDER,

CASE

WHEN D.DATA\_TYPE <> C.DATA\_TYPE THEN 'D1'

WHEN D.REQUIRED = 0 AND C.NULLABLE = 'N' THEN 'D2'

WHEN D.DATA\_TYPE IN ('CHAR', 'VARCHAR2', 'NVARCHAR2') AND COALESCE(D.LENGTH, 1) <> COALESCE(C.CHAR\_LENGTH, 1) THEN 'D3'

WHEN D.DATA\_TYPE = 'NUMBER' AND COALESCE(D.LENGTH, 38) <> COALESCE(C.DATA\_PRECISION, 38) OR COALESCE(D.SCALE, 0) <> COALESCE(C.DATA\_SCALE, 0) THEN 'D4'

WHEN D.DATA\_TYPE = 'FLOAT' AND COALESCE(D.LENGTH, 126) <> COALESCE(C.DATA\_PRECISION, 126) THEN 'D5'

WHEN D.DATA\_TYPE IN ('DATE') AND D.DATA\_TYPE <> C.DATA\_TYPE THEN 'D6'

WHEN

D.DEF\_VALUE IS NOT NULL

AND F\_READ\_DATA\_DEFAULT(D.TABLE\_NAME, D.COL\_NAME) IS NOT NULL

AND '(' || D.DEF\_VALUE || ')' <> TRIM(F\_READ\_DATA\_DEFAULT(D.TABLE\_NAME, D.COL\_NAME))

AND D.DEF\_VALUE <> TRIM(F\_READ\_DATA\_DEFAULT(D.TABLE\_NAME, D.COL\_NAME)) THEN 'D8'

ELSE 'NO\_DIFFERENCE'

END DIFFERENCE

FROM PON\_DICT D

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

) T

WHERE DIFFERENCE LIKE 'D%'

ORDER BY TABLE\_NAME, COL\_ORDER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Alter Columns.');

OPEN curColData;

FETCH curColData INTO v\_TN1, v\_CN1, v\_DT1, v\_DT0, v\_L1, v\_L0, v\_S1, v\_S0, V\_REQUIRED1, V\_REQUIRED0, v\_DV1, v\_DV0, V\_LENGTHINESS1, V\_COL\_ORDER, V\_DIFFERENCE;

WHILE (curColData%FOUND)

LOOP

BEGIN

v\_q := '';

-- D.3.9 | Check the Data Type

IF v\_L1 IS NOT NULL THEN

IF v\_S1 IS NOT NULL THEN

v\_q := v\_DT1 || '(' || v\_L1 || ',' || v\_S1 || ') ' ;

ELSE

v\_q := v\_DT1 || '(' || v\_L1 || ') ' ;

END IF;

END IF;

IF V\_REQUIRED1 = 0 AND V\_REQUIRED0 = 1 THEN

V\_Q := V\_Q || ' NULL';

END IF;

v\_q := 'ALTER TABLE "' || v\_TN1 || '" MODIFY "' || v\_CN1 || '" ' || v\_q ;

DBMS\_OUTPUT.PUT\_LINE(V\_Q || ' -- ' || V\_DIFFERENCE);

execute immediate v\_q;

/\* D.3.13 | Add, change and drop default values.

A new Oracle column without a default does not have a default clause.

(user\_tab\_cols.data\_default is null)

Once a column has a default clause it can not be removed. But the default can

be set to NULL.

(user\_tab\_cols.data\_default = 'NULL')

Despite this the Oracle is very simple.

Tested twice with each of v\_tn1 = 'PON\_SESSION' and vcn1 = 'STARTED\_ON' and v\_dv1 =

null, 'NULL', 'SYSDATE', 'SYSDATE+1', NULL, 'NULL'.

\*/

if UPPER(nvl(v\_dv0,'NULL')) != UPPER(nvl(v\_dv1,'NULL')) then

v\_q := 'alter table ' || v\_tn1 || ' modify ' || v\_cn1 || ' default ' || nvl(v\_dv1, 'NULL');

begin

-- wip DBMS\_OUTPUT.PUT\_LINE(V\_Q || ' -- ' || V\_DIFFERENCE); -- wip

execute immediate v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D3.0: Altering default failed [' || v\_q || ']; ' || SQLERRM);

end;

end if;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D3.1: Altering column failed [' || v\_q || ']; ' || SQLERRM);

END;

FETCH curColData INTO v\_TN1, v\_CN1, v\_DT1, v\_DT0, v\_L1, v\_L0, v\_S1, v\_S0, V\_REQUIRED1, V\_REQUIRED0, v\_DV1, v\_DV0, V\_LENGTHINESS1, V\_COL\_ORDER, V\_DIFFERENCE;

END LOOP;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR at end of alter columns [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

END;

/

-- Add missing columns. Do not make them required because they are added to existing tables.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'ALTER TABLE ' || D.TABLE\_NAME || ' ADD (' ||

D.COL\_NAME || ' ' || D.DATA\_TYPE ||

CASE WHEN D.SCALE IS NOT NULL THEN '(' || D.LENGTH || ',' || D.SCALE || ')'

WHEN D.LENGTH IS NOT NULL THEN '(' || D.LENGTH || ')'

END || ' ' ||

CASE WHEN D.DEF\_VALUE IS NOT NULL THEN 'DEFAULT (' || D.DEF\_VALUE || ') ' END ||

')'

FROM PON\_DICT D

LEFT OUTER JOIN USER\_TAB\_COLS C ON C.TABLE\_NAME = D.TABLE\_NAME AND C.COLUMN\_NAME = D.COL\_NAME

LEFT OUTER JOIN USER\_TABLES UT ON UT.TABLE\_NAME = D.TABLE\_NAME

WHERE UT.TABLE\_NAME IS NOT NULL AND C.TABLE\_NAME IS NULL AND GET\_VAR('TSC') = 1

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

REV(V\_CUR);

END;

/

-- Add missing tables.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'CREATE TABLE ' || TABLE\_NAME || ' (' ||

PON\_CONCAT(

CURSOR(

SELECT

COL\_NAME || ' ' || DATA\_TYPE ||

CASE WHEN SCALE IS NOT NULL THEN '(' || LENGTH || ',' || SCALE || ')'

WHEN LENGTH IS NOT NULL THEN '(' || LENGTH || ')'

END || ' ' ||

CASE WHEN DEF\_VALUE IS NOT NULL THEN 'DEFAULT (' || DEF\_VALUE || ') ' END ||

CASE

WHEN NEW\_COL\_OLD\_TABLE = 1 OR REPLACEMENT\_KEY = 1 OR REQUIRED = 0 THEN 'NULL'

ELSE 'NOT NULL'

END

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

ORDER BY COL\_ORDER

),

', '

) ||

')'

FROM PON\_TABLE T

WHERE TABLE\_NAME NOT IN (SELECT TABLE\_NAME FROM USER\_TABLES) AND GET\_VAR('TSC') = 1

ORDER BY TABLE\_NAME;

REV(V\_CUR);

END;

/

-- Make new columns on old tables required. Columns must have a default value.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

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

DBMS\_OUTPUT.PUT\_LINE('Update new columns with default values.');

OPEN V\_CUR FOR

SELECT 'UPDATE ' || TABLE\_NAME || ' SET ' || COL\_NAME || ' = ' || DEF\_VALUE

FROM PON\_DICT

WHERE NEW\_COL\_OLD\_TABLE = 1

AND REPLACEMENT\_KEY = 0

AND DEF\_VALUE IS NOT NULL;

REV(V\_CUR);

DBMS\_OUTPUT.PUT\_LINE('Make new required columns on old tables required.');

OPEN V\_CUR FOR

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

CASE

WHEN LENGTH IS NOT NULL AND SCALE IS NOT NULL THEN '(' || LENGTH || ', ' || SCALE || ') '

WHEN LENGTH IS NOT NULL THEN '(' || LENGTH || ') '

WHEN SCALE IS NOT NULL THEN '(' || SCALE || ') '

ELSE ' '

END

|| 'NOT NULL'

FROM PON\_DICT

WHERE REQUIRED = 1

AND NEW\_COL\_OLD\_TABLE = 1

AND REPLACEMENT\_KEY = 0

AND DEF\_VALUE IS NOT NULL;

REV(V\_CUR);

CLOSE V\_CUR;

END IF;

END;

/

-- D.7.0 | Oracle Identity triggers

DECLARE

V\_TN1 VARCHAR2(30);

V\_CN1 VARCHAR2(30);

V\_SEQUENCE\_NAME VARCHAR2(30);

V\_CACHE\_SIZE NUMBER :=0;

V\_TRIGGER\_NAME VARCHAR2(30);

V\_NEXTVAL INTEGER :=0;

V\_Q VARCHAR2(32767);

V\_TEMP VARCHAR2(100);

CURSOR curColData IS SELECT DD.TABLE\_NAME, DD.COL\_NAME, SEQU.SEQUENCE\_NAME, SEQU.CACHE\_SIZE, TRIG.TRIGGER\_NAME

FROM (SELECT TABLE\_NAME, COL\_NAME, ID

FROM PON\_DICT

WHERE ID = 1) DD

LEFT OUTER JOIN USER\_SEQUENCES SEQU ON SEQU.SEQUENCE\_NAME = 'S\_' || DD.TABLE\_NAME

LEFT OUTER JOIN USER\_TRIGGERS TRIG ON TRIG.TRIGGER\_NAME = 'TUB\_' || DD.TABLE\_NAME;

BEGIN

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

IF v\_temp = 1 THEN

OPEN curColData;

FETCH curColData INTO v\_TN1, v\_CN1, V\_SEQUENCE\_NAME, V\_CACHE\_SIZE, V\_TRIGGER\_NAME;

WHILE (curColData%FOUND) LOOP

BEGIN

V\_Q := 'SELECT NVL(MAX(' || V\_CN1 || '),0) + 1 FROM ' || V\_TN1;

EXECUTE IMMEDIATE V\_Q INTO V\_NEXTVAL;

IF V\_CACHE\_SIZE IS NOT NULL AND V\_CACHE\_SIZE <> 0 THEN

BEGIN

V\_Q := 'DROP SEQUENCE S\_' || V\_TN1;

BEGIN

--Drop the sequence in its own block so it is dropped before

--it is recreated.

EXECUTE IMMEDIATE V\_Q;

END;

V\_Q := 'CREATE SEQUENCE S\_' || V\_TN1 || ' INCREMENT BY 1 START WITH ' || V\_NEXTVAL || ' NOMAXVALUE NOMINVALUE NOCYCLE NOORDER NOCACHE' ;

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D7: Failed Modifying Sequence [' || v\_q || ']; ' || SQLERRM);

END;

ELSIF V\_SEQUENCE\_NAME IS NULL THEN

V\_Q := 'CREATE SEQUENCE S\_' || V\_TN1 || ' INCREMENT BY 1 START WITH ' || V\_NEXTVAL || ' NOMAXVALUE NOMINVALUE NOCYCLE NOORDER NOCACHE';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D7: Failed Creating Sequence [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

V\_Q := 'CREATE OR REPLACE TRIGGER TUB\_' || V\_TN1 || '

BEFORE INSERT ON ' || V\_TN1 || ' FOR EACH ROW

BEGIN

SELECT CASE WHEN :NEW.' || V\_CN1 || ' IS NULL THEN S\_' || V\_TN1 || '.NEXTVAL ELSE :NEW.' || V\_CN1 || ' END INTO :NEW.' || V\_CN1 || ' FROM DUAL;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL;

WHEN OTHERS THEN

RAISE;

END;';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D7: Failed Creating Trigger [' || v\_q || ']; ' || SQLERRM);

END;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR processing triggers: ' || SQLERRM );

DBMS\_OUTPUT.PUT\_LINE(DBMS\_UTILITY.FORMAT\_ERROR\_BACKTRACE);

END;

FETCH curColData INTO v\_TN1, v\_CN1, V\_SEQUENCE\_NAME, V\_CACHE\_SIZE, V\_TRIGGER\_NAME;

END LOOP;

END IF;

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

END;

/

-- REMOVE ITEMS FROM PRODUCTION DATABASES.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('REMOVE ITEMS FROM PRODUCTION DATABASES.');

END;

/

/\* This block is for code that removes data or pieces of schema from databases that are

being used by customers.

Use EXECUTE IMMEDIATE to hide your SQL statements. The statements will be

parsed even if they are inside a PL/SQL IF statement. A SELECT statement within

an IF will cause an error if the tables it refers to do not exist yet.

Finish each DECLARE block with:

END;

/

This code needs to be run twice to test it properly.

\*/

-- 11/23/2014 Andrew Bloss. Delete 8 rows in PON\_BENEFIT\_ELEMENTS that do not have a matching ELEM\_KEY in PON\_ELEM\_DEFS. These orphans were in the default data.

DELETE PON\_BENEFIT\_ELEMENTS WHERE ELEM\_KEY = '308' AND NOT EXISTS(SELECT \* FROM PON\_ELEM\_DEFS WHERE ELEM\_KEY = '308');

--Remove bad entries from PON\_NAV\_CONTROL\_GROUP

DECLARE

V\_X PLS\_INTEGER;

BEGIN

SELECT COUNT(1) into V\_X FROM PON\_NAV\_CONTROL\_GROUP WHERE

(TAB\_ID = 4) AND (TASK\_ID = 18) AND (CONTROL\_GROUP\_ID = 3) AND

(NAME = 'Validation Results') AND

(DESKTOP\_URL = '/PontisModules/shared/UiValidationResultsGrid.ascx');

IF V\_X > 0 THEN

DELETE FROM PON\_APP\_CONTROL\_GROUP\_SECURITY WHERE (TAB\_ID = 4) AND (TASK\_ID = 18) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_APP\_CONTROL\_SECURITY WHERE (TAB\_ID = 4) AND (TASK\_ID = 18) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_NAV\_CONTROL WHERE (TAB\_ID = 4) AND (TASK\_ID = 18) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_NAV\_CONTROL\_GROUP WHERE (TAB\_ID = 4) AND (TASK\_ID = 18) AND (CONTROL\_GROUP\_ID = 3);

END IF;

END;

/

DECLARE

V\_X PLS\_INTEGER;

BEGIN

SELECT COUNT(1) into V\_X FROM PON\_NAV\_CONTROL\_GROUP WHERE

(TAB\_ID = 7) AND (TASK\_ID = 7) AND (CONTROL\_GROUP\_ID = 3) AND

(NAME = 'Validation Results') AND

(DESKTOP\_URL = '/PontisModules/shared/UiValidationResultsGrid.ascx');

IF V\_X > 0 THEN

DELETE FROM PON\_APP\_CONTROL\_GROUP\_SECURITY WHERE (TAB\_ID = 7) AND (TASK\_ID = 7) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_APP\_CONTROL\_SECURITY WHERE (TAB\_ID = 7) AND (TASK\_ID = 7) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_NAV\_CONTROL WHERE (TAB\_ID = 7) AND (TASK\_ID = 7) AND (CONTROL\_GROUP\_ID = 3);

DELETE FROM PON\_NAV\_CONTROL\_GROUP WHERE (TAB\_ID = 7) AND (TASK\_ID = 7) AND (CONTROL\_GROUP\_ID = 3);

END IF;

END;

/

-- Drop unique indexes that cause the merges to error.

DECLARE

V\_Q VARCHAR2(1000);

BEGIN

FOR I IN (

SELECT INDEX\_NAME FROM USER\_INDEXES WHERE INDEX\_NAME IN (

'IX\_ACCESSFILTER' --Kentucky pre 52, unique index causes trouble with the merge.

,'IX\_FILTER\_NAME' --Kentucky, Wisconsin, California pre 52, unique index causes trouble with the merge.

,'IX\_LAYOUT\_NAME' --Kentucky, Wisconsin, CA pre 52, unique index causes trouble with the merge.

,'XAK1PON\_EXCHANGE\_OPTION' --Remove because it is a unique index on a non primary key on PON\_EXCHANGE\_OPTION, and the index below creates a merge error.

,'XAK2PON\_EXCHANGE\_OPTION' --Oklahoma\_512, unique index causes trouble with the merge.

,'XAK1PON\_NAV\_CONTROL\_GROUP'

,'XAK1PON\_NAV\_CONTROL'

,'XAK1PON\_NAV\_TASK'

)

)

LOOP

BEGIN

V\_Q := 'DROP INDEX ' || I.INDEX\_NAME;

--DBMS\_OUTPUT.PUT\_LINE(V\_Q);

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR 14: Dropping index[ ' || V\_Q || '; ' || SQLERRM );

END;

END LOOP;

END;

/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=25 AND TAB\_ID=7 AND NAME='Edit Layout SQL' AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=25 AND TAB\_ID=7;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=25 AND TAB\_ID=7;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=25 AND TAB\_ID=7;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=25 AND TAB\_ID=7 AND NAME='Edit Layout SQL';

end if;

end;

/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=26 AND TAB\_ID=7 AND NAME='Create New Layout' AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=26 AND TAB\_ID=7;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=26 AND TAB\_ID=7;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=26 AND TAB\_ID=7;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=26 AND TAB\_ID=7 AND NAME='Create New Layout';

end if;

end;

/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=29 AND TAB\_ID=7 AND NAME='Create New Filter' AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=29 AND TAB\_ID=7;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=29 AND TAB\_ID=7;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=29 AND TAB\_ID=7;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=29 AND TAB\_ID=7 AND NAME='Create New Filter';

end if;

end;

/

/\* Deleting 'Admin>Actions' task (tab\_id=11, task\_id=5), its children tasks (tab\_id=11, task\_id = 13 and 14) and related records.

-- Deleting first task children, to prevent violation of foreign key constraint.

-- Deletion is done to refresh database with the new task configuration.

-- Name of 'Admin>Actions>Flex Actions' is changed to 'Action Defs' and its parent task is changed to 'Admin>Definitions'.

-- 'Admin>Actions' and 'Admin>Actions>Action Type Defs' are made invisible.

\*/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=13 AND TAB\_ID=11 AND NAME='Action Type Defs' AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=13 AND TAB\_ID=11;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=13 AND TAB\_ID=11;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=13 AND TAB\_ID=11;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=13 AND TAB\_ID=11 AND NAME='Action Type Defs';

end if;

end;

/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=14 AND TAB\_ID=11 AND (NAME='Action Defs' OR NAME='Flex Actions') AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=14 AND TAB\_ID=11;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=14 AND TAB\_ID=11;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=14 AND TAB\_ID=11;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=14 AND TAB\_ID=11 AND (NAME='Action Defs' OR NAME='Flex Actions');

end if;

end;

/

declare

c int;

begin

select count(\*) into c from PON\_NAV\_TASK WHERE TASK\_ID=5 AND TAB\_ID=11 AND NAME='Actions' AND GET\_VAR('HAS\_GUIDS') = 0;

if c = 1 AND GET\_VAR('HAS\_GUIDS') = 0 then

DELETE PON\_NAV\_CONTROL WHERE TASK\_ID=5 AND TAB\_ID=11;

DELETE PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=5 AND TAB\_ID=11;

DELETE PON\_APP\_TASK\_SECURITY WHERE TASK\_ID=5 AND TAB\_ID=11;

DELETE PON\_NAV\_TASK WHERE TASK\_ID=5 AND TAB\_ID=11 AND NAME='Actions';

end if;

end;

/

-- Drop FKs on retired tables

DECLARE

V\_TABLE\_NAME VARCHAR2(30);

V\_CONSTRAINT\_NAME VARCHAR2(30);

V\_CONSTRAINT\_TYPE VARCHAR2(30);

V\_Q VARCHAR2(300);

--Drop FKs.

CURSOR CONSTRAINTS\_TO\_DROP IS

SELECT DISTINCT C.TABLE\_NAME, C.CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS C

JOIN USER\_CONSTRAINTS RC ON RC.CONSTRAINT\_NAME = C.R\_CONSTRAINT\_NAME

WHERE

(

C.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_RETIRED\_TABLE)

AND

RC.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_TABLE)

OR

C.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_TABLE)

AND

RC.TABLE\_NAME IN (SELECT TABLE\_NAME FROM PON\_RETIRED\_TABLE)

)

AND C.CONSTRAINT\_TYPE = 'R';

BEGIN

OPEN CONSTRAINTS\_TO\_DROP;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME;

WHILE CONSTRAINTS\_TO\_DROP%FOUND

LOOP

BEGIN

V\_Q := 'ALTER TABLE ' || V\_TABLE\_NAME || ' DROP CONSTRAINT ' || V\_CONSTRAINT\_NAME;

--DBMS\_OUTPUT.PUT\_LINE(V\_Q);

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error dropping foreign key on retired table: ' || V\_Q || ': ' || SQLERRM);

END;

FETCH CONSTRAINTS\_TO\_DROP INTO V\_TABLE\_NAME, V\_CONSTRAINT\_NAME;

END LOOP;

CLOSE CONSTRAINTS\_TO\_DROP;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error dropping foreign key on retired table final: ' || SQLERRM);

CLOSE CONSTRAINTS\_TO\_DROP;

END;

/

--================== END OF REMOVE ITEMS FROM PRODUCTION DATABASES ========

-- Grant permissions to tables to PONTISUSER.

DECLARE

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

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

BEGIN

SELECT VAR\_VALUE INTO V\_PONTISUSER\_PERMISSIONS FROM PON\_GLOB\_VAR WHERE VARI = 'PUP';

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

IF V\_PONTISUSER\_EXISTS = 1 AND V\_PONTISUSER\_PERMISSIONS = 1 THEN

DECLARE

V\_Q VARCHAR2(300);

V\_TN1 VARCHAR2(30);

CURSOR CUR IS SELECT DISTINCT TABLE\_NAME FROM PON\_DICT;

BEGIN

OPEN CUR;

LOOP

FETCH CUR INTO V\_TN1;

EXIT WHEN CUR%NOTFOUND;

V\_Q := 'GRANT SELECT, UPDATE, INSERT, DELETE ON ' || V\_TN1 || ' TO PONTISUSER';

BEGIN

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D2: Granting permissions on Table [' || v\_q || ']; ' || SQLERRM);

END;

END LOOP;

CLOSE CUR;

END;

END IF;

END;

/

-- Add PKs.

set define off;

DECLARE

v\_TN1 VARCHAR2(30);

v\_TN\_OLD VARCHAR2(30) := '';

v\_CN1 VARCHAR2(30);

V\_Q VARCHAR2(3000);

V\_COLS VARCHAR2(3000) := '';

V\_TEMP PLS\_INTEGER;

CURSOR curColData IS

SELECT DICT.TABLE\_NAME, DICT.COL\_NAME

FROM PON\_DICT DICT

LEFT OUTER JOIN USER\_CONSTRAINTS UC ON DICT.TABLE\_NAME = UC.TABLE\_NAME AND UC.CONSTRAINT\_TYPE = 'P'

WHERE UC.TABLE\_NAME IS NULL AND ACTIVE\_PK = 1

ORDER BY DICT.TABLE\_NAME, DICT.COL\_ORDER;

BEGIN

SELECT VAR\_VALUE INTO V\_TEMP FROM PON\_GLOB\_VAR WHERE VARI = 'SPK';

IF V\_TEMP = 1 THEN

OPEN curColData;

FETCH curColData INTO V\_TN1, V\_CN1;

V\_TN\_OLD := V\_TN1;

V\_COLS := '"' || V\_CN1 || '"';

IF curColData%FOUND THEN

LOOP

FETCH curColData INTO v\_TN1,v\_CN1;

--DBMS\_OUTPUT.PUT\_LINE('TN\_OLD|' || V\_TN\_OLD ||'| TN1|' || V\_TN1 || '| CN1|' || V\_CN1 || '|');

IF (V\_TN1 <> V\_TN\_OLD) OR curColData%NOTFOUND THEN

BEGIN

v\_q := 'ALTER TABLE ' || v\_TN\_OLD || ' ADD CONSTRAINT PK\_' || SUBSTR(v\_TN\_OLD,1,27) || ' PRIMARY KEY (' || V\_COLS || ')';

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D4: Failed Creating Primary Key [' || v\_q || ';' || SQLERRM);

END;

V\_TN\_OLD := V\_TN1;

V\_COLS := '"' || V\_CN1 || '"';

ELSE

V\_COLS := V\_COLS || ', "' || V\_CN1 || '"';

END IF;

EXIT WHEN curColData%NOTFOUND;

END LOOP;

CLOSE curColData;

END IF;

END IF;

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

END;

/

-- Add FKs.

DECLARE

v\_TN1 VARCHAR2(30);

v\_TN\_OLD VARCHAR2(30);

v\_R\_TN1 VARCHAR2(30);

v\_R\_TN\_OLD VARCHAR2(30);

v\_FK1 VARCHAR2(30);

v\_FK\_OLD VARCHAR2(30) := '';

v\_CN1 VARCHAR2(30);

v\_R\_CN1 VARCHAR2(30);

V\_DELETE\_RULE VARCHAR2(30);

V\_DELETE\_RULE\_OLD VARCHAR2(30);

V\_Q VARCHAR2(3000);

V\_COLS VARCHAR2(3000) := '';

V\_R\_COLS VARCHAR2(3000) := '';

V\_TEMP PLS\_INTEGER;

CURSOR CURCOLDATA IS

SELECT FK.FK\_NAME, FK.TABLE\_NAME, FK.COL\_NAME, FK.R\_TABLE\_NAME, FK.R\_COL\_NAME, FK.DELETE\_RULE

FROM PON\_FK FK

LEFT OUTER JOIN USER\_CONSTRAINTS C ON UPPER(FK.FK\_NAME) = UPPER(C.CONSTRAINT\_NAME)

WHERE C.CONSTRAINT\_NAME IS NULL AND

(GET\_VAR('HAS\_GUIDS') = 0 AND MOD(PURPOSE, 2) = 1 OR GET\_VAR('HAS\_GUIDS') = 1 AND PURPOSE > 1)

ORDER BY FK.FK\_NAME, FK.COL\_ORDER;

BEGIN

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

OPEN CURCOLDATA;

FETCH CURCOLDATA INTO V\_FK1, V\_TN1, V\_CN1, V\_R\_TN1, V\_R\_CN1, V\_DELETE\_RULE;

IF CURCOLDATA%FOUND THEN

V\_TN\_OLD := V\_TN1;

V\_R\_TN\_OLD := V\_R\_TN1;

V\_FK\_OLD := V\_FK1;

V\_COLS := V\_CN1;

V\_R\_COLS := V\_R\_CN1;

V\_DELETE\_RULE\_OLD := V\_DELETE\_RULE;

LOOP

FETCH CURCOLDATA INTO V\_FK1, V\_TN1, V\_CN1, V\_R\_TN1, V\_R\_CN1, V\_DELETE\_RULE;

--DBMS\_OUTPUT.PUT\_LINE('TN\_OLD|' || V\_TN\_OLD ||'| TN1|' || V\_TN1 || '| FK\_OLD|' || V\_FK\_OLD ||'| FK1|' || V\_FK1 || '| CN1|' || V\_CN1 || '|');

IF (V\_FK1 <> V\_FK\_OLD) OR CURCOLDATA%NOTFOUND THEN

BEGIN

v\_q := 'ALTER TABLE ' || V\_TN\_OLD || ' ADD CONSTRAINT ' || V\_FK\_OLD || ' FOREIGN KEY (' || V\_COLS

|| ') REFERENCES ' || V\_R\_TN\_OLD || ' (' || V\_R\_COLS || ')';

IF V\_DELETE\_RULE\_OLD = 1 THEN

V\_Q := V\_Q || ' ON DELETE CASCADE';

END IF;

/\* Note: Do not append the default on delete rule 'NO ACTION' because this causes an error. \*/

--DBMS\_OUTPUT.PUT\_LINE(V\_Q);

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR D6: Failed Creating Foreign Key [' || v\_q || ';' || SQLERRM);

END;

V\_TN\_OLD := V\_TN1;

V\_R\_TN\_OLD := V\_R\_TN1;

V\_FK\_OLD := V\_FK1;

V\_COLS := V\_CN1;

V\_R\_COLS := V\_R\_CN1;

V\_DELETE\_RULE\_OLD := V\_DELETE\_RULE;

ELSE

V\_COLS := V\_COLS || ', ' || V\_CN1;

V\_R\_COLS := V\_R\_COLS || ', ' || V\_R\_CN1;

END IF;

EXIT WHEN CURCOLDATA%NOTFOUND;

END LOOP;

END IF;

CLOSE CURCOLDATA;

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

END;

/

-- Update and delete data.

/\* SR: 6/10/2014 Update PON\_NAV\_TASK so agency tasks point at new dynamic forms\*/

UPDATE PON\_NAV\_TASK SET NAVIGATE\_URL = '/Inspection/Form.aspx' WHERE NAVIGATE\_URL = '/Inspection/AgencyInspection.aspx';

-- ??? Remove?

DECLARE v\_temp number :=0;

BEGIN

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_PON\_INSP\_WORKCAND\_FLEX';

IF v\_temp > 0 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

execute immediate('alter table PON\_INSP\_WORKCAND DISABLE constraint FK\_PON\_INSP\_WORKCAND\_FLEX');

END IF;

END;

/

/\* 07/15/2013 Delete User Synchronization task. This was inserted into the upgrade scripts and quickly removed. \*/

DECLARE V\_TEMP NUMBER := 0;

BEGIN

SELECT COUNT(\*) INTO V\_TEMP FROM PON\_NAV\_TASK WHERE TAB\_ID = 11 AND TASK\_ID = 22 AND NAME = 'User Synchronization';

IF V\_TEMP > 0 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

DELETE PON\_APP\_TASK\_SECURITY WHERE TAB\_ID = 11 AND TASK\_ID = 22;

DELETE PON\_NAV\_TASK WHERE TAB\_ID = 11 AND TASK\_ID = 22 AND NAME = 'User Synchronization';

END IF;

END;

/

-- ??? Remove?

DECLARE v\_temp number :=0;

BEGIN

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

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_PON\_INSP\_WORKCAND\_FLEX';

IF v\_temp > 0 THEN

execute immediate('ALTER TABLE PON\_INSP\_WORKCAND ENABLE CONSTRAINT FK\_PON\_INSP\_WORKCAND\_FLEX');

END IF;

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_USERS\_GROUPS\_603\_USERS';

IF v\_temp > 0 THEN

execute immediate('alter table PON\_APP\_USERS\_GROUPS DISABLE constraint FK\_USERS\_GROUPS\_603\_USERS');

END IF;

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_USERS\_ROLES\_607\_ROLES';

IF v\_temp > 0 THEN

execute immediate('alter table PON\_APP\_USERS\_ROLES DISABLE constraint FK\_USERS\_ROLES\_607\_ROLES');

END IF;

END IF;

END;

/

-- Migrate USERS into PON\_APP\_USERS.

--Pon\_app\_users.1 | If there is no data in pon\_app\_users then import data from users.

--Userkey is a number in PON\_APP\_USERS and a VARCHAR2 in USERS.

--Only import rows with number values (4 or 0004 etc).

--Upgrading to 52 Tennessee has non numeric Userkeys in Users and Userkey = 1 has a UserId of 'Bridgeware'.

--Upgrading to 52 Alaska has users that have duplicate Userkeys once the leading 0s stripped off.

-- Andrew Bloss 1/16/2015 WIP Moved to Part 1, as it is in SQL.

-- CREATE OR REPLACE FUNCTION IS\_INTEGER (theNumber IN VARCHAR2) RETURN NUMBER AS

-- V\_INT INTEGER;

-- V\_NUM NUMBER;

-- V\_CHAR VARCHAR2(30);

-- BEGIN

-- V\_INT := TO\_NUMBER(LTRIM(theNumber, ' 0'));

-- V\_NUM := TO\_NUMBER(LTRIM(theNumber, ' 0'));

-- V\_CHAR := LTRIM(THENUMBER, ' 0');

-- IF V\_INT = V\_NUM AND V\_CHAR = TO\_CHAR(V\_INT) THEN

-- RETURN 1;

-- ELSE

-- RETURN 0;

-- END IF;

-- RETURN 0;

-- EXCEPTION

-- WHEN OTHERS THEN

-- RETURN 0;

-- END;

-- /

-- Move users into pon\_app\_users.

DECLARE

V\_PON\_APP\_USER\_HAS\_ROWS number :=0;

V\_USER\_EXISTS number :=0;

V\_USEROLES\_EXISTS number :=0;

BEGIN

select COUNT(1) into V\_PON\_APP\_USER\_HAS\_ROWS from PON\_APP\_USERS where rownum=1;

select COUNT(1) into V\_USER\_EXISTS from USER\_TABLES where table\_name = 'USERS';

select COUNT(1) into V\_USEROLES\_EXISTS from USER\_TABLES where table\_name = 'USEROLES';

IF V\_PON\_APP\_USER\_HAS\_ROWS = 0 AND V\_USER\_EXISTS = 1 AND V\_USEROLES\_EXISTS = 1 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Migrate users into pon\_app\_users.');

EXECUTE IMMEDIATE('

DECLARE

CURSOR curUsers IS

SELECT

U.USERKEY, U.USERID, U.LAST\_NAME, U.FIRST\_NAME, U.AGENCY, U.PHONE, U.EMAIL, UPPER(U.SUPERUSER),

CASE

WHEN UPPER(U.SUPERUSER) = ''Y'' THEN 3 --Superuser

WHEN I.USERKEY IS NOT NULL THEN 2 --Advanced user (Inspector)

ELSE 1

END AS ROLEKEY,

U.ADDRESS1,

U.ADDRESS2,

U.CITY,

U.DISTRICT,

U.FAX,

U.INITIALS,

U.MIDDLE,

U.STATE,

U.TITLE,

U.ZIP

FROM USERS U

LEFT OUTER JOIN (SELECT DISTINCT USERKEY, PERMISSION FROM USEROLES) I ON U.USERKEY = I.USERKEY AND I.PERMISSION = ''INSPECTOR''

WHERE IS\_INTEGER(U.USERKEY) = 1 AND USERID IS NOT NULL

ORDER BY CAST(U.USERKEY AS NUMBER), CASE WHEN SUBSTR(U.USERKEY,1,1) = ''0'' THEN 0 ELSE 1 END;

--Process the new Userkeys first so Userkey 0002 becomes 2 and

--Userkey 2 gets a new Userkey.

vUserKey varchar2(8);

vUserId varchar2(12);

vLastName varchar2(20);

vFirstName varchar2(20);

vAgency varchar2(40);

vPhone varchar2(20);

vEmail varchar2(80);

vSuperUser CHAR(1);

intUserKey PLS\_INTEGER;

countMatch PLS\_INTEGER;

vRoleKey PLS\_INTEGER;

vADDRESS1 varchar2(80);

vADDRESS2 varchar2(80);

vCITY varchar2(80);

vDISTRICT varchar2(80);

vFAX varchar2(80);

vINITIALS varchar2(80);

vMIDDLE varchar2(80);

vSTATE varchar2(80);

vTITLE varchar2(80);

vZIP varchar2(80);

BEGIN

INSERT INTO PON\_APP\_GROUPS (GROUPKEY,GROUPNAME,STATUS,NOTES)

SELECT GROUPKEY,GROUPNAME,STATUS,NOTES FROM PON\_APP\_GROUPS\_T;

INSERT INTO PON\_APP\_ROLES (ROLEKEY, ROLENAME, DEFAULTFLAG, STATUS, PONTIS\_STANDARD\_IND)

SELECT ROLEKEY, ROLENAME, DEFAULTFLAG, STATUS, PONTIS\_STANDARD\_IND FROM PON\_APP\_ROLES\_T;

COMMIT;

OPEN curUsers;

FETCH curUsers INTO vUserKey, vUserId, vLastName, vFirstName, vAgency, vPhone, vEmail, vSuperUser, vRoleKey,

vADDRESS1, vADDRESS2, vCITY, vDISTRICT, vFAX, vINITIALS, vMIDDLE, vSTATE, vTITLE, vZIP;

--DBMS\_OUTPUT.PUT\_LINE(vUserKey || '' '' || vUserId);

WHILE (curUsers%FOUND) LOOP

--Make sure this UserKey does not already exist in PON\_APP\_USERS.

--This can happen when one row in Users has a Userkey of 4 and another row has a Userkey of 0004.

--They will both be converted to 4.

intUserKey := vUserKey; --Implicit conversion to PLS\_INTEGER

SELECT COUNT(1) INTO countMatch FROM PON\_APP\_USERS WHERE USERKEY = intUserKey;

--Make a new UserKey if this one already exists in PON\_APP\_USERS.

--Get the greatest Userkey from both tables.

IF (countMatch > 0) THEN

SELECT MAX(USERKEY) + 1 INTO intUserKey

FROM (SELECT CAST(USERKEY AS NUMBER) AS USERKEY

FROM USERS

WHERE IS\_INTEGER(USERKEY) = 1

UNION

SELECT USERKEY

FROM PON\_APP\_USERS);

DBMS\_OUTPUT.PUT\_LINE(''During the upgrade the Userkey of user '' || vUserID || '' was changed to prevent a conflict with another user.'');

DBMS\_OUTPUT.PUT\_LINE(''The Userkey was changed from '' || vUserKey || '' to Userkey '' || intUserKey || ''.'');

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

END IF;

BEGIN

INSERT INTO PON\_APP\_USERS (USERKEY, USERID, STATUS, LAST\_NAME, FIRST\_NAME, AGENCY, PHONE, EMAIL, CREATEDATETIME, IS\_DISABLED,

ADDRESS1, ADDRESS2, CITY, DISTRICT, FAX, INITIALS, MIDDLE\_NAME, STATE, TITLE, ZIP)

VALUES (intUserKey, vUserId, 1, NVL(vLastName, ''<<Empty>>''), NVL(vFirstName, ''<<Empty>>''), vAgency, vPhone, vEmail, SYSDATE, 0,

vADDRESS1, vADDRESS2, vCITY, vDISTRICT, vFAX, vINITIALS, vMIDDLE, vSTATE, vTITLE, vZIP);

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(''Error inserting row from USERS to PON\_APP\_USERS. Userkey: '' || vUserkey || '' Oracle code: '' || SQLERRM);

END;

BEGIN

INSERT INTO PON\_APP\_USERS\_GROUPS (USERKEY, GROUPKEY, NOTES)

VALUES (intUserKey, 0, ''<<Migrated from Pontis 5.1.0>>'');

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(''Error inserting row from USERS to PON\_APP\_USERS\_GROUPS. Userkey: '' || vUserkey || '' Rolekey: '' || vRolekey || '' Oracle code: '' || SQLERRM);

END;

BEGIN

INSERT INTO PON\_APP\_USERS\_ROLES (USERKEY, ROLEKEY, NOTES)

VALUES (intUserKey, vRoleKey, ''<<Migrated from Pontis 5.1.0>>'');

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(''Error inserting row from USERS to PON\_APP\_USERS\_ROLES. Userkey: '' || vUserkey || '' Oracle code: '' || SQLERRM);

END;

FETCH curUsers INTO vUserKey, vUserId, vLastName, vFirstName, vAgency, vPhone, vEmail, vSuperUser, vRoleKey,

vADDRESS1, vADDRESS2, vCITY, vDISTRICT, vFAX, vINITIALS, vMIDDLE, vSTATE, vTITLE, vZIP;

--DBMS\_OUTPUT.PUT\_LINE(vUserKey || '' '' || vUserId);

END LOOP;

Close curUsers;

UPDATE PON\_APP\_USERS SET PASSWORD = ''98-39-E1-73-F6-E2-C0-0E-66-7E-E2-1E-BB-C1-28-36'' WHERE UPPER(USERID)=''PONTIS'' AND PASSWORD IS NULL;

UPDATE PON\_APP\_USERS SET PASSWORD = ''B0-81-DB-E8-5E-1E-C3-FF-C3-D4-E7-D0-22-74-00-CD'' WHERE PASSWORD IS NULL;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(''Error upgrading USERS to PON\_APP\_USERS. Userkey: '' || vUserkey || '' Oracle code: '' || SQLERRM);

END;');

END IF;

END;

/

-- Drop PON\_APP\_USERS.GROUPKEY and ROLEKEY.

DECLARE v\_temp number :=0;

BEGIN

/\* Test with any 513 (Wisconsin 513). \*/

select COUNT(1) into v\_temp

from user\_tab\_cols

where UPPER(TABLE\_NAME) = 'PON\_APP\_USERS' AND UPPER(COLUMN\_NAME) = 'GROUPKEY' and rownum=1;

IF v\_temp > 0 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

EXECUTE IMMEDIATE('

INSERT INTO PON\_APP\_USERS\_GROUPS (USERKEY, GROUPKEY, NOTES)

SELECT U.USERKEY, U.GROUPKEY, ''<<Migrated from Pontis 5.1.0>>''

FROM PON\_APP\_USERS U

LEFT OUTER JOIN PON\_APP\_USERS\_GROUPS G ON U.USERKEY = G.USERKEY AND U.GROUPKEY = G.GROUPKEY

WHERE U.GROUPKEY IS NOT NULL AND G.GROUPKEY IS NULL');

EXECUTE IMMEDIATE ('

INSERT INTO PON\_APP\_USERS\_ROLES (USERKEY, ROLEKEY, NOTES)

SELECT U.USERKEY, U.ROLEKEY, ''<<Migrated from Pontis 5.1.0>>''

FROM PON\_APP\_USERS U

LEFT OUTER JOIN PON\_APP\_USERS\_ROLES R ON U.USERKEY = R.USERKEY AND U.ROLEKEY = R.ROLEKEY

WHERE U.ROLEKEY IS NOT NULL AND R.ROLEKEY IS NULL');

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(TABLE\_NAME) = 'PON\_APP\_USERS' AND UPPER(COLUMN\_NAME) = 'GROUPKEY' and rownum=1;

IF v\_temp > 0 THEN

EXECUTE IMMEDIATE('ALTER TABLE PON\_APP\_USERS DROP COLUMN GROUPKEY');

END IF;

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(TABLE\_NAME) = 'PON\_APP\_USERS' AND UPPER(COLUMN\_NAME) = 'ROLEKEY' and rownum=1;

IF v\_temp > 0 THEN

EXECUTE IMMEDIATE('ALTER TABLE PON\_APP\_USERS DROP COLUMN ROLEKEY');

END IF;

END IF;

END;

/

-- Move PON\_APP\_GROUPS.FILTERKEY TO PON\_APP\_GROUP\_ACCESS\_FILTERS when upgrading from 5103.

DECLARE v\_temp number :=0;

BEGIN

select COUNT(1) into v\_temp from user\_tab\_cols where UPPER(TABLE\_NAME) = 'PON\_APP\_GROUPS' AND UPPER(COLUMN\_NAME) = 'FILTERKEY' and rownum=1;

IF v\_temp > 0 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

EXECUTE IMMEDIATE('DECLARE

CURSOR curGroups IS

SELECT GROUPKEY, FILTERKEY FROM PON\_APP\_GROUPS ORDER BY GROUPKEY;

iUserKey integer;

iGroupKey integer;

iRoleKey integer;

iFilterKey integer;

iRowCtr integer;

v\_temp number :=0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(''IMPORTING PON\_APP\_GROUPS.FILTERKEY into PON\_APP\_GROUP\_ACCESS\_FILTERS.FILTERKEY'');

OPEN curGroups;

FETCH curGroups INTO iGroupKey, iFilterKey;

WHILE (curGroups%FOUND)

LOOP

--migrate into pon\_app\_group\_access\_filters

iRowCtr := 0;

SELECT COUNT(1) INTO iRowCtr FROM PON\_APP\_GROUP\_ACCESS\_FILTERS WHERE FILTERKEY=iFilterKey and GROUPKEY=iGroupKey;

IF iRowCtr = 0 AND iFilterKey <> 0 THEN

INSERT INTO PON\_APP\_GROUP\_ACCESS\_FILTERS (GROUPKEY, FILTERKEY, NOTES) VALUES (iGroupKey, iFilterKey, ''<<Migrated from Pontis 5103>>'');

END IF;

FETCH curGroups INTO iGroupKey, iFilterKey;

END LOOP;

CLOSE curGroups;

EXECUTE IMMEDIATE ''ALTER TABLE PON\_APP\_GROUPS DROP COLUMN FILTERKEY'';

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(''Error migrating PON\_APP\_GROUPS.FILTERKEY to PON\_APP\_GROUP\_ACCESS\_FILTERS.FILTERKEY: '' || SQLERRM);

END;');

END IF;

END;

/

-- Migrate PON\_APP\_GROUPS.GROUPKEY AND ROLEKEY into PON\_APP\_USERS\_GROUPS, PON\_APP\_USERS\_ROLES,

DECLARE

v\_temp number;

V\_Q VARCHAR2(300);

iUserKey integer;

iGroupKey integer;

iRoleKey integer;

iFilterKey integer;

iRowCtr integer;

CURSOR curRoles IS

SELECT ROLEKEY FROM PON\_APP\_ROLES ORDER BY ROLEKEY;

curUsers sys\_refcursor;

curGroups sys\_refcursor;

BEGIN

SELECT COUNT(1) INTO v\_temp FROM user\_tab\_cols

WHERE TABLE\_NAME = 'PON\_APP\_GROUPS' AND

COLUMN\_NAME IN('GROUPKEY', 'ROLEKEY');

IF v\_temp = 2 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

BEGIN

OPEN curUsers FOR 'SELECT USERKEY, GROUPKEY, ROLEKEY FROM PON\_APP\_USERS ORDER BY USERKEY';

FETCH curUsers INTO iUserKey, iGroupKey, iRoleKey;

WHILE (curUsers%FOUND) LOOP

iRowCtr := 0;

SELECT COUNT(1) INTO iRowCtr FROM PON\_APP\_USERS\_GROUPS WHERE USERKEY=iUserKey and GROUPKEY=iGroupKey;

-- D.8.2 | Migrate into pon\_app\_users\_groups.

IF iRowCtr = 0 THEN

INSERT INTO PON\_APP\_USERS\_GROUPS (USERKEY, GROUPKEY, NOTES) VALUES (iUserKey, iGroupKey, '<<Migrated from Pontis 5.1.0>>');

END IF;

iRowCtr := 0;

SELECT COUNT(1) INTO iRowCtr FROM PON\_APP\_USERS\_ROLES WHERE USERKEY=iUserKey and ROLEKEY=iRoleKey;

-- D.8.3 | Migrate into pon\_app\_users\_roles.

IF iRowCtr = 0 THEN

INSERT INTO PON\_APP\_USERS\_ROLES (USERKEY, ROLEKEY, NOTES) VALUES (iUserKey, iRoleKey, '<<Migrated from Pontis 5.1.0>>');

END IF;

FETCH curUsers INTO iUserKey, iGroupKey, iRoleKey;

END LOOP;

CLOSE curUsers;

BEGIN

V\_Q := 'ALTER TABLE PON\_APP\_GROUPS DROP (GROUPKEY, ROLEKEY)';

DBMS\_OUTPUT.PUT\_LINE('IMPORTED PON\_APP\_GROUPS.GROUPKEY into PON\_APP\_USERS\_GROUPS, and ROLEKEY into PON\_APP\_USERS\_ROLES');

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || V\_Q || ';' || SQLERRM);

END;

END;

END IF;

END;

/

-- Check if PON\_LAYOUTS needs updated.

DECLARE

v\_str VARCHAR2(3000);

BEGIN

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

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", b.featint AS "Feature\_Intersected", pd.shortdesc AS "District", pc.shortdesc AS "County", pl.shortdesc AS "Place", b.length AS "Meters", b.yearbuilt AS "Built", b.facility AS "Facility\_Carried", mt.shortdesc AS "Material" FROM bridge b, paramtrs pd, paramtrs pc, paramtrs po, paramtrs pm, paramtrs pl, paramtrs mt WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pl.table\_name=''bridge'' AND pl.field\_name=''placecode'' AND pl.parmvalue=b.placecode AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND po.table\_name=''bridge'' AND po.field\_name=''custodian'' AND po.parmvalue=b.owner AND pm.table\_name=''bridge'' AND pm.field\_name=''custodian'' AND pm.parmvalue=b.custodian AND mt.table\_name=''bridge'' AND mt.field\_name=''materialmain'' AND mt.parmvalue=b.materialmain AND b.designmain=''19''' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 3 AND SQL\_QUERY = v\_str;

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", b.featint AS "Feature\_Intersected", pd.shortdesc AS "District", pc.shortdesc AS "County", pl.shortdesc AS "Place", b.length AS "Meters", b.yearbuilt AS "Built", b.facility AS "Facility\_Carried", mt.shortdesc AS "Material" FROM bridge b, paramtrs pd, paramtrs pc, paramtrs po, paramtrs pm, paramtrs pl, paramtrs mt WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pl.table\_name=''bridge'' AND pl.field\_name=''placecode'' AND pl.parmvalue=b.placecode AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND po.table\_name=''bridge'' AND po.field\_name=''custodian'' AND po.parmvalue=b.owner AND pm.table\_name=''bridge'' AND pm.field\_name=''custodian'' AND pm.parmvalue=b.custodian AND mt.table\_name=''bridge'' AND mt.field\_name=''materialmain'' AND mt.parmvalue=b.materialmain AND b.designmain=''18''' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 4 AND SQL\_QUERY = v\_str;

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", b.featint AS "Feature\_Intersected", pd.shortdesc AS "District", pc.shortdesc AS "County", pl.shortdesc AS "Place", b.length AS "Meters", b.yearbuilt AS "Built", b.facility AS "Facility\_Carried" FROM bridge b, paramtrs pd, paramtrs pc, paramtrs po, paramtrs pm, paramtrs pl WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pl.table\_name=''bridge'' AND pl.field\_name=''placecode'' AND pl.parmvalue=b.placecode AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND po.table\_name=''bridge'' AND po.field\_name=''custodian'' AND po.parmvalue=b.owner AND pm.table\_name=''bridge'' AND pm.field\_name=''custodian'' AND pm.parmvalue=b.custodian AND b.brkey IN (SELECT DISTINCT brkey FROM roadway WHERE on\_under<>''1'')' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 5 AND SQL\_QUERY = v\_str;

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", pd.shortdesc AS "District", pc.shortdesc AS "County", facility AS "Facility\_Carried", featint AS "Feature\_Intersected", po.shortdesc AS "Own", pm.shortdesc AS "Maint", yearbuilt AS "Built" FROM bridge b, paramtrs pd, paramtrs pc, paramtrs po, paramtrs pm WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND po.table\_name=''bridge'' AND po.field\_name=''custodian'' AND po.parmvalue=b.owner AND pm.table\_name=''bridge'' AND pm.field\_name=''custodian'' AND pm.parmvalue=b.custodian AND b.srstatus <> ''0''' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 8 AND SQL\_QUERY = v\_str;

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", pd.shortdesc AS "District", pc.shortdesc AS "County", facility AS "Facility\_Carried", featint AS "Feature\_Intersected", po.shortdesc AS "Own", pm.shortdesc AS "Maint", yearbuilt AS "Built" FROM bridge b, paramtrs pd, paramtrs pc, paramtrs po, paramtrs pm WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND po.table\_name=''bridge'' AND po.field\_name=''custodian'' AND po.parmvalue=b.owner AND pm.table\_name=''bridge'' AND pm.field\_name=''custodian'' AND pm.parmvalue=b.custodian AND b.brkey not in (select brkey from inspevnt)' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 9 AND SQL\_QUERY = v\_str;

v\_str := 'SELECT b.brkey AS "key\_", bridge\_id AS "Bridge\_ID", pa.shortdesc AS "Changed\_Since\_Check\_Out", pd.shortdesc AS "District", pc.shortdesc AS "County", facility AS "Facility\_Carried", featint AS "Feature\_Intersected", b.length AS "Meters", yearbuilt AS "Built" FROM bridge b, cicocntl c1, paramtrs pd, paramtrs pc, paramtrs pa WHERE pc.table\_name=''bridge'' AND pc.field\_name=''county'' AND pc.parmvalue=b.county AND pd.table\_name=''bridge'' AND pd.field\_name=''district'' AND pd.parmvalue=b.district AND pa.table\_name=''cicocntl'' AND pa.field\_name=''atrigger'' AND pa.parmvalue=c1.atrigger AND (c1.brkey = b.brkey) AND (c1.ioflag = ''O'') AND c1.iomoment = (select max(c2.iomoment) from cicocntl c2 where c2.brkey = c1.brkey and c2.ioflag = ''O'') AND NOT EXISTS ( select 1 from CICOCNTL c3 where c3.ioflag = ''I'' and c1.brkey = c3.brkey and c3.iomoment > c1.iomoment)' ;

DELETE PON\_LAYOUTS WHERE LAYOUTKEY = 11 AND SQL\_QUERY = v\_str;

-- indicate pre 5.2 layouts are obsolete, don't update the records more than once

UPDATE PON\_LAYOUTS SET NAME = 'OBSOLETE ' || NAME WHERE SQL\_QUERY LIKE 'SELECT %FROM %' AND NAME NOT LIKE '%OBSOLETE%' AND PONTIS\_STANDARD\_IND <> 'T';

UPDATE PON\_LAYOUTS SET CONTEXT = 'Inspection' WHERE CONTEXT = 'BRIDGES';

UPDATE PON\_FILTERS SET CONTEXT = 'Inspection' WHERE CONTEXT = 'BRIDGES';

END IF;

END;

/

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('USERS, GROUPS AND LAYOUTS', CURRENT\_TIMESTAMP);

-- Add pontis user.

/\*

There must be a row Userkey = 1 and Userid = 'Pontis'. If a Pontis user is added then tell user to change the password for the Pontis

user.

Move users that have a Userkey of 1 and are not called Pontis.

Update PON\_APP\_USERS\_ROLES and PON\_APP\_USERS\_GROUPS.

IF Pontis exists and its Userkey is not equal to 1 then change its name to 'OldPontis'.

Upgrading to 52 Tennessee has non numeric Userkeys in Users and Userkey = 1 has a UserId of 'Bridgeware'.

\*/

DECLARE

V\_SQUATTERID VARCHAR2(30);

V\_NEW\_USERKEY PLS\_INTEGER;

V\_PONTIS\_USER\_FOUND PLS\_INTEGER := 0;

BEGIN

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

BEGIN

SELECT COUNT(\*) INTO V\_PONTIS\_USER\_FOUND FROM PON\_APP\_USERS WHERE USERKEY = 1 AND UPPER(USERID) = 'PONTIS';

--Is a non Pontis user squatting on UserKey 1?

SELECT MIN(USERID) INTO V\_SQUATTERID FROM PON\_APP\_USERS WHERE USERKEY = 1 AND UPPER(USERID) <> 'PONTIS';

IF V\_SQUATTERID IS NOT NULL THEN

SELECT NVL(MAX(USERKEY)+ 1,1) INTO V\_NEW\_USERKEY FROM PON\_APP\_USERS;

--Change USERID to a junk value for USERKEY 1, it will be overwritten with 'Pontis' during the merges.

--UserIDs must be unique.

UPDATE PON\_APP\_USERS SET USERID = V\_NEW\_USERKEY WHERE USERKEY = 1;

INSERT INTO PON\_APP\_USERS (USERKEY, USERID, LAST\_NAME, FIRST\_NAME, MIDDLE\_NAME, STATUS, AGENCY, PHONE, EMAIL, CREATEDATETIME, PASSWORD, DISTRICT, TITLE, ADDRESS1, ADDRESS2, CITY, STATE, ZIP, FAX, INITIALS)

SELECT V\_NEW\_USERKEY, V\_SQUATTERID, LAST\_NAME, FIRST\_NAME, MIDDLE\_NAME, STATUS, AGENCY, PHONE, EMAIL, CREATEDATETIME, PASSWORD, DISTRICT, TITLE, ADDRESS1, ADDRESS2, CITY, STATE, ZIP, FAX, INITIALS

FROM PON\_APP\_USERS

WHERE USERKEY = 1;

UPDATE PON\_APP\_USERS\_ROLES SET USERKEY = V\_NEW\_USERKEY WHERE USERKEY = 1;

UPDATE PON\_APP\_USERS\_GROUPS SET USERKEY = V\_NEW\_USERKEY WHERE USERKEY = 1;

--Merge the standard setup data and password into the Pontis user.

--If the Pontis user is missing the default merge of missing data will add it.

UPDATE PON\_DICT SET FORCE\_DEF = 1 WHERE TABLE\_NAME = 'PON\_APP\_USERS';

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

DBMS\_OUTPUT.PUT\_LINE('Userkey 1 is reserved for the Pontis user.');

DBMS\_OUTPUT.PUT\_LINE('User ' || V\_SQUATTERID || ' had a userkey of 1 and has been changed to ' || V\_NEW\_USERKEY || '.');

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

END IF;

--Always remove any Pontis user that does not have a USERKEY of 1.

UPDATE PON\_APP\_USERS

SET USERID = 'OldPontis' || CAST(USERKEY AS VARCHAR(30)), IS\_DISABLED = 1

WHERE UPPER(USERID) = 'PONTIS' AND USERKEY <> 1;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Could not move User ' || V\_SQUATTERID || ' away from UserKey 1 (Reserved for Pontis). Error: ' || SQLERRM);

END;

END IF;

END;

/

-- Fix bug in default captions.

BEGIN

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

UPDATE PON\_APP\_CAPTION SET PONTIS\_STANDARD\_IND = 'F';

MERGE INTO PON\_APP\_CAPTION T

USING PON\_APP\_CAPTION\_T S

ON (

T.CAPTION\_ID = S.CAPTION\_ID AND

T.DEFAULT\_CAPTION = S.DEFAULT\_CAPTION

)

WHEN MATCHED THEN UPDATE SET T.PONTIS\_STANDARD\_IND = 'T';

END IF;

END;

/

-- ReformatCustom

DECLARE

V\_N PLS\_INTEGER := 0;

V\_Q VARCHAR2(1000);

BEGIN

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

select COUNT(1) into V\_N from user\_tab\_cols where UPPER(TABLE\_NAME) = 'PON\_LAYOUTS' AND UPPER(COLUMN\_NAME) = 'PONTIS\_STANDARD\_IND';

IF V\_N = 0 THEN

EXECUTE IMMEDIATE ('UPDATE PON\_LAYOUTS SET PONTIS\_STANDARD\_IND = ''T'' WHERE LAYOUTKEY IN (SELECT t1.LAYOUTKEY FROM PON\_LAYOUTS t1 INNER JOIN PON\_LAYOUTS\_T t2 ON t1.LAYOUTKEY = t2.LAYOUTKEY WHERE t1.LAYOUTKEY = t2.LAYOUTKEY AND t1.NAME = t2.NAME)');

END IF;

WITH CTE AS (

SELECT 'PON\_NAV\_TAB', TAB\_ID AS ID, PONTIS\_STANDARD\_IND FROM PON\_NAV\_TAB UNION ALL

SELECT 'PON\_APP\_ROLES', ROLEKEY, PONTIS\_STANDARD\_IND FROM PON\_APP\_ROLES UNION ALL

SELECT 'PON\_NAV\_CONTROL\_GROUP', CONTROL\_GROUP\_ID, PONTIS\_STANDARD\_IND FROM PON\_NAV\_CONTROL\_GROUP UNION ALL

SELECT 'PON\_NAV\_TASK', TASK\_ID, PONTIS\_STANDARD\_IND FROM PON\_NAV\_TASK UNION ALL

SELECT 'PON\_NAV\_CONTROL', CONTROL\_ID, PONTIS\_STANDARD\_IND FROM PON\_NAV\_CONTROL UNION ALL

SELECT 'PON\_APP\_CAPTION', CAPTION\_ID, PONTIS\_STANDARD\_IND FROM PON\_APP\_CAPTION

)

SELECT COUNT(\*) INTO V\_N FROM CTE WHERE ID < 15000 AND (PONTIS\_STANDARD\_IND IS NULL OR PONTIS\_STANDARD\_IND <> 'T');

IF V\_N > 0 THEN

--Disable all constraints and indexes before calling ReformatCustom.

--The code to make indexes relies on all of the indexes being dropped.

DECLARE

V\_Q VARCHAR2(1000);

BEGIN

FOR I IN

(SELECT TABLE\_NAME, CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS

WHERE STATUS = 'ENABLED'

AND TABLE\_NAME IN (SELECT DISTINCT TABLE\_NAME FROM PON\_DICT)

ORDER BY CASE CONSTRAINT\_TYPE WHEN 'R' THEN 1 ELSE 2 END)

LOOP

V\_Q := 'ALTER TABLE ' ||I.TABLE\_NAME|| ' DISABLE CONSTRAINT ' ||I.CONSTRAINT\_NAME || ' CASCADE';

BEGIN

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR disabling constraint before reformat: [' || v\_q || ';' || SQLERRM);

END;

END LOOP;

END;

DECLARE

v\_temp number :=0;

BEGIN

UPDATE PON\_APP\_ROLES

SET PONTIS\_STANDARD\_IND = 'T'

WHERE ROLEKEY IN (SELECT R.ROLEKEY

FROM PON\_APP\_ROLES R

INNER JOIN PON\_APP\_ROLES\_T T ON T.ROLEKEY = R.ROLEKEY AND T.ROLENAME = R.ROLENAME);

ReformatCustom('PON\_APP\_ROLES', 'ROLEKEY');

ReformatCustom('PON\_NAV\_TAB', 'TAB\_ID');

ReformatCustom('PON\_NAV\_TASK', 'TASK\_ID');

ReformatCustom('PON\_NAV\_CONTROL\_GROUP', 'CONTROL\_GROUP\_ID');

ReformatCustom('PON\_NAV\_CONTROL', 'CONTROL\_ID');

ReformatCustom('PON\_APP\_CAPTION', 'CAPTION\_ID');

ReformatCustom('PON\_FILTERS', 'FILTERKEY');

ReformatCustom('PON\_LAYOUTS', 'LAYOUTKEY');

END;

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

--Enable all constraints

DECLARE

V\_Q VARCHAR2(1000);

BEGIN

FOR I IN

(SELECT TABLE\_NAME, CONSTRAINT\_NAME

FROM USER\_CONSTRAINTS WHERE STATUS = 'DISABLED'

AND TABLE\_NAME IN (SELECT DISTINCT TABLE\_NAME FROM PON\_DICT)

ORDER BY CONSTRAINT\_TYPE)

LOOP

V\_Q := 'ALTER TABLE ' ||I.TABLE\_NAME|| ' ENABLE CONSTRAINT ' || I.CONSTRAINT\_NAME;

BEGIN

EXECUTE IMMEDIATE V\_Q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR enabling constraint after reformat: [' || v\_q || ';' || SQLERRM);

END;

END LOOP;

END;

END IF;

END IF;

END;

/

-- Table updates (again).

--1/17/2013 Ben Daniels - Removing column ELEM\_WEARINGSURFACE from PON\_ELEM\_DEFS if it exists

DECLARE

V\_X PLS\_INTEGER;

BEGIN

SELECT COUNT(1) INTO V\_X FROM USER\_TAB\_COLS WHERE TABLE\_NAME = 'PON\_ELEM\_DEFS' AND COLUMN\_NAME = 'ELEM\_WEARINGSURFACE';

IF V\_X > 0 THEN

BEGIN

EXECUTE IMMEDIATE 'ALTER TABLE PON\_ELEM\_DEFS DROP COLUMN ELEM\_WEARINGSURFACE';

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error deleting column ELEM\_WEARINGSURFACE from PON\_ELEM\_DEFS at start of script' || SQLERRM);

END;

END IF;

END;

/

BEGIN

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

DELETE FROM (

SELECT \* FROM PON\_APP\_CONTROL\_SECURITY t1

INNER JOIN PON\_NAV\_CONTROL t2 ON t2.tab\_id=t1.tab\_id AND t2.task\_id=t1.task\_id AND t2.control\_group\_id=t1.control\_group\_id AND t2.control\_id=t1.control\_id

WHERE t2.pontis\_standard\_ind='T' and t2.name='R\_E\_M\_O\_V\_E');

END IF;

END;

/

DELETE FROM PON\_NAV\_CONTROL WHERE name='R\_E\_M\_O\_V\_E' AND pontis\_standard\_ind='T';

-- = E | Non-Dynamic Inserts With Checks ================================================= --

declare

c int;

begin

select count(1) into c from user\_tables where table\_name = 'PON\_NAV\_CONTROL\_GROUP';

if c = 1 then

execute immediate 'select count(1) from PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=20 AND TAB\_ID=4 AND CONTROL\_GROUP\_ID = 15 AND NAME=''App. Cond. States'''

into c;

if c = 1 then

execute immediate 'DELETE FROM PON\_APP\_CONTROL\_SECURITY WHERE TASK\_ID=20 AND TAB\_ID=4 AND CONTROL\_GROUP\_ID = 15';

execute immediate 'DELETE FROM PON\_APP\_CONTROL\_GROUP\_SECURITY WHERE TASK\_ID=20 AND TAB\_ID=4 AND CONTROL\_GROUP\_ID = 15';

execute immediate 'DELETE FROM PON\_NAV\_CONTROL WHERE TASK\_ID=20 AND TAB\_ID=4 AND CONTROL\_GROUP\_ID = 15 ';

execute immediate 'DELETE FROM PON\_NAV\_CONTROL\_GROUP WHERE TASK\_ID=20 AND TAB\_ID=4 AND CONTROL\_GROUP\_ID = 15';

end if;

end if;

end;

/

/\*\*\* F.1.0 | Update User/Group Tables including null password. \*\*\*/

DECLARE v\_temp number :=0;

v\_q VARCHAR2(500);

BEGIN

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

SELECT COUNT(1) INTO v\_temp FROM user\_tab\_cols WHERE UPPER(TABLE\_NAME) = 'PON\_APP\_GROUPS' AND UPPER(COLUMN\_NAME) = 'FILTERKEY' AND ROWNUM = 1;

IF v\_temp > 0 THEN

DELETE FROM DATADICT WHERE TABLE\_NAME = 'PON\_APP\_GROUPS' AND COL\_NAME = 'FILTERKEY';

END IF;

SELECT COUNT(1) INTO v\_temp FROM user\_tab\_cols WHERE UPPER(TABLE\_NAME) = 'PON\_APP\_USERS' AND UPPER(COLUMN\_NAME) = 'GROUPKEY' AND ROWNUM = 1;

IF v\_temp > 0 THEN

DELETE FROM DATADICT WHERE TABLE\_NAME = 'PON\_APP\_USERS' AND COL\_NAME = 'GROUPKEY';

END IF;

SELECT COUNT(1) INTO v\_temp FROM user\_tab\_cols WHERE UPPER(TABLE\_NAME) = 'PON\_APP\_USERS' AND UPPER(COLUMN\_NAME) = 'ROLEKEY' AND ROWNUM = 1;

IF v\_temp > 0 THEN

DELETE FROM DATADICT WHERE TABLE\_NAME = 'PON\_APP\_USERS' AND COL\_NAME = 'ROLEKEY';

END IF;

SELECT COUNT(1) INTO v\_temp FROM PON\_APP\_ROLES\_PERMISSIONS WHERE PERMISSIONKEY=600;

IF v\_temp > 0 THEN

DELETE FROM PON\_APP\_ROLES\_PERMISSIONS WHERE PERMISSIONKEY=600;

END IF;

SELECT COUNT(1) INTO v\_temp FROM PON\_APP\_ROLES\_PERMISSIONS WHERE PERMISSIONKEY=600;

IF v\_temp > 0 THEN

DELETE FROM PON\_APP\_PERMISSIONS WHERE PERMISSIONKEY=600;

END IF;

END IF;

END;

/

-- Migrate COPTIONS to PON\_COPTIONS.

DECLARE

V\_ROWS\_IN\_COPTIONS PLS\_INTEGER;

V\_ROWS\_IN\_PON\_COPTIONS PLS\_INTEGER;

V\_X PLS\_INTEGER;

BEGIN

SELECT COUNT(1) INTO V\_X FROM USER\_TABLES WHERE TABLE\_NAME IN ('PON\_COPTIONS', 'PON\_COPTIONS\_T', 'COPTIONS');

IF V\_X = 3 THEN

BEGIN

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM COPTIONS' INTO V\_ROWS\_IN\_COPTIONS;

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM PON\_COPTIONS' INTO V\_ROWS\_IN\_PON\_COPTIONS;

IF V\_ROWS\_IN\_COPTIONS > 0 AND V\_ROWS\_IN\_PON\_COPTIONS = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Migrate COPTIONS to PON\_COPTIONS.');

EXECUTE IMMEDIATE('MERGE INTO PON\_COPTIONS\_T T

USING (SELECT \* FROM COPTIONS WHERE OPTIONVAL <> ''-1'') S

ON (T.OPTIONNAME = S.OPTIONNAME)

WHEN MATCHED THEN UPDATE SET T.OPTIONVAL = S.OPTIONVAL');

END IF;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('COPTIONS has failed migrating into PON\_COPTIONS. BrM options will be reset to default;' || SQLERRM);

END;

END IF;

END;

/

-- 12/7/2014 Lower case in datadict.

UPDATE DATADICT SET

TABLE\_NAME = LOWER(TABLE\_NAME),

COL\_NAME = LOWER(COL\_NAME),

COL\_ALIAS = LOWER(COL\_ALIAS),

DATATYPE = LOWER(DATATYPE),

UNIQUE\_FLD = LOWER(UNIQUE\_FLD);

-- BQ-1295 Match pon\_insp\_statuses with paramtrs.

-- Pon\_insp\_statuses is force merged. Paramtrs is not.

-- Approved is now 3.

DELETE PON\_INSP\_STATUSES

WHERE STATUS\_KEY = 5

AND STATUS\_NAME = 'Approved'

AND STATUS\_IS\_APPROVED = 'T';

-- Archived is now 6.

DELETE PARAMTRS

WHERE TABLE\_NAME = 'inspevnt'

AND FIELD\_NAME = 'inspstat'

AND PARMVALUE = '5'

AND SHORTDESC = 'Archived';

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

------------------------------------------------------------ END OF --------------------------------------------------------------

--------------------------------------------------------- TABLE UPDATES ----------------------------------------------------------

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

-- Disable recursive constraints before merging setup data.

DECLARE v\_temp number :=0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Disable recursive constraints before merging setup data.');

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_NAV\_CTRLGRP\_RECURSIVE';

IF v\_temp > 0 THEN

execute immediate('ALTER TABLE PON\_NAV\_CONTROL\_GROUP DISABLE CONSTRAINT FK\_NAV\_CTRLGRP\_RECURSIVE');

END IF;

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_NAV\_TASK\_RECURSIVE';

IF v\_temp > 0 THEN

execute immediate('ALTER TABLE PON\_NAV\_TASK DISABLE CONSTRAINT FK\_NAV\_TASK\_RECURSIVE');

END IF;

END;

/

-- Merge setup data.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Merge optional setup data from temp tables.');

END;

/

DECLARE

V\_Q VARCHAR2(8000);

V\_T\_COLS VARCHAR2(8000);

V\_S\_COLS VARCHAR2(8000);

V\_MATCH\_COLS VARCHAR2(8000);

BEGIN

FOR X IN (

SELECT T1.TABLE\_NAME

FROM PON\_TABLE T1

JOIN USER\_TABLES UT ON SUBSTR(T1.TABLE\_NAME, 1, 28) || '\_T' = UT.TABLE\_NAME

WHERE (

RANK BETWEEN 1 AND 98

-- A non primary column.

AND EXISTS(SELECT \*

FROM PON\_DICT

WHERE TABLE\_NAME = T1.TABLE\_NAME

AND ACTIVE\_PK = 0)

AND EXISTS(SELECT \*

FROM PON\_DICT

WHERE TABLE\_NAME = T1.TABLE\_NAME

AND ACTIVE\_PK = 1)

AND T1.TABLE\_NAME <> 'DBDESCRP'

)

-- Add a row to DBDESCRP if it is empty.

OR T1.TABLE\_NAME = 'DBDESCRP' AND NOT EXISTS(SELECT \* FROM DBDESCRP)

ORDER BY T1.RANK, T1.TABLE\_NAME

)

LOOP

BEGIN

SELECT PON\_CONCAT(CURSOR(SELECT 'T.' || DICT.COL\_NAME || ' = S.' || DICT.COL\_NAME

FROM PON\_DICT DICT

WHERE DICT.TABLE\_NAME = X.TABLE\_NAME

AND ACTIVE\_PK = 1

ORDER BY DICT.COL\_ORDER)

,' AND ')

INTO V\_MATCH\_COLS

FROM DUAL;

SELECT PON\_CONCAT(CURSOR(SELECT 'T.' || COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = X.TABLE\_NAME

ORDER BY COL\_ORDER)

,', ')

INTO V\_T\_COLS

FROM DUAL;

V\_S\_COLS := REPLACE(V\_T\_COLS, 'T.', 'S.');

SELECT

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

USING ' || SUBSTR(X.TABLE\_NAME, 1, 28) || '\_T' || ' S

ON (' || V\_MATCH\_COLS || ')

WHEN NOT MATCHED THEN INSERT (' || V\_T\_COLS || ')

VALUES (' || V\_S\_COLS || ')'

INTO V\_Q

FROM DUAL;

EXECUTE IMMEDIATE V\_Q;

COMMIT;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR : ' || V\_Q);

DBMS\_OUTPUT.PUT\_LINE(' ' || SQLERRM);

END;

END LOOP;

END;

/

COMMIT;

-- Merge required setup data from temp tables.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Merge required setup data from temp tables.');

END;

/

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT

'UPDATE ' || T.TABLE\_NAME || ' ' ||

'SET (' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 0

AND FORCE\_DEF = 1

ORDER BY COL\_ORDER

),

', '

) ||

') = (' ||

'SELECT ' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 0

AND FORCE\_DEF = 1

ORDER BY COL\_ORDER

),

', '

) || ' ' ||

'FROM ' || SUBSTR(T.TABLE\_NAME, 1, 28) || '\_T ' ||

'WHERE ' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME || ' = ' || T.TABLE\_NAME || '.' || COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 1

ORDER BY COL\_ORDER

),

' AND '

) || ' ' ||

') ' ||

'WHERE ( ' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 1

ORDER BY COL\_ORDER

),

', '

) ||

') IN (' ||

'SELECT ' ||

PON\_CONCAT(

CURSOR(

SELECT COL\_NAME

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 1

ORDER BY COL\_ORDER

),

', '

) || ' ' ||

'FROM ' || SUBSTR(T.TABLE\_NAME, 1, 28) || '\_T' ||

')'

FROM PON\_TABLE T

JOIN USER\_TABLES UT ON SUBSTR(T.TABLE\_NAME, 1, 28) || '\_T' = UT.TABLE\_NAME

WHERE RANK BETWEEN 1 AND 98

-- A non PK column with FORCE\_DEF = 1.

AND EXISTS(

SELECT \*

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 0

AND FORCE\_DEF = 1

)

AND EXISTS(

SELECT \*

FROM PON\_DICT

WHERE TABLE\_NAME = T.TABLE\_NAME

AND ACTIVE\_PK = 1

)

ORDER BY T.RANK, T.TABLE\_NAME;

REV(V\_CUR);

END;

/

-- Enable recursive constraints after merging setup data.

DECLARE v\_temp number :=0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Enable recursive constraints after merging setup data.');

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_NAV\_CTRLGRP\_RECURSIVE';

IF v\_temp > 0 THEN

execute immediate('ALTER TABLE PON\_NAV\_CONTROL\_GROUP ENABLE CONSTRAINT FK\_NAV\_CTRLGRP\_RECURSIVE');

END IF;

SELECT COUNT(1) INTO v\_temp FROM USER\_CONSTRAINTS WHERE CONSTRAINT\_NAME = 'FK\_NAV\_TASK\_RECURSIVE';

IF v\_temp > 0 THEN

execute immediate('ALTER TABLE PON\_NAV\_TASK ENABLE CONSTRAINT FK\_NAV\_TASK\_RECURSIVE');

END IF;

END;

/

-- Migrate FLEXACTIONS to PON\_FLEXACTIONS\_SETS.

DECLARE

V\_FLEXACTIONS\_EXISTS PLS\_INTEGER;

V\_FLEXACTIONS\_HAS\_ROWS PLS\_INTEGER;

V\_ROWS\_ALREADY\_IMPORTED PLS\_INTEGER;

BEGIN

SELECT COUNT(1) INTO V\_FLEXACTIONS\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'FLEXACTIONS';

IF V\_FLEXACTIONS\_EXISTS = 1 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

NULL;

EXECUTE IMMEDIATE 'SELECT COUNT(1) INTO :FLEXACTIONS\_HAS\_ROWS FROM FLEXACTIONS WHERE ROWNUM = 1'

INTO V\_FLEXACTIONS\_HAS\_ROWS;

--SELECT COUNT(1) INTO V\_ROWS\_ALREADY\_IMPORTED FROM PON\_FLEXACTIONS\_SETS WHERE FLEX\_ACTION\_KEY='-1' AND ROWNUM = 1;

IF V\_FLEXACTIONS\_EXISTS = 1 AND V\_ROWS\_ALREADY\_IMPORTED = 0 THEN

EXECUTE IMMEDIATE

'INSERT INTO PON\_FLEXACTIONS\_SETS(FLEX\_ACTION\_KEY, FLEX\_ACT\_NAME\_SHORT, FLEX\_ACT\_NAME\_DESC, FLEX\_NOTES)

SELECT FXACTKEY, FXNAME, FXLONGNAME, DESCRIPTION

FROM FLEXACTIONS

WHERE FLEXACTIONS.FXACTKEY IS NOT NULL';

END IF;

END IF;

END;

/

-- Add -1 to PON\_FLEXACTIONS\_SETS.

DECLARE v\_temp number;

BEGIN

SELECT COUNT(1) INTO v\_temp FROM PON\_FLEXACTIONS\_SETS WHERE FLEX\_ACTION\_KEY='-1' AND ROWNUM = 1;

IF v\_temp < 1 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

INSERT INTO PON\_FLEXACTIONS\_SETS(FLEX\_ACTION\_KEY, FLEX\_ACT\_NAME\_SHORT, FLEX\_ACT\_NAME\_DESC, FLEX\_NOTES)

VALUES ('-1', 'Converted Work Candidates', 'Converted Work Candidates', 'Converted Work Candidates' );

END IF;

END;

/

-- Migrate INSP\_WCAND to PON\_INSP\_WORKCAND.

DECLARE

V\_TEMP PLS\_INTEGER;

V\_NEW\_TABLE\_IS\_EMPTY PLS\_INTEGER;

V\_OLD\_TABLE\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(1) INTO V\_TEMP FROM PON\_INSP\_WORKCAND WHERE ROWNUM = 1;

IF V\_TEMP < 1 THEN

V\_NEW\_TABLE\_IS\_EMPTY := 1;

END IF;

SELECT COUNT(1) INTO V\_OLD\_TABLE\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'INSP\_WCAND';

IF V\_NEW\_TABLE\_IS\_EMPTY = 1 AND V\_OLD\_TABLE\_EXISTS = 1 AND GET\_VAR('HAS\_GUIDS') = 0 THEN

EXECUTE IMMEDIATE('INSERT INTO PON\_INSP\_WORKCAND(IWC\_WORKCAND\_ID, IWC\_WORKCAND\_NAME, BRKEY, STRUNITKEY, INSPKEY, FLEX\_ACTION\_KEY, IWC\_AGENCY\_PRIORITY, IWC\_WORK\_REC\_DATE, IWC\_TARGET\_YEAR,

IWC\_WORK\_ASSIGNED, IWC\_EST\_COST, IWC\_EST\_QTY, IWC\_STATUS, IWC\_CREATEDATETIME, IWC\_CREATEUSERKEY, IWC\_MODTIME, IWC\_MODUSERKEY, IWC\_DOCREFKEY, IWC\_NOTES)

SELECT DISTINCT INSP\_WCAND.WCKEY, INSP\_WCAND.WC\_ID, INSP\_WCAND.BRKEY, INSP\_WCAND.STRUNITKEY, INSP\_WCAND.INSPKEY,

(CASE WHEN INSP\_WCAND.ACTKIND = ''3'' THEN ACTCODE ELSE ''-1'' END), INSP\_WCAND.AGENCY\_PRIORITY, INSP\_WCAND.WORKRECDATE, INSP\_WCAND.TARGETYEAR, INSP\_WCAND.WORKASSIGNMENT,

INSP\_WCAND.ESTIMCOST, INSP\_WCAND.ESTIMQUANTITY, INSP\_WCAND.AGENCY\_STATUS, INSP\_WCAND.CREATEDATETIME, INSP\_WCAND.CREATEUSERKEY, INSP\_WCAND.MODTIME, INSP\_WCAND.USERKEY,

INSP\_WCAND.DOCREFKEY, INSP\_WCAND.NOTES

FROM INSP\_WCAND INNER JOIN INSPEVNT ON INSP\_WCAND.BRKEY=INSPEVNT.BRKEY AND INSP\_WCAND.INSPKEY=INSPEVNT.INSPKEY

WHERE INSP\_WCAND.WC\_ID IS NOT NULL AND INSP\_WCAND.BRKEY IS NOT NULL AND INSP\_WCAND.INSPKEY IS NOT NULL

AND ((CASE WHEN INSP\_WCAND.ACTKIND = ''3'' THEN ACTCODE ELSE ''-1'' END) IN (SELECT FLEX\_ACTION\_KEY FROM PON\_FLEXACTIONS\_SETS) OR ACTCODE IS NULL)

AND (INSP\_WCAND.BRKEY, INSP\_WCAND.INSPKEY) IN (SELECT BRKEY, INSPKEY FROM INSPEVNT)');

INSERT INTO PARAMTRS(TABLE\_NAME, FIELD\_NAME, PARMVALUE, SHORTDESC, LONGDESC, MISVALFLG, HELPID)

SELECT 'PON\_INSP\_WORKCAND', 'IWC\_WORK\_ASSIGNED', t1.PARMVALUE, t1.SHORTDESC, t1.LONGDESC, t1.MISVALFLG, t1.HELPID

FROM (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='INSP\_WCAND' AND (FIELD\_NAME='workassignment')) t1

LEFT JOIN (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='PON\_INSP\_WORKCAND' AND (FIELD\_NAME='IWC\_WORK\_ASSIGNED')) t2 ON t1.PARMVALUE = t2.PARMVALUE

WHERE t2.FIELD\_NAME IS NULL;

INSERT INTO PARAMTRS(TABLE\_NAME, FIELD\_NAME, PARMVALUE, SHORTDESC, LONGDESC, MISVALFLG, HELPID)

SELECT 'PON\_INSP\_WORKCAND', 'IWC\_AGENCY\_PRIORITY', t1.PARMVALUE, t1.SHORTDESC, t1.LONGDESC, t1.MISVALFLG, t1.HELPID

FROM (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='INSP\_WCAND' AND (FIELD\_NAME='agency\_priority')) t1

LEFT JOIN (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='PON\_INSP\_WORKCAND' AND (FIELD\_NAME='IWC\_AGENCY\_PRIORITY')) t2 ON t1.PARMVALUE = t2.PARMVALUE

WHERE t2.FIELD\_NAME IS NULL;

INSERT INTO PARAMTRS(TABLE\_NAME, FIELD\_NAME, PARMVALUE, SHORTDESC, LONGDESC, MISVALFLG, HELPID)

SELECT 'PON\_INSP\_WORKCAND', 'IWC\_STATUS', t1.PARMVALUE, t1.SHORTDESC, t1.LONGDESC, t1.MISVALFLG, t1.HELPID

FROM (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='INSP\_WCAND' AND (FIELD\_NAME='agency\_status')) t1

LEFT JOIN (SELECT \* FROM PARAMTRS WHERE TABLE\_NAME='PON\_INSP\_WORKCAND' AND (FIELD\_NAME='IWC\_STATUS')) t2 ON t1.PARMVALUE = t2.PARMVALUE

WHERE t2.FIELD\_NAME IS NULL;

END IF;

END;

/

-- Set PON\_INSP\_WORKCAND.IWC\_WORKCAND\_NAME = IWC\_WORKCAND\_ID WHERE IWC\_WORKCAND\_NAME IS NULL;

UPDATE PON\_INSP\_WORKCAND

SET IWC\_WORKCAND\_NAME = IWC\_WORKCAND\_ID

WHERE IWC\_WORKCAND\_NAME IS NULL;

/

-- Convert DDMMSS lat/lon to DDDDDD to precise\_lat and precise\_lon if they are null.

DECLARE

latitude NUMERIC(8,2); longitude NUMERIC(9,2); latStr VARCHAR(10); lonStr VARCHAR(10); deg VARCHAR(3); minutes VARCHAR(2); sec VARCHAR(5);

decLat FLOAT; decLon FLOAT; negLatFlag INT; negLonFlag INT;

CURSOR latlon\_cursor IS

SELECT latitude, longitude, precise\_lat, precise\_lon

FROM bridge

FOR UPDATE of precise\_lat, precise\_lon;

BEGIN

OPEN latlon\_cursor;

LOOP

FETCH latlon\_cursor INTO latitude, longitude, decLat, decLon;

EXIT WHEN latlon\_cursor%NOTFOUND;

IF(latitude <> -1.00 AND longitude <> -1.00 AND decLat IS NULL AND decLon IS NULL)

THEN

negLatFlag := 0;

negLonFlag := 0;

IF (latitude < 0)

THEN

latitude := 0 - latitude;

negLatFlag := 1;

END IF;

IF (longitude < 0)

THEN

longitude := 0 - longitude;

negLonFlag := 1;

END IF;

----pad zeroes

latStr := to\_char(latitude, 'FM0000000.00');

lonStr := to\_char(longitude, 'FM0000000.00');

----retrieve substr for deg, min, sec for latitude field

deg := SUBSTR(latStr, 1, 3);

minutes := SUBSTR(latStr, 4, 2);

sec := SUBSTR(latStr, 6);

decLat := to\_number(deg, '999.99') + (to\_number(minutes, '99.99') / 60) + (to\_number(sec, '99.99999999999') / 3600);

decLat := ROUND(decLat,8); -- BQ-955 - Limit number of decimal places

----repeat for longitude

deg := SUBSTR(lonStr, 1, 3);

minutes := SUBSTR(lonStr, 4, 2);

sec := SUBSTR(lonStr, 6);

decLon := to\_number(deg, '999.99') + (to\_number(minutes, '99.99') / 60) + (to\_number(sec, '99.99999999999') / 3600);

decLon := ROUND(decLon,8); -- BQ-955 - Limit number of decimal places

----re-apply negative values

IF (negLatFlag = 1)

THEN

decLat := 0 - decLat;

END IF;

IF (negLonFlag = 1)

THEN

decLon := 0 - decLon;

END IF;

----save the values

UPDATE BRIDGE

SET precise\_lat = decLat, precise\_lon = decLon

WHERE CURRENT OF latlon\_cursor;

END IF;

END LOOP;

CLOSE latlon\_cursor;

END;

/

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('MISC. TABLE UPDATES', CURRENT\_TIMESTAMP);

-- Clean up Order Num

BEGIN

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

MERGE INTO PON\_NAV\_TASK t1 USING (Select ROW\_NUMBER() OVER(PARTITION BY PARENT\_TASK\_ID, TAB\_ID ORDER BY ORDER\_NUM) AS row\_number, TAB\_ID, TASK\_ID from PON\_NAV\_TASK WHERE TASK\_ID != 0) t2

ON (t1.TAB\_ID = t2.TAB\_ID AND t1.TASK\_ID = t2.TASK\_ID) WHEN MATCHED THEN UPDATE SET t1.ORDER\_NUM = t2.row\_number;

MERGE INTO PON\_NAV\_TAB t1 USING (Select ROW\_NUMBER() OVER(ORDER BY ORDER\_NUM) AS row\_number, TAB\_ID from PON\_NAV\_TAB) t2

ON (t1.TAB\_ID = t2.TAB\_ID) WHEN MATCHED THEN UPDATE SET t1.ORDER\_NUM = t2.row\_number;

MERGE INTO PON\_NAV\_CONTROL\_GROUP t1 USING (Select ROW\_NUMBER() OVER(PARTITION BY PARENT\_CONTROL\_GROUP\_ID, TAB\_ID, TASK\_ID ORDER BY ORDER\_NUM) AS row\_number, TAB\_ID, TASK\_ID, PARENT\_CONTROL\_GROUP\_ID, CONTROL\_GROUP\_ID from PON\_NAV\_CONTROL\_GROUP) t2

ON (t1.TAB\_ID = t2.TAB\_ID AND t1.TASK\_ID = t2.TASK\_ID AND t1.CONTROL\_GROUP\_ID = t2.CONTROL\_GROUP\_ID) WHEN MATCHED THEN UPDATE SET t1.ORDER\_NUM = t2.row\_number;

MERGE INTO PON\_NAV\_CONTROL t1 USING (Select ROW\_NUMBER() OVER(PARTITION BY CONTROL\_GROUP\_ID, TAB\_ID, TASK\_ID ORDER BY ORDER\_NUM) AS row\_number, TAB\_ID, TASK\_ID, CONTROL\_GROUP\_ID, CONTROL\_ID from PON\_NAV\_CONTROL) t2

ON (t1.TAB\_ID = t2.TAB\_ID AND t1.TASK\_ID = t2.TASK\_ID AND t1.CONTROL\_GROUP\_ID = t2.CONTROL\_GROUP\_ID AND t1.CONTROL\_ID = t2.CONTROL\_ID) WHEN MATCHED THEN UPDATE SET t1.ORDER\_NUM = t2.row\_number;

END IF;

END;

/

UPDATE DATADICT SET CONVERSIONRULES = NULL WHERE TABLE\_NAME = 'roadway'AND COL\_NAME = 'bypasslen';

-- Convert Metric to English.

DECLARE

V\_CUR SYS\_REFCURSOR;

V\_X PLS\_INTEGER;

BEGIN

IF GET\_VAR('CONVERT\_METRIC\_TO\_ENGLISH') <> 'OFF' THEN

SELECT COUNT (\*) INTO V\_X FROM DBDESCRP WHERE COALESCE(UNITS\_OF\_MEASURE, 'ME') = 'EN';

IF V\_X = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Convert metric to english');

-- Update columns from metric to English.

DECLARE

V\_CUR SYS\_REFCURSOR;

BEGIN

OPEN V\_CUR FOR

SELECT 'UPDATE ' || TABLE\_NAME || ' SET ' || COL\_NAME ||

' = CASE ''' || COALESCE(CONVERSIONRULES, 'NO\_RULE') || ''' ' ||

' WHEN ''L3'' THEN ' ||

' CASE ROUND(' || COL\_NAME || ', 2) ' ||

' WHEN 99.99 THEN 99.99 ' ||

' ELSE ' || COL\_NAME || ' / ' || FACTOR ||

' END ' ||

' WHEN ''L4'' THEN ' ||

' CASE ROUND(' || COL\_NAME || ', 1) ' ||

' WHEN 99.9 THEN 99.9 ' ||

' ELSE ' || COL\_NAME || ' / ' || FACTOR ||

' END ' ||

' WHEN ''L5'' THEN ' ||

' CASE ROUND(' || COL\_NAME || ', 1) ' ||

' WHEN 99.9 THEN 99.9 ' ||

' WHEN 99.8 THEN 99.8 ' ||

' ELSE ' || COL\_NAME || ' / ' || FACTOR ||

' END ' ||

' WHEN ''W1'' THEN ' ||

' CASE ROUND(' || COL\_NAME || ', 1) ' ||

' WHEN 99.9 THEN 99 ' ||

' WHEN 99 THEN 99.9 ' ||

' ELSE ' || COL\_NAME || ' / ' || FACTOR ||

' END ' ||

' ELSE ' || COL\_NAME || ' / ' || FACTOR ||

' END ' ||

' WHERE COALESCE(' || COL\_NAME || ', 0.0) > 0.0'

FROM (

SELECT DD.TABLE\_NAME, DD.COL\_NAME, ME.FACTOR FACTOR, DD.PAIRCODE, REPLACE(CONVERSIONRULES, ' ', '') CONVERSIONRULES

FROM DATADICT DD

JOIN METRIC\_ENGLISH ME ON ME.PAIRCODE = DD.PAIRCODE

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

WHERE COALESCE(DD.PAIRCODE, -1) <> -1

AND ME.FACTOR <> 1

AND ME.FACTOR > 0

AND (

GET\_VAR('CONVERT\_METRIC\_TO\_ENGLISH') = 'ON\_(ONLY\_COLUMNS\_IN\_PON\_DICT)' AND PD.TABLE\_NAME IS NOT NULL

OR GET\_VAR('CONVERT\_METRIC\_TO\_ENGLISH') = 'ON\_(ALL\_COLUMNS\_IN\_DATADICT)'

)

) T

ORDER BY PAIRCODE, TABLE\_NAME, COL\_NAME;

REV(V\_CUR);

CLOSE V\_CUR;

END;

MERGE INTO PON\_ELEM\_INSP T

USING (

SELECT

EI.BRKEY, EI.INSPKEY, EI.ELEM\_KEY, EI.ELEM\_PARENT\_KEY, EI.ENVKEY, EI.STRUNITKEY,

ME.FACTOR,

EI.ELEM\_QUANTITY,

EI.ELEM\_QTYSTATE1,

EI.ELEM\_QTYSTATE2,

EI.ELEM\_QTYSTATE3,

EI.ELEM\_QTYSTATE4

FROM PON\_ELEM\_INSP EI

JOIN PON\_ELEM\_DEFS DEFS ON DEFS.ELEM\_KEY = EI.ELEM\_KEY

JOIN METRIC\_ENGLISH ME ON ME.PAIRCODE = DEFS.ELEM\_PAIRCODE

WHERE EI.ELEM\_QUANTITY > 0

AND ME.PAIRCODE <> -1

AND ME.FACTOR <> 1

AND ME.FACTOR > 0

) S

ON (

T.BRKEY = S.BRKEY AND

T.INSPKEY = S.INSPKEY AND

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

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

T.ENVKEY = S.ENVKEY AND

T.STRUNITKEY = S.STRUNITKEY

)

WHEN MATCHED THEN

UPDATE SET

T.ELEM\_QUANTITY = CASE WHEN S.ELEM\_QUANTITY > 0 THEN S.ELEM\_QUANTITY / S.FACTOR ELSE S.ELEM\_QUANTITY END,

T.ELEM\_QTYSTATE1 = CASE WHEN S.ELEM\_QTYSTATE1 > 0 THEN S.ELEM\_QTYSTATE1 / S.FACTOR ELSE S.ELEM\_QTYSTATE1 END,

T.ELEM\_QTYSTATE2 = CASE WHEN S.ELEM\_QTYSTATE2 > 0 THEN S.ELEM\_QTYSTATE2 / S.FACTOR ELSE S.ELEM\_QTYSTATE2 END,

T.ELEM\_QTYSTATE3 = CASE WHEN S.ELEM\_QTYSTATE3 > 0 THEN S.ELEM\_QTYSTATE3 / S.FACTOR ELSE S.ELEM\_QTYSTATE3 END,

T.ELEM\_QTYSTATE4 = CASE WHEN S.ELEM\_QTYSTATE4 > 0 THEN S.ELEM\_QTYSTATE4 / S.FACTOR ELSE S.ELEM\_QTYSTATE4 END;

UPDATE DBDESCRP SET UNITS\_OF\_MEASURE = 'EN';

END IF;

END IF;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR CONVERTING METRIC TO ENGLISH ' || SQLERRM);

END;

/

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('CLEAN UP ORDER NUM', CURRENT\_TIMESTAMP);

-- END PART 5 ORACLE