-- =============== DELETE TEMP BLOCK BELOW BEFORE BETA TESTING ================================= --

-- Put your name and the date before any code in this block.

/\* Andrew Bloss 1/25/2015 A manual check before you upgrade.

SELECT \* FROM DBDESCRP;

\*/

-- Andrew Bloss 1/16/2015.

-- Stop the "Too many open cursors" error.

-- Changing the OPEN\_CURSORS paramter does not change the parameter in the current session.

-- Before running this scrip on a customer's machine log in an run this:

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

-- Logout and log back in and the script will not throw a "Too many open cursors" error.

-- When the customer restarts their machine the OPEN\_CURSORS parameter will return to its previous value.

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

-- Andrew Bloss 1/25/2015 Drop constraint in Kentucky 5103.

DECLARE

V\_FK\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_FK\_EXISTS FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'SCRATCH\_FILENET\_IDS' AND CONSTRAINT\_NAME = 'FK\_SCRATCH\_FILENET\_IDS\_BRIDGE';

IF (V\_FK\_EXISTS = 1) THEN

EXECUTE IMMEDIATE 'ALTER TABLE SCRATCH\_FILENET\_IDS DROP CONSTRAINT FK\_SCRATCH\_FILENET\_IDS\_BRIDGE';

END IF;

END;

/

-- Clayton Garcia (01.06.2015) - Delete Controls that may be left over after clean up for Work page

DECLARE

v\_column\_exists number := 0;

BEGIN

Select count(1) into v\_column\_exists from user\_tab\_cols where column\_name = 'PON\_NAV\_CONTROL\_GD' and table\_name = 'PON\_NAV\_CONTROL';

if (v\_column\_exists = 1) then

execute immediate 'DELETE PON\_NAV\_CONTROL WHERE PON\_NAV\_CONTROL\_GD in (''23A2D402BC0741EF84BD9467A90BE75D'', ''F010F8E1DEE7431882ED23C0ECD4618D'', ''06713676A5D34907A38A86510FCB0E5C'',

''450C9755CB6A4097924338984D22175E'', ''7651AA75D8E64C19B8734C17E602C16D'', ''FB2E472C30CD457A963AC4E497DD4B2D'')';

end if;

end;

/

-- Andrew Bloss 1/12/2015 - Drop constraint in Idaho 4X that prevents the primary key on inspevnt from being dropped.

DECLARE

V\_FK\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_FK\_EXISTS FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'ID\_UWINSP' AND CONSTRAINT\_NAME = 'FK\_ID\_UWINSP\_1';

IF (V\_FK\_EXISTS = 1) THEN

EXECUTE IMMEDIATE 'ALTER TABLE ID\_UWINSP DROP CONSTRAINT FK\_ID\_UWINSP\_1';

END IF;

END;

/

-- Andrew Bloss 1/14/2015 - Drop constraint in Iowa 44 that prevents the primary key on bridge from being dropped.

DECLARE

V\_FK\_EXISTS PLS\_INTEGER;

BEGIN

SELECT COUNT(\*) INTO V\_FK\_EXISTS FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'ABW\_ASSET\_XREF' AND CONSTRAINT\_NAME = 'R\_3187';

IF (V\_FK\_EXISTS = 1) THEN

EXECUTE IMMEDIATE 'ALTER TABLE ABW\_ASSET\_XREF DROP CONSTRAINT R\_3187';

END IF;

END;

/

-- Andrew Bloss 1/23/2015 - Drop constraints in Kansas 443 that prevents the primary key on bridge from being dropped.

BEGIN

FOR X IN (

SELECT TABLE\_NAME, CONSTRAINT\_NAME FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'KTAREPORT' AND CONSTRAINT\_NAME = 'FK\_BRKEY' UNION

SELECT TABLE\_NAME, CONSTRAINT\_NAME FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = 'USERNOTE' AND CONSTRAINT\_NAME = 'FK\_USERNOTE\_14\_BRIDGE'

)

LOOP

EXECUTE IMMEDIATE 'ALTER TABLE ' || X.TABLE\_NAME || ' DROP CONSTRAINT ' || X.CONSTRAINT\_NAME;

END LOOP;

END;

/

-- =============== DELETE TEMP BLOCK AVOVE BEFORE BETA TESTING ================================= --

-- ================================================================================================== --

-- \_\_\_ \_\_\_ \_\_\_\_\_ \_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_ \_\_ --

-- / | / | / \_\_\_// / / /\_ \_\_/ \_\_ \ | / /\_\_\_ \_\_\_\_\_\_\_\_\_ --

-- / /| | / /| | \\_\_ \/ /\_/ / / / / / / / | /| / / \_\_ `/ \_\_\_/ \_ \ --

-- / \_\_\_ |/ \_\_\_ |\_\_\_/ / \_\_ / / / / /\_/ /| |/ |/ / /\_/ / / / \_\_/ --

--/\_/ |\_/\_/ |\_/\_\_\_\_/\_/ /\_/ /\_/ \\_\_\_\_/ |\_\_/|\_\_/\\_\_,\_/\_/ \\_\_\_/ --

-- \_\_\_\_ \_ \_\_ \_\_ \_\_\_ \_\_ --

-- / \_\_ )\_\_\_\_\_(\_)\_\_\_/ /\_\_\_ \_\_\_\_ / |/ /\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_ \_\_\_ \_\_\_\_ / /\_ --

-- / \_\_ / \_\_\_/ / \_\_ / \_\_ `/ \_ \ / /|\_/ / \_\_ `/ \_\_ \/ \_\_ `/ \_\_ `/ \_ \/ \_\_ `\_\_ \/ \_ \/ \_\_ \/ \_\_/ --

-- / /\_/ / / / / /\_/ / /\_/ / \_\_/ / / / / /\_/ / / / / /\_/ / /\_/ / \_\_/ / / / / / \_\_/ / / / /\_ --

--/\_\_\_\_\_/\_/ /\_/\\_\_,\_/\\_\_, /\\_\_\_/ /\_/ /\_/\\_\_,\_/\_/ /\_/\\_\_,\_/\\_\_, /\\_\_\_/\_/ /\_/ /\_/\\_\_\_/\_/ /\_/\\_\_/ --

-- /\_\_\_\_/ /\_\_\_\_/ 5.2.2 --

-- ================================================================================================== --

-- Author: Bentley Systems - AASHTOWare Bridge Management Team

-- Create date: 05/22/2013

-- Description: Upgrade any version of AASHTOWare Bridge Management to AASHTOWare Bridge Management 5.2.2

-- ======================================================================================================

/\*

-- Prevent "Too many open cursors" error.

Check the open\_cursors parameter is high enough before running this script on a customer's machine:

SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'open\_cursors';

WRITE DOWN THE VALUE!

If open\_cursors is less than 3000 temporarily increase it:

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

Log out and back in get a session with the new value.

Run the scripts.

Reset open\_cursors:

ALTER SYSTEM SET OPEN\_CURSORS=<THE NUMBER YOU WROTE DOWN!> SCOPE=MEMORY;

If you forget to reset open\_cursors it will be reset when Oracle is restarted.

\*/

/\* This block is for code that removes data or pieces of schema that have not been

given to 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.

The following example checks to see if the table PON\_APP\_USERS exists and checks to see if there is a Pontis user.

It uses EXECUTE IMMEDIATE so the select from PON\_APP\_USERS is only parsed if the table exists. The query of USER\_TABLES

uses SELECT COUNT() INTO V\_X because the table USER\_TABLES always exists.

can be access

DECLARE

V\_X PLS\_INTEGER;

V\_Q VARCHAR2(300);

V\_USERID VARCHAR2(30) := 'Pontis';

BEGIN

SELECT COUNT(\*) INTO V\_X FROM USER\_TABLES WHERE TABLE\_NAME = 'PON\_APP\_USERS';

IF V\_X = 1 THEN

V\_Q := 'SELECT COUNT (\*) INTO :A FROM PON\_APP\_USERS WHERE USERID = :USERID';

EXECUTE IMMEDIATE V\_Q INTO V\_X USING V\_USERID;

DBMS\_OUTPUT.PUT\_LINE('There are ' || V\_X || ' rows in PON\_APP\_USERS with the USERID ' || V\_USERID || '.');

END IF;

EXCEPTION WHEN OTHERS THEN

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

END;

/

USE EXECUTE IMMEDIATE TO EXECUTE SQL THAT REFERENCES TABLES OR COLUMNS THAT MAY NOT EXIST.

OTHERWISE YOUR CODE MAY ERROR.

You can not use EXECUTE IMMEDIATE ... USING to build data definition statements. This will not work.

EXECUTE IMMEDIATE 'DROP INDEX :INDEX' USING V\_INDEX\_NAME;

Do this instead:

V\_Q := 'DROP INDEX ' || V\_INDEX\_NAME;

EXECUTE IMMEDIATE V\_Q;

\*/

/\*

Search for:

REMOVE ITEMS FROM PRODUCTION DATABASES

to locate where to remove items from a production database.

\*/

/\*

SELECT \* FROM DBDESCRP;

Editing this document

USE UPPER CASE FOR ENTRIES IN PON\_DICT, PON\_FK, PON\_INDEX, AND PON\_TABLE.

Use VARCHAR2 or CHAR for character.

Oracle reserves the right to change the behaviour of VARCHAR to match the ANSI standard. They recommend

VARCHAR2.

Use NUMBER(precision, scale).

Specify precision and scale or they will default to 38, 0, which is much more than we need.

Precision is the length of a number and scale is the number of decimal places.

INT and FLOAT are synonyms for NUMBER. INT uses the default 38 decimal digits of precision.

FLOATs have a precision of 126 binary digits. Oracle does not have a native binary integer that can

be stored in the database. The PLS\_INTEGER is only for calculations in PL/SQL (NOT SQL) and can not be

stored in the database.

Oracle recommends BINARY\_FLOAT (32-bits) and BINARY\_DOUBLE (64-bits) for floating point numbers that

can be stored in the database.

\*/

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;

DECLARE

V\_I PLS\_INTEGER;

V\_Q VARCHAR2(200);

BEGIN

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

DBMS\_OUTPUT.PUT\_LINE('RUNNING AASHTOWare Bridge Management 5.2.2 GUID upgrade script.');

SELECT COUNT(\*) INTO V\_I FROM USER\_TAB\_COLUMNS WHERE TABLE\_NAME = 'DBDESCRP' AND COLUMN\_NAME = 'THISDBDESCR';

IF V\_I = 1 THEN

EXECUTE IMMEDIATE 'SELECT THISDBDESCR FROM DBDESCRP WHERE THISDBTIME = (SELECT MAX(THISDBTIME) FROM DBDESCRP)' INTO V\_Q;

DBMS\_OUTPUT.PUT\_LINE('Run on ' || SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') || ' ' || V\_Q || ' on ' || TO\_CHAR(SYSDATE, 'FMDAY MM/DD/YYYY "at" HH24:MI AM') || '.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Run on empty ' || SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') || ' on ' || TO\_CHAR(SYSDATE, 'FMDAY MM/DD/YYYY "at" HH24:MI AM') || '.');

END IF;

END;

/

DECLARE

V\_TEMP PLS\_INTEGER;

V\_Q VARCHAR2(1000);

BEGIN

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

IF v\_temp > 0 THEN

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE PON\_GLOB\_VAR';

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR I1: Failed Dropping Existing Table. [DROP TABLE [PON\_GLOB\_VAR]]; ' || SQLERRM);

DBMS\_OUTPUT.PUT\_LINE('TROUBLESHOOTING ERROR I1 | A PON\_GLOB\_VAR table does exist but can not be deleted. Check the permissions to the table.');

END;

END IF;

v\_q := 'CREATE TABLE PON\_GLOB\_VAR(

VARI VARCHAR2(100) NOT NULL,

VAR\_VALUE VARCHAR2(30) NOT NULL,

CONSTRAINT PON\_GLOB\_VAR\_PK PRIMARY KEY (VARI)

)';

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR I1: Failed Creating Table. [' || v\_q || ']; ' || SQLERRM);

END;

END;

/

BEGIN

-- Save variables into a temp table to make them act like global variables within the script.

-- Insert the values into PON\_GLOB\_VAR

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('PCB', 1); -- Pre-Clean up

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('ARCHIVE\_PON\_ELEM\_INSP', 'DEFAULT'); -- DEFAULT | ALWAYS | NEVER

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('CONVERT\_METRIC\_TO\_ENGLISH', 'ON\_(ONLY\_COLUMNS\_IN\_PON\_DICT)'); -- 'OFF' | 'ON\_(ONLY\_COLUMNS\_IN\_PON\_DICT)' | 'ON\_(ALL\_COLUMNS\_IN\_DATADICT)'

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('TSC', 1); -- Schema update/creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('PUP', 1); -- Grant 'PONTISUSER' permissions to new tables.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('DDV', 1); -- Default Data verification.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('FDV', 1); -- Force Set Default Data.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('SUI', 1); -- Index update/creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('SPK', 1); -- Primary Key creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('SFK', 1); -- Foreign Key creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('SVW', 1); -- Views creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('STG', 1); -- Triggers creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('PDR', 1); -- Procedures creation.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('PCA', 0); -- Post-Clean up.

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('HAS\_GUIDS', CASE WHEN EXISTS(SELECT 1 FROM USER\_TAB\_COLUMNS WHERE COLUMN\_NAME = 'BRIDGE\_GD')

OR NOT EXISTS(SELECT 1 FROM USER\_TABLES WHERE TABLE\_NAME = 'BRIDGE') THEN 1 ELSE 0 END);

INSERT INTO PON\_GLOB\_VAR (VARI, VAR\_VALUE) VALUES ('NEW\_INSTALL', CASE WHEN NOT EXISTS(SELECT 1 FROM USER\_TABLES WHERE TABLE\_NAME = 'BRIDGE') THEN 1 ELSE 0 END);

END;

/

-- GET\_VAR - Return value (var\_value) for a key (vari) in pon\_glob\_var.

CREATE OR REPLACE FUNCTION GET\_VAR(V\_KEY VARCHAR2)

RETURN VARCHAR2 AS

V\_RETURN VARCHAR(30);

BEGIN

SELECT VAR\_VALUE INTO V\_RETURN FROM PON\_GLOB\_VAR WHERE VARI = V\_KEY;

RETURN V\_RETURN;

EXCEPTION WHEN OTHERS THEN RETURN NULL;

END;

/

-- TABLE\_EXISTS - Return 1 or 0.

CREATE OR REPLACE FUNCTION TABLE\_EXISTS(V\_KEY VARCHAR2)

RETURN VARCHAR2 AS

V\_RETURN NUMERIC(1);

BEGIN

SELECT COUNT(\*) INTO V\_RETURN FROM USER\_TABLES WHERE TABLE\_NAME = V\_KEY;

RETURN V\_RETURN;

END;

/

-- REV - Fetch and execute each line of a cursor.

CREATE OR REPLACE PROCEDURE REV (

P\_CURSOR SYS\_REFCURSOR,

P\_PRINT PLS\_INTEGER DEFAULT 0,

P\_ERROR VARCHAR2 DEFAULT 'ERROR : '

)

AS

L\_Q VARCHAR2(32767);

BEGIN

FETCH P\_CURSOR INTO L\_Q;

WHILE P\_CURSOR%FOUND

LOOP

BEGIN

EXECUTE IMMEDIATE L\_Q;

IF P\_PRINT = 1 THEN

DBMS\_OUTPUT.PUT\_LINE(L\_Q);

END IF;

EXCEPTION WHEN OTHERS THEN

IF P\_PRINT = 1 THEN

DBMS\_OUTPUT.PUT\_LINE(P\_ERROR || ' ' || SQLERRM || ' in ' || SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') || '.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(P\_ERROR || ' ' || L\_Q || '; ' || SQLERRM || ' in ' || SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') || '.');

END IF;

END;

FETCH P\_CURSOR INTO L\_Q;

END LOOP;

END;

/

-- IS\_INTEGER - Return 1 if the input is an integer, 0 if it is not.

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;

/

-- PON\_CONCAT - Return the text of a single column cursor rolled up into one line and separated by P\_IN\_BETWEEN.

/\*

Returns

1

2

3

as 1, 2, 3.

\*/

CREATE OR REPLACE FUNCTION PON\_CONCAT(P\_CURSOR SYS\_REFCURSOR, P\_IN\_BETWEEN VARCHAR2 DEFAULT ', ')

RETURN VARCHAR2

IS

L\_RETURN VARCHAR2(32767);

L\_TEMP VARCHAR2(32767);

BEGIN

LOOP

FETCH P\_CURSOR INTO L\_TEMP;

EXIT WHEN P\_CURSOR%NOTFOUND;

L\_RETURN := L\_RETURN || P\_IN\_BETWEEN || L\_TEMP;

END LOOP;

RETURN SUBSTR(L\_RETURN, LENGTH(P\_IN\_BETWEEN) + 1, 32767);

CLOSE P\_CURSOR;

END;

/

-- F\_READ\_DATA\_DEFAULT - Return VARCHAR2 of USER\_TAB\_COLS.DATA\_DEFAULT for comparison in SQL statements.

-- Oracle 11g Data\_default is of type LONG and can not be compared to VARCHAR in SQL.

CREATE OR REPLACE FUNCTION F\_READ\_DATA\_DEFAULT (P\_TABLE\_NAME IN VARCHAR2, P\_COLUMN\_NAME IN VARCHAR2) RETURN VARCHAR2 IS

L\_TEXT VARCHAR2(32767);

BEGIN

SELECT DATA\_DEFAULT

INTO L\_TEXT

FROM USER\_TAB\_COLS

WHERE TABLE\_NAME = P\_TABLE\_NAME AND COLUMN\_NAME = P\_COLUMN\_NAME;

RETURN SUBSTR(L\_TEXT,1,4000);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN '-1';

END;

/

-- F\_READ\_SEARCH\_CONDITION - Return VARCHAR2 of USER\_CONSTRAINTS.SEARCH\_CONDITION for comparison in SQL statements.

-- Oracle 11g Search\_Condition is of type LONG and can not be compared to VARCHAR2 in SQL.

CREATE OR REPLACE FUNCTION F\_READ\_SEARCH\_CONDITION (P\_CONSTRAINT\_NAME IN VARCHAR2) RETURN VARCHAR2 IS

L\_TEXT VARCHAR2(32767);

BEGIN

SELECT SEARCH\_CONDITION

INTO L\_TEXT

FROM USER\_CONSTRAINTS

WHERE CONSTRAINT\_NAME = P\_CONSTRAINT\_NAME;

RETURN SUBSTR(L\_TEXT,1,4000);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

/

/\*

Tell users if a Pontis user will be added.

If there is no Pontis user (Userkey = 1 and upper(UserId) = 'PONTIS) then one will be added with a password of 'Pontis'.

A Pontis user exists if there is one in PON\_APP\_USERS or one in USERS that will be imported into PON\_APP\_USERS.

Users are imported from USERS to PON\_APP\_USERS if PON\_APP\_USERS is empty or does not exist.

\*/

DECLARE

V\_PON\_APP\_USER\_TABLE\_EXISTS PLS\_INTEGER := 0;

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

V\_PON\_APP\_USERS\_COUNT PLS\_INTEGER := 0;

V\_USER\_TABLE\_EXISTS PLS\_INTEGER := 0;

BEGIN

SELECT COUNT(\*) INTO V\_PON\_APP\_USER\_TABLE\_EXISTS FROM USER\_TABLES WHERE TABLE\_NAME = 'PON\_APP\_USERS';

IF V\_PON\_APP\_USER\_TABLE\_EXISTS = 1 THEN

EXECUTE IMMEDIATE 'SELECT COUNT(\*)

FROM PON\_APP\_USERS WHERE USERKEY = 1 AND UPPER(USERID) = ''PONTIS'''

INTO V\_PONTIS\_USER\_EXISTS;

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM PON\_APP\_USERS'

INTO V\_PON\_APP\_USERS\_COUNT;

END IF;

--If the Pontis user does not exist and there are no rows in PON\_APP\_USERS

--(because it does not exist or is empty) then a Pontis user (if one exists)

--will be imported from USERS.

IF V\_PONTIS\_USER\_EXISTS = 0 AND V\_PON\_APP\_USERS\_COUNT = 0 THEN

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

IF V\_USER\_TABLE\_EXISTS = 1 THEN

EXECUTE IMMEDIATE 'SELECT COUNT(\*)

FROM USERS WHERE USERKEY = ''1'' AND UPPER(USERID) = ''PONTIS'''

INTO V\_PONTIS\_USER\_EXISTS;

END IF;

END IF;

IF V\_PONTIS\_USER\_EXISTS = 0 THEN

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

DBMS\_OUTPUT.PUT\_LINE('A DEFAULT USER CALLED ''PONTIS'' HAS OR WILL BE ADDED WITH A PASSWORD OF ''pontis''.');

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

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

END IF;

END;

/

-- REFORMAT\_CUSTOM - Increment primary key number values by 15000 in rows added by users to tables that have standard setup data.

-- This prevents user added rows from being overwritten by standard setup data.

CREATE OR REPLACE PROCEDURE ReformatCustom(I\_REFORMAT\_TABLE IN VARCHAR2,

I\_REFORMAT\_COL IN VARCHAR2)

AS

V\_HAS\_ROWS\_TO\_REFORMAT PLS\_INTEGER;

TYPE T\_COLS\_TABLE IS TABLE OF VARCHAR2(30);

V\_PRECEDING\_COLS\_TABLE T\_COLS\_TABLE;

V\_REFORMAT\_COL\_TABLE T\_COLS\_TABLE;

TYPE T\_Q\_TABLE IS TABLE OF VARCHAR2(3000);

V\_Q\_TABLE T\_Q\_TABLE;

V\_Q VARCHAR2(3000);

V\_X PLS\_INTEGER;

V\_ADD\_TO\_SOURCE PLS\_INTEGER;

V\_NEXT\_ID PLS\_INTEGER;

V\_PRECEDING\_COLS VARCHAR2(300);-- THE OTHER COLUMNS OF THE PRIMARY KEY.

BEGIN

BEGIN

--Is there something to reformat?

V\_Q := 'SELECT COUNT(\*) FROM ' || I\_REFORMAT\_TABLE || ' WHERE ' || I\_REFORMAT\_COL || ' < 15000 AND (PONTIS\_STANDARD\_IND IS NULL OR PONTIS\_STANDARD\_IND <> ''T'') AND ROWNUM = 1';

EXECUTE IMMEDIATE V\_Q

INTO V\_HAS\_ROWS\_TO\_REFORMAT;

IF V\_HAS\_ROWS\_TO\_REFORMAT = 0 THEN

RETURN;

END IF;

--DBMS\_OUTPUT.PUT\_LINE(V\_HAS\_ROWS\_TO\_REFORMAT);

--Find amount to add to the smallest value in I\_REFORMAT\_COL to reformat it to 15000 (or more if there are already rows that have

--been reformatted).

V\_Q := 'SELECT

(SELECT GREATEST(15000, MAX(' || I\_REFORMAT\_COL || ')) + 1 FROM ' || I\_REFORMAT\_TABLE || ')

-

(SELECT NVL(MIN(' || I\_REFORMAT\_COL || '),0) FROM ' || I\_REFORMAT\_TABLE || ' WHERE ' || I\_REFORMAT\_COL || ' < 15000 AND (PONTIS\_STANDARD\_IND IS NULL OR PONTIS\_STANDARD\_IND <> ''T''))

FROM

DUAL';

EXECUTE IMMEDIATE V\_Q INTO V\_ADD\_TO\_SOURCE;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in ReformatCustom: ' || I\_REFORMAT\_TABLE || ' - ' || I\_REFORMAT\_COL || '; ' || SQLERRM);

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

RETURN;

END;

-- Set V\_PRECEDING\_COLS with a comma separated list of the columns in the primary key of I\_REFORMAT\_TABLE up to the column to be reformatted.

SELECT COLUMN\_NAME BULK COLLECT INTO V\_PRECEDING\_COLS\_TABLE

FROM USER\_CONS\_COLUMNS CC

JOIN USER\_CONSTRAINTS C ON CC.CONSTRAINT\_NAME = C.CONSTRAINT\_NAME

WHERE C.TABLE\_NAME = I\_REFORMAT\_TABLE AND C.CONSTRAINT\_TYPE = 'P';

V\_REFORMAT\_COL\_TABLE := T\_COLS\_TABLE();

V\_REFORMAT\_COL\_TABLE.EXTEND;

V\_REFORMAT\_COL\_TABLE(1) := I\_REFORMAT\_COL;

V\_PRECEDING\_COLS\_TABLE := V\_PRECEDING\_COLS\_TABLE MULTISET EXCEPT V\_REFORMAT\_COL\_TABLE;

IF V\_PRECEDING\_COLS\_TABLE IS NOT EMPTY THEN

FOR I IN V\_PRECEDING\_COLS\_TABLE.FIRST..V\_PRECEDING\_COLS\_TABLE.LAST

LOOP

V\_PRECEDING\_COLS := V\_PRECEDING\_COLS || V\_PRECEDING\_COLS\_TABLE(I) || ', ';

END LOOP;

END IF;

-- Build the update statements to reformat tables related to I\_REFORMAT\_TABLE and collect them in V\_Q\_TABLE.

SELECT DISTINCT

'UPDATE ' || TARGET\_TABLE || ' SET ' || TARGET\_COL || ' = ' || TARGET\_COL || '

+ ' || V\_ADD\_TO\_SOURCE || '

WHERE (' || V\_PRECEDING\_COLS || TARGET\_COL || ') IN (

SELECT ' || V\_PRECEDING\_COLS || I\_REFORMAT\_COL || ' FROM ' || I\_REFORMAT\_TABLE || '

WHERE (PONTIS\_STANDARD\_IND IS NULL OR PONTIS\_STANDARD\_IND <> ''T'') AND ' || I\_REFORMAT\_COL || ' < 15000)'

BULK COLLECT INTO V\_Q\_TABLE

FROM (SELECT P.TABLE\_NAME, P.CONSTRAINT\_NAME, P\_COLS.COLUMN\_NAME, P\_COLS.POSITION

FROM USER\_CONSTRAINTS P --PARENT

JOIN USER\_CONS\_COLUMNS P\_COLS ON P\_COLS.CONSTRAINT\_NAME = P.CONSTRAINT\_NAME

WHERE P.CONSTRAINT\_TYPE IN ('P', 'U') AND P\_COLS.COLUMN\_NAME = I\_REFORMAT\_COL AND P.TABLE\_NAME NOT LIKE '%\\_T' ESCAPE '\') P\_COLS2

JOIN (SELECT R\_CONSTRAINT\_NAME, C.TABLE\_NAME TARGET\_TABLE, C\_COLS.COLUMN\_NAME TARGET\_COL, C\_COLS.POSITION

FROM USER\_CONSTRAINTS C --CHILD

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

JOIN USER\_TABLES CHILD\_TABLE ON CHILD\_TABLE.TABLE\_NAME = C.TABLE\_NAME

WHERE C.CONSTRAINT\_TYPE = 'R' AND C.TABLE\_NAME NOT LIKE '%\\_T' ESCAPE '\') C\_COLS2

ON C\_COLS2.R\_CONSTRAINT\_NAME = P\_COLS2.CONSTRAINT\_NAME AND C\_COLS2.POSITION = P\_COLS2.POSITION;

--Execute the update statements.

IF V\_Q\_TABLE IS NOT EMPTY THEN

FOR I IN V\_Q\_TABLE.FIRST..V\_Q\_TABLE.LAST

LOOP

--DBMS\_OUTPUT.PUT\_LINE(I || '/' || V\_Q\_TABLE.COUNT || ': ' || V\_Q\_TABLE(I));

--DBMS\_OUTPUT.PUT\_LINE(NULL);

BEGIN

EXECUTE IMMEDIATE(V\_Q\_TABLE(I));

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error reformatting ' || V\_Q\_TABLE(I) || '; ' || SQLERRM);

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

ROLLBACK;

RETURN;

END;

END LOOP;

END IF;

-- Reformat I\_REFORMAT\_TABLE.

SELECT DISTINCT

'UPDATE ' || I\_REFORMAT\_TABLE || '

SET ' || I\_REFORMAT\_COL || ' = ' || I\_REFORMAT\_COL || ' + ' || V\_ADD\_TO\_SOURCE || '

' || CASE WHEN EXISTS(SELECT 1 FROM USER\_TAB\_COLS WHERE TABLE\_NAME = I\_REFORMAT\_TABLE AND COLUMN\_NAME = 'ORDER\_NUM') THEN ', ORDER\_NUM = ORDER\_NUM + 15000 ' END || '

WHERE ' || I\_REFORMAT\_COL || ' < 15000 AND (PONTIS\_STANDARD\_IND IS NULL OR PONTIS\_STANDARD\_IND <> ''T'')'

INTO V\_Q

FROM DUAL;

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

BEGIN

EXECUTE IMMEDIATE(V\_Q);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error reformatting ' || I\_REFORMAT\_TABLE || ' and ' || I\_REFORMAT\_COL || ': ' || V\_Q || '; ' || SQLERRM);

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

ROLLBACK;

RETURN;

END;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error at end of reformatting ' || I\_REFORMAT\_TABLE || ' and ' || I\_REFORMAT\_COL || '; ' || SQLERRM);

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

ROLLBACK;

END;

/

-- = A | Dropping temp tables if exist to re-create ===================================== --

DECLARE v\_temp number :=0;

v\_q VARCHAR2(1000);

BEGIN

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

IF v\_temp = 1 THEN

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

IF v\_temp > 0 THEN

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

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR A1: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

IF v\_temp > 0 THEN

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

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR A2: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

IF v\_temp > 0 THEN

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

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR A1: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

IF v\_temp > 0 THEN

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

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR A3: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

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

IF v\_temp > 0 THEN

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

BEGIN

EXECUTE IMMEDIATE v\_q;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR A3: Failed Dropping Existing Table. [' || v\_q || ']; ' || SQLERRM);

END;

END IF;

END IF;

END;

/

-- = B | Create the temp tables schema ================================================== --

DECLARE v\_temp number :=0;

v\_q VARCHAR2(2000);

BEGIN

v\_q := 'CREATE TABLE PON\_TABLE (

TABLE\_NAME VARCHAR2(30) NOT NULL,

SHORT\_NAME VARCHAR(20),

RANK NUMBER(10,0) NOT NULL,

CONSTRAINT PON\_TABLE\_PK PRIMARY KEY (TABLE\_NAME))';

execute immediate v\_q;

v\_q := 'CREATE TABLE PON\_RETIRED\_TABLE (

TABLE\_NAME VARCHAR2(30) NOT NULL,

CONSTRAINT PON\_RETIRED\_TABLE\_PK PRIMARY KEY(TABLE\_NAME))';

execute immediate v\_q;

V\_Q := 'CREATE TABLE PON\_DICT(

TABLE\_NAME VARCHAR(30) NOT NULL,

PK NUMBER(1) DEFAULT 3 NOT NULL CHECK (PK BETWEEN 0 AND 3),

REQUIRED NUMBER(1) NOT NULL,

COL\_ORDER NUMBER(3) NOT NULL,

COL\_NAME VARCHAR(30) NOT NULL,

DATA\_TYPE VARCHAR(30) NOT NULL,

LENGTH NUMBER(4),

SCALE NUMBER(2),

DEF\_VALUE VARCHAR(100),

ID NUMBER(1) DEFAULT 0 NOT NULL,

FORCE\_DEF NUMBER(1) NOT NULL,

ACTIVE\_PK NUMBER(1) NULL, -- NO DEFAULT DATA FOR THIS COLUMN. IT JUST HOLDS THE RESULTS OF A COMPLEX CALCULATION.

NEW\_COL\_OLD\_TABLE NUMBER(1) DEFAULT 0 NOT NULL,

REPLACEMENT\_KEY NUMBER(1) DEFAULT 0 NOT NULL,

CONSTRAINT PON\_DICT\_PK PRIMARY KEY (TABLE\_NAME, COL\_ORDER),

CONSTRAINT PON\_DICT\_UQ UNIQUE (TABLE\_NAME, COL\_NAME),

CONSTRAINT PON\_DICT\_\_PON\_TABLE\_FK FOREIGN KEY (TABLE\_NAME) REFERENCES PON\_TABLE (TABLE\_NAME)

)';

EXECUTE IMMEDIATE V\_Q;

V\_Q :=

'CREATE TABLE PON\_FK(

FK\_NAME VARCHAR2(30) NOT NULL,

COL\_ORDER NUMBER(2) NOT NULL,

TABLE\_NAME VARCHAR2(30) NOT NULL,

COL\_NAME VARCHAR2(30) NOT NULL,

R\_TABLE\_NAME VARCHAR2(30) NOT NULL,

R\_COL\_NAME VARCHAR2(30) NOT NULL,

DELETE\_RULE NUMBER(1) NOT NULL,

-- 0 for No Action

-- 1 for Cascade

-- 2 for Set Null

PURPOSE INT DEFAULT 3 NOT NULL CHECK (PURPOSE BETWEEN 1 AND 3),

-- 1 for old keys that are dropped after the upgrade to GUID keys.

-- 2 for GUID keys.

-- 3 for that are used before and after the upgrade.

CREATE\_FK INT DEFAULT 1 NOT NULL, -- 1 for FKs that are created.

-- 0 for PON\_DICT rows used to create merges during the upgrade but are not created.

CONSTRAINT PON\_FK\_PK PRIMARY KEY (FK\_NAME, TABLE\_NAME, COL\_NAME),

CONSTRAINT PON\_FK\_\_PON\_DICT\_FK FOREIGN KEY (TABLE\_NAME, COL\_NAME) REFERENCES PON\_DICT (TABLE\_NAME, COL\_NAME),

CONSTRAINT PON\_FK\_R\_\_PON\_DICT\_FK FOREIGN KEY (R\_TABLE\_NAME, R\_COL\_NAME) REFERENCES PON\_DICT (TABLE\_NAME, COL\_NAME)

)';

EXECUTE IMMEDIATE V\_Q;

v\_q :=

'CREATE TABLE PON\_INDEX (

TABLE\_NAME VARCHAR2(30) NOT NULL,

INDEX\_NAME VARCHAR2(30) NOT NULL,

COL\_NAME VARCHAR2(30) NOT NULL,

COL\_ORDER NUMBER(3) NOT NULL,

IS\_UNIQUE NUMBER(1) NOT NULL,

CONSTRAINT PON\_INDEX\_PK PRIMARY KEY (TABLE\_NAME, INDEX\_NAME, COL\_ORDER),

CONSTRAINT PON\_INDEX\_\_PON\_DICT\_FK FOREIGN KEY (TABLE\_NAME, COL\_NAME) REFERENCES PON\_DICT (TABLE\_NAME, COL\_NAME)

)';

execute immediate v\_q;

--INSERT INTO PROFILING(PLACE, END\_TIME) VALUES ('DROP TABLE AND OTHER STUFF', CURRENT\_TIMESTAMP);

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error creating table: ' || v\_q);

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

-- end part 1 Oracle