PL/SQL Examples and Guide

Oracle and PostGreSQL:

·Oracle: (function, cursor, record, vars, arg, count, dbms)

CREATE OR REPLACE **FUNCTION** Fun2 (I VARCHAR2)

RETURN VARCHAR2

IS

PRAGMA AUTONOMOUS\_TRANSACTION; -- 🡨 this allows to have a cursor interacting with inner DMLs

cursor c is select \* from t1 where a1 >= 0;

r c%rowtype; -- 🡨 this emulates a Record

cont number := 0;

value varchar2(100) := -1;

BEGIN

open c;

loop

fetch c into r;

exit when c%notfound;

cont := cont + 1;

dbms\_output.put\_line(cont || ' --->' || r.a2 || ',' || r.a1); -- 🡨 clause RETURN overwrites this

SELECT LOWER(A2) into value FROM T1 WHERE A1 = 0;

insert into T1 values (20 + cont, (r.a2 || ',' || r.a1));

commit; -- 🡨 this is necessary to avoid error: “\* see below”

end loop;

close c;

RETURN lower(r.a2) ||'-'|| r.a1 || 'cont: ' || cont;

END;

/

Select Fun2('string') from dual;

“\*” : ORA-06519: active autonomous transaction detected and rolled back

ORA-06512: at "SYSTEM.FUN2", line 21

·Oracle: (procedure, cursor, record, vars, arg, count, dbms)

CREATE OR REPLACE **PROCEDURE** Pun2 (I VARCHAR2)

-- RETURN VARCHAR2

IS

PRAGMA AUTONOMOUS\_TRANSACTION; -- 🡨 this allows to have a cursor interacting with inner DMLs

cursor c is select \* from t1 where a1 >= 0;

r c%rowtype; -- 🡨 this emulates a Record

cont number := 0;

value varchar2(100) := -1; -- 🡨 MUST ALWAYS BE DEFINED WITH LENGTH (EG: VARCHAR2(XXXXX) )

BEGIN

open c;

loop

fetch c into r;

exit when c%notfound;

cont := cont + 1;

dbms\_output.put\_line(cont || ' --->' || r.a2 || ',' || r.a1);

SELECT LOWER(A2) into value FROM T1 WHERE A1 = 0;

insert into T1 values (20 + cont, (r.a2 || ',' || r.a1));

commit; -- 🡨 this is necessary to avoid error: “\*” see below…

end loop;

close c;

-- RETURN lower(r.a2) ||'-'|| r.a1 || 'cont: ' || cont;

END;

/

exec Pun2('-----');

“\*” : ORA-06519: active autonomous transaction detected and rolled back

ORA-06512: at "SYSTEM.FUN2", line 21

·PostGreSQL: (function, cursor, record, vars, arg, count, raise)

CREATE OR REPLACE FUNCTION FUN2

(VARR1 IN VARCHAR) -- 🡨 see PostGreSQL only know VARCHAR (NOT VARCHAR2)

RETURNS VARCHAR

LANGUAGE PLPGSQL -- 🡨 this shit is only for PostGreSQL

AS

$$

DECLARE -- 🡨 DECLARE is only used in named functions and procedures in PostGreSQL

C CURSOR FOR select \* from A where a1 = 'A' AND a2 like '%f%';

R RECORD;

CONT INTEGER := 0;

VALUE VARCHAR := ''; -- 🡨 VARCHAR DOES NOT NEED TO BE DECLARED WITH LENGTH (EG VARCHAR(100) )

BEGIN

OPEN C;

LOOP

FETCH C INTO R;

EXIT WHEN NOT FOUND;

CONT := CONT + 1;

SELECT LOWER(A2) INTO VALUE FROM A WHERE a1 = 'A' AND a2 like '%f%';

SELECT \* INTO R FROM A WHERE a1 = 'A' AND a2 like '%f%'; -- 🡨 SELECT \* MAKE R TO MEET TABLE’S COLS NUM

INSERT INTO A VALUES (30 + CONT, (R.A1 || '\_\_\_\_\_\_\_' || R.A2));

RAISE NOTICE 'CONT: ---> %,{%,%}', CONT, R.A1, R.A2; -- 🡨 raise is overwritten by RETURN

RETURN VARR1||','||CONT||','||VALUE||' , R.A2='||R.A2||' , R.A1='||R.A1;

END LOOP;

CLOSE C;

COMMIT;

END;

$$

SELECT FUN2('X');

·PostGreSQL: (procedure, cursor, record, vars, arg, count, raise)

CREATE OR REPLACE PROCEDURE PUN2

(VARR1 IN VARCHAR)

--RETURNS VARCHAR

LANGUAGE PLPGSQL

AS

$$

DECLARE

C CURSOR FOR select \* from A where a1 = 'A' AND a2 like '%f%';

R RECORD;

CONT INTEGER := 0;

VALUE VARCHAR := '';

BEGIN

OPEN C;

LOOP

FETCH C INTO R;

EXIT WHEN NOT FOUND;

CONT := CONT + 1;

SELECT LOWER(A2) INTO VALUE FROM A WHERE a1 = 'A' AND a2 like '%f%';

SELECT \* INTO R FROM A WHERE a1 = 'A' AND a2 like '%f%';

INSERT INTO A VALUES (40 + CONT, (R.A1 || '\_\_\_\_\_\_\_' || R.A2));

RAISE NOTICE 'CONT: ---> %,{%,%}', CONT, TRIM(R.A1),TRIM(R.A2);

-- RETURN VARR1||','||CONT||','||VALUE||' , R.A2='||R.A2||' , R.A1='||R.A1;

END LOOP;

CLOSE C;

commit;

END;

$$

CALL PUN2('X')