**Simple commands:**

create table employees(

employee\_id number(10),

salary number(10));

insert into employees values(100,2000);

insert into employees values(200,52000);

insert into employees values(300,267000);

ALTER TABLE employees add department\_id number(10)

constraint c1 check(department\_id in(101,102));

insert into employees values(201,4543,101);

insert into employees values(202,455443,102);

insert into employees values(203,454543,102);

insert into employees values(204,4578543,102);

alter table employees modify department\_id number(14);

alter table employees modify salary number(14) not null primary key;

-------U CANNOT MODIFY THE CONSTRAINTS. U HAVE TO DROP AND CREATE AGAIN

alter table employees modify constraint c1 check(department\_id > 0);

http://sqlfiddle.com/#!4/5edec

create table employees(

employee\_id number(10) unique,

salary number(10));

insert into employees values(100,2000);

insert into employees values(200,52000);

insert into employees values(300,267000);

ALTER TABLE employees add department\_id number(10)

constraint c1 check(department\_id in(101,102));

insert into employees values(201,4543,101);

insert into employees values(202,455443,102);

insert into employees values(203,454543,102);

insert into employees values(204,4578543,102);

alter table employees modify department\_id number(14);

alter table employees modify salary number(14) not null;

alter table employees DROP constraint c1;

ALTER TABLE EMPLOYEES ADD COnstraint c1 check(department\_id > 0);

create table customers(

customer\_id number(10)

references employees(employee\_id),

bill integer

);

alter table employees rename to emp;

alter table emp rename column salary to e\_salary

**REVERSE A NUMBER**

declare

num number(10) default 123;

rev number(10) default 0;

digit number(10);

begin

while(num > 0) loop

digit:=mod(num,10);

rev:=(rev\*10)+digit;

num:=floor(num/10);

end loop;

insert into output values (rev);

end;

/

select \* from output

/

13/02/2018

**PROCEDURE TO SQUARE A NUMBER**

create or replace procedure proc(v\_num in integer) as

v\_sqr integer;

begin

v\_sqr:=(v\_num\*v\_num);

DBMS\_OUTPUT.PUT\_LINE(v\_sqr);

end;

/

EXECUTE PROC(25);

**2ND METHOD**

create table output(

v\_num integer

);

create or replace procedure proc(v\_num in integer,v\_sqr out integer) as

begin

v\_sqr:=(v\_num\*v\_num);

end;

/

declare

V\_sqr integer;

begin

proc(5,V\_sqr);

insert into output values(V\_sqr);

end;

/

select \* from output

/

**FUNCTION TO RETURN INCREMENTED SALARY**

create table employees(

employee\_id integer,

salary number(10));

insert into employees values(100,2000);

insert into employees values(200,52000);

insert into employees values(300,267000);

ALTER TABLE employees add department\_id number(10)

constraint c1 check(department\_id in(101,102));

insert into employees values(201,4543,101);

insert into employees values(202,455443,102);

insert into employees values(203,454543,102);

insert into employees values(204,4578543,102);

create table output(

v\_num integer

);

CREATE OR REPLACE function func(v\_emp in integer) return integer as

v\_sal integer;

sal integer;

begin

select salary into v\_sal from employees where employee\_id = v\_emp;

sal := v\_sal + 100;

return sal;

end;

/

declare

salry integer;

begin

salry:=FUNC(201);

insert into output values(salry);

end;

/

select \* from output

/

**2) don’t use number(10) in function parameter and return datatype**

CREATE OR REPLACE function func(v\_emp in number) return number as

v\_sal number(10);

sal number(10);

begin

  select salary into v\_sal from a where id = v\_emp;

  sal := v\_sal + 100;

  insert into output values (sal);

  return sal;

end;

/

declare

salry number(10);

begin

salry:=FUNC(1);

insert into output values(salry);

end;

/

select \* from output

/

**CURSOR:**

create table employees(

employee\_id integer,

salary integer);

insert into employees values(100,2000);

insert into employees values(200,52000);

insert into employees values(300,267000);

ALTER TABLE employees add department\_id integer

constraint c1 check(department\_id in(101,102));

insert into employees values(201,4543,101);

insert into employees values(202,455443,102);

insert into employees values(203,454543,102);

insert into employees values(204,4578543,102);

create table output(

emp integer,

v\_sal integer,

v\_dept integer

);

declare

cursor cur is select \* from employees;

v\_emp integer;

v\_sal integer;

v\_dept integer;

begin

open cur;

loop

exit when cur%notfound;

fetch cur into v\_emp, v\_sal, v\_dept;

insert into output values(v\_emp, v\_sal, v\_dept);

end loop;

close cur;

end;

/

select \* from output

/

**SEQUENCE**

CREATE TABLE Persons (

ID int NOT NULL AUTO\_INCREMENT,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (ID)

);

CREATE SEQUENCE seq\_person

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

INSERT INTO Persons (ID,FirstName,LastName)

VALUES (seq\_person.nextval,'Lars','Monsen');

**14/02/2018**

**Trigger:**

create table a(

t\_name varchar(20)

);

create table output(

name varchar(20)

);

create or replace trigger t1

before insert on a

for eACH ROW

begin

insert into output values('Inserting');

end;

/

insert into a values('viditha')

/

select \* from output

/

**Package:**

**PROCEDURE TO SQUARE A NUMBER USING IN OUT PARAMETER**

create table output(

num integer

);

create or replace package pkg is

procedure proc2(v\_num in out integer);

end pkg;

/

create or replace package body pkg is

procedure proc2(v\_num in out integer) is

begin

v\_num := v\_num \* v\_num;

end proc2;

end pkg;

/

declare

v\_res integer;

begin

v\_res := 25;

pkg.proc2(v\_res);

insert into output values(v\_res);

end;

/

select \* from output

/

**Function in a package to square a number**

create table output(

num integer

);

create or replace package pkg is

FUNCTION proc2(v\_num integer) RETURN INTEGER;

end pkg;

/

create or replace package body pkg is

FUNCTION proc2(v\_num integer) RETURN INTEGER is

V\_RES INTEGER;

begin

v\_RES := v\_num \* v\_num;

RETURN V\_RES;

end proc2;

end pkg;

/

declare

v\_res integer;

begin

v\_res := pkg.proc2(10);

insert into output values(v\_res);

end;

/

select \* from output

/

**PACKAGE – RECURSIVE FUNCTION FOR FACTORIAL**

create table output(

num integer

);

create or replace package pkg is

FUNCTION proc2(v\_num integer) RETURN INTEGER;

end pkg;

/

create or replace package body pkg is

FUNCTION proc2(v\_num integer) RETURN INTEGER is

V\_RES INTEGER;

begin

if v\_num = 1 then

return 1;

else

v\_res := v\_num \* proc2(v\_num-1);

end if;

return v\_res;

end proc2;

end pkg;

/

declare

v\_res integer;

begin

v\_res := pkg.proc2(10);

insert into output values(v\_res);

end;

/

select \* from output

/

**Fabinocii series with function in a package**

create or replace package pkg is

FUNCTION FUNC2 RETURN INTEGER;

end pkg;

/

create or replace package body pkg is

FUNCTION FUNC2 RETURN INTEGER is

A INTEGER :=0;

B INTEGER :=1;

C INTEGER DEFAULT 0;

D INTEGER :=0;

begin

INSERT INTO OUTPUT VALUES(A);

INSERT INTO OUTPUT VALUES(B);

LOOP

D:=D+1;

C:=a+B;

EXIT WHEN D=10;

INSERT INTO OUTPUT VALUES(C);

A:=B;

B:=C;

END LOOP;

return C;

end FUNC2;

end pkg;

/

declare

v\_res integer;

begin

v\_res := pkg.FUNC2;

insert into output values(v\_res);

end;

/

select \* from output

/

Try this: function in package with recursion fabinocci series

create or replace package pkg is

FUNCTION FUNC2(a in integer,b in integer) RETURN INTEGER;

end pkg;

/

create or replace package body pkg is

FUNCTION FUNC2(a in integer,b in integer) RETURN INTEGER is

C INTEGER;

D INTEGER;

begin

if (a>152) then

return c;

else

C:=a+B;

INSERT INTO OUTPUT VALUES(C);

A:=B;

B:=C;

d:=func2(a,b);

END if;

return c;

end;

end FUNC2;

end pkg;

/

declare

A INTEGER :=0;

B INTEGER :=1;

v\_res integer;

begin

INSERT INTO OUTPUT VALUES(A);

INSERT INTO OUTPUT VALUES(B);

v\_res := pkg.FUNC2(0,1);

insert into output values(v\_res);

end;

/

select \* from output

/

**RECORD:**

declare

type rec2 is record(id integer,

name varchar(60),

salary integer);

record2 rec2;

begin

record2.id := 101;

record2.name := 'jhgui';

record2.salary := 98689;

DBMS\_OUTPUT.PUT\_LINE(record2.id);

DBMS\_OUTPUT.PUT\_LINE(record2.NAME);

DBMS\_OUTPUT.PUT\_LINE(record2.SALARY);

END;

/

**COLLECTIONS:**

**VARRAY:**

CREATE TABLE OUTPUT(

NAME varchar(70)

);

declare

type v\_arr is varray(7) of varchar(70);

v\_array v\_arr;

begin

v\_array := v\_arr('viditha','sumanth','high');

insert into output values(v\_array(1));

insert into output values(v\_array(2));

insert into output values(v\_array(3));

end;

/

select \* from output

/

**NESTED TABLE:**

declare

type nestbl is table of integer;

x integer;

tbl2 nestbl;

begin

tbl2 := nestbl(1,4,7,8,9,46,437);

x:=tbl2.count;

for i in 1..x loop

dbms\_output.put\_line(tbl2(i));

end loop;

end;

/

**INDEX BY TABLE(associated array):**

declare

type nestbl is table of integer index by varchar(20);

name varchar(20);

tbl2 nestbl;

begin

tbl2('viditha') := 32656;

tbl2('sumanth') := 2485689;

name := tbl2.first;

while name is not null loop

dbms\_output.put\_line(tbl2(name));

name := tbl2.next(name);

end loop;

end;

/

**Package with function:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PACKAGE PKG2 IS

function func2(v\_num employees.salary%type) return employees.salary%type;

end pkg2;

/

create or replace package body pkg2 is

function func2(v\_num employees.salary%type) return employees.salary%type is

begin

return(v\_num + (v\_num \* 0.2));

end func2;

end pkg2;

/

declare

v\_id employees.employee\_id%type;

v\_sal employees.salary%type;

begin

select employee\_id, func2(salary) into v\_id,v\_sal from employees

where employee\_id = 100;

dbms\_output.put\_line(v\_id || 'has salary' ||v\_sal);

end;

/

Or for testing:

/\*

declare

x double precision;

begin

x := pkg2.func2(7);

dbms\_output.put\_line(x);

end;

/

\*/

15/02/2018

**CURSOR WITH CURSOR ROWTYPE VARIABLE**

SET SERVEROUTPUT ON;

declare

cursor cur is select EMPLOYEE\_ID,SALARY,DEPARTMENT\_ID from employees WHERE DEPARTMENT\_ID = 60;

CUR1 CUR%ROWTYPE;

begin

open cur;

loop

exit when cur%notfound;

fetch cur into CUR1;

DBMS\_OUTPUT.PUT\_LINE(CUR1.EMPLOYEE\_ID || 'HAS SALARY' || CUR1.SALARY);

end loop;

close cur;

end;

/

**SEE THE OUTPUT: WHEN CURSOR DOESNOT RETURN ANY ROWS IT WON’T GIVE ANY ERROR**

SET SERVEROUTPUT ON;

declare

cursor cur is select EMPLOYEE\_ID,SALARY,DEPARTMENT\_ID from employees WHERE DEPARTMENT\_ID = 23060;

CUR1 CUR%ROWTYPE;

begin

open cur;

loop

exit when cur%notfound;

fetch cur into CUR1;

DBMS\_OUTPUT.PUT\_LINE(CUR1.EMPLOYEE\_ID || 'HAS SALARY' || CUR1.SALARY);

end loop;

close cur;

end;

/

**OUTPUT:**

HAS SALARY

PL/SQL procedure successfully completed.

16/02/2018

Exceptions:

<https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/errors.htm#BABGIIBI>

Difference between union and union all: