**Practical 8: Starting with PLSQL-%type and bind variables**

**--Using %type atribute**

declare

v\_ename emp.ename%type;

n\_empno emp.empno%type;

n\_sal emp.sal%type;

begin

v\_ename:=&ename;

select empno,sal into n\_empno,n\_sal from emp where ename=upper(v\_ename);

dbms\_output.put\_line('Name: '||v\_ename||' Empno: '||n\_empno|| ' Salary: '||n\_sal);

end;

/

begin

update emp set sal = sal+100 where sal>2500;

dbms\_output.put\_line('No. of rows updated are '||sql%rowcount);

rollback;

end;

/

**--Bind variables**

variable bind\_1 number;

exec :bind\_1:=10;

begin

dbms\_output.put\_line('The value of bind variable bind\_1 is '||:bind\_1);

end;

/

select :bind\_1+20 from dual;

**Practical 9:Collections and Records**

**--index by table**

declare

type abc is table of number index by binary\_integer;

a abc;

i binary\_integer;

begin

a(1):=10;

a(2):=20;

a(3):=30;

a(4):=40;

for i in 1..a.last

loop

dbms\_output.put\_line(a(i)||a.last||a.first||a.prior(i)||a.next(i)||a.count);

end loop;

end;

/

**--record type**

declare

 type emp\_record\_type is record

 ( ename varchar2(20),

 job varchar2(10),

 sal number(7,2));

 emp\_record emp\_record\_type;

 v\_ename emp\_record.ename%type;

 v\_job emp\_record.job%type;

 n\_sal emp\_record.sal%type;

 begin

 v\_ename:=&v\_ename;

 select job,sal into v\_job,n\_sal from emp where ename=v\_ename;

 dbms\_output.put\_line('Name: '||v\_ename|| ' Job: '||v\_job||' Salary: '||n\_sal);

 dbms\_output.put\_line('No. of records affected: '||sql%rowcount);

 end;

 /

**Practical 10: Cursors**

**--explicit cursor steps**

declare

cursor cur\_emp is select \* from emp where sal>2000;

rec\_cur\_emp cur\_emp%rowtype;

begin

open cur\_emp;

loop

fetch cur\_emp into rec\_cur\_emp;

dbms\_output.put\_line(rec\_cur\_emp.empno||’ ‘||rec\_cur\_emp.ename);

exit when cur\_emp%notfound;

end loop;

close cur\_emp;

end;

**--cursor for loop**

declare

cursor cur\_emp is select \* from emp where sal>2000;

begin

for rec\_cur\_emp in cur\_emp loop

dbms\_output.put\_line(rec\_cur\_emp.empno||’ ‘||rec\_cur\_emp.ename);

end loop;

end;

**Practical 11: Functions and Procedures**

**--function**

create or replace function area\_circle\_func(r number) return number is

a number;

pi constant number(3,2):=3.14;

begin

a:=pi\*r\*r;

dbms\_output.put\_line(‘In the function:The area of circle with radius ‘||r||’ is ‘||a);

return a;

end;

/

**--calling a function**

declare

radius number(1);

a number;

begin

radius:=&radius;

a:=area\_circle\_func(radius);

dbms\_output.put\_line(‘The area of circle with radius ‘||radius||’ is ’||a ||‘.Back to main program’);

End;

/

**--Another way of calling a function**

select area\_circle\_func(2) from dual;

**--procedure**

create or replace procedure area\_circle(r number) is

pi constant number(3,2):=3.14;

area number(8,2);

begin

area:=pi\*r\*r;

dbms\_output.put\_line(‘The area of circle with radius ‘||r||’ is ‘||area);

End;

/

**--calling a procedure**

declare

radius number(1);

begin

area\_circle(&radius);

Dbms\_output.put\_line(‘Back to main program’);

End;

/

**--Another way of calling procedure**

Exec Area\_circle(&radius);

Exec Area\_circle(5);

**-- Procedure with in out parameter**

Create or replace procedure area\_circle\_in\_out(r in number, a out number) is

Pi constant number(3,2):=3.14;

Area number(8,2);

Begin

a:=pi\*r\*r;

Dbms\_output.put\_line(‘In the procedure:The area of circle with radius ‘||r||’is’||a);

End;

/

**--Calling procedure**

Declare

radius number(1);

a number;

Begin

Area\_circle\_in\_out(&radius,a);

Dbms\_output.put\_line(‘Area of circle with radius ‘||radius||’is’||a ||‘.Back to main program’);

End;

/

**Practical 12: Exceptions**

**--Predefined named Exceptions**

set serveroutput on

 declare

 v\_empno emp.empno%type;

v\_ename emp.ename%type;

begin

select empno,ename into v\_empno, v\_ename from emp where empno=&v\_empno;

dbms\_output.put\_line(‘Employee number is ‘||v\_empno || ‘ and name is ‘||v\_ename);

exception

when no\_data\_found then

dbms\_output.put\_line(‘OOPS!!! NO SUCH RECORD’);

end;

**--user defined named exceptions**

Declare

v\_empno emp.empno%type;

v\_ename emp.ename%type;

v\_sal emp.sal%type;

a number;

lo\_sal exception;

begin

a:=&a;

select empno,ename,sal into v\_empno,v\_ename,v\_sal from emp where empno=&v\_empno;

dbms\_output.put\_line(‘employee number is ‘||v\_empno|| ‘ and name is ‘||v\_ename);

dbms\_output.put\_line(‘salary is ‘||v\_sal);

v\_sal:=v\_sal+v\_sal/a;

if v\_sal<2000 then raise lo\_sal;

end if;

dbms\_output.put\_line(‘new salary is ‘||v\_sal);

exception

when lo\_sal then

dbms\_output.put\_line(‘in these times of inflation increase the salary!!!’);

when others then

dbms\_output.put\_line(‘some error!!!god knows what???’);

end;

/

**--user defined exceptions: attaching error code**

Declare

V\_empno emp.empno%type;

V\_ename emp.ename%type;

V\_sal emp.sal%type;

A number;

Lo\_sal exception;

Begin

A:=&a;

Select empno,ename,sal into v\_empno,v\_ename,v\_sal from emp where empno=&v\_empno;

Dbms\_output.put\_line(‘Employee number is ‘||v\_empno|| ‘ and name is ‘||v\_ename);

Dbms\_output.put\_line(‘Salary is ‘||v\_sal);

V\_sal:=v\_sal+v\_Sal/a;

If v\_sal<2000 then raise lo\_sal;

End if;

Dbms\_output.put\_line(‘New Salary is ‘||v\_sal);

Exception

When lo\_sal then Raise\_application\_error(-20001,‘In these times of Inflation INCREASE THE SALARY!!!’);

When others then

Dbms\_output.put\_line(‘Some Error!!!God knows What???’);

End;

**--pragma exception\_init : assigning name and number to unnamed exceptions**

Declare

Dup\_pr\_key exception;

Pragma exception\_init(dup\_pr\_key,-1);

Begin

Insert into emp(empno,ename) values(1111,’ABCD’);

Dbms\_output.put\_line(‘One record inserted’);

Insert into emp(empno,ename) values(1111,’EFGH’);

Dbms\_output.put\_line(‘One more record inserted’);

Exception

When dup\_pr\_key then

Dbms\_output.put\_line(sqlcode||‘ :How come more employees with the same employee number???’);

End;

**Practical 13: Triggers**

**--For trigger**

Create table tbl\_audit(tname varchar2(20),dml\_op varchar2(20),date\_of\_dml date);

**--statement /table level trigger**

create or replace trigger trig\_emp\_audit

after insert or delete or update on emp

begin

if inserting then

insert into tbl\_audit values('EMP','Insert',sysdate);

end if;

if updating then

insert into tbl\_audit values('EMP','Update',sysdate);

end if;

if deleting then

insert into tbl\_audit values('EMP','Delete',sysdate);

end if;

end;

**--for row level trigger**

Create table empcheck(event varchar2(20),level\_check varchar2(20),date\_check date);

**-- row level trigger**

create or replace trigger before\_update\_row\_emp\_sal

before update of sal on emp

for each row

begin

insert into empcheck values('Before update',’Row', sysdate);

dbms\_output.put\_line('Some updation done on Emp Sal column');

end;

update emp set sal=sal+100;

select \* from empcheck;

rollback;

Create or replace trigger trig\_emp\_upper\_name

before insert or update on emp

for each row

begin

if inserting then

:new.ename:=upper(:new.ename);

end if;

if updating then

:new.ename:=upper(:new.ename);

end if;

end;