

# PL/SQL in Oracle 12c

**Arup Nanda**

*Longtime Oracle DBA*

# Agenda

- PL/SQL has evolved over the years
- Adoption is still bit challenged
  - Binding with Java
  - SQL to PL/SQL context switching is slow
- Notable PL/SQL features you should know
- Will not cover 11g new features
- Lots of demos
- Downloadable scripts

# Setup

```
begin
  for i in 1..100000 loop
    insert into accounts values (
      i,
      dbms_random.string('u',30),
      ltrim(to_char(dbms_random.value(100000000,999999999), '999999999' )),
      sysdate - 30*365 - dbms_random.value(1,60*365),
      dbms_random.value(1,100000),
      dbms_random.value(1,10000),
      sysdate - dbms_random.value(1,365*5)
    );
  end loop;
end;
```

Table ACCOUNTS

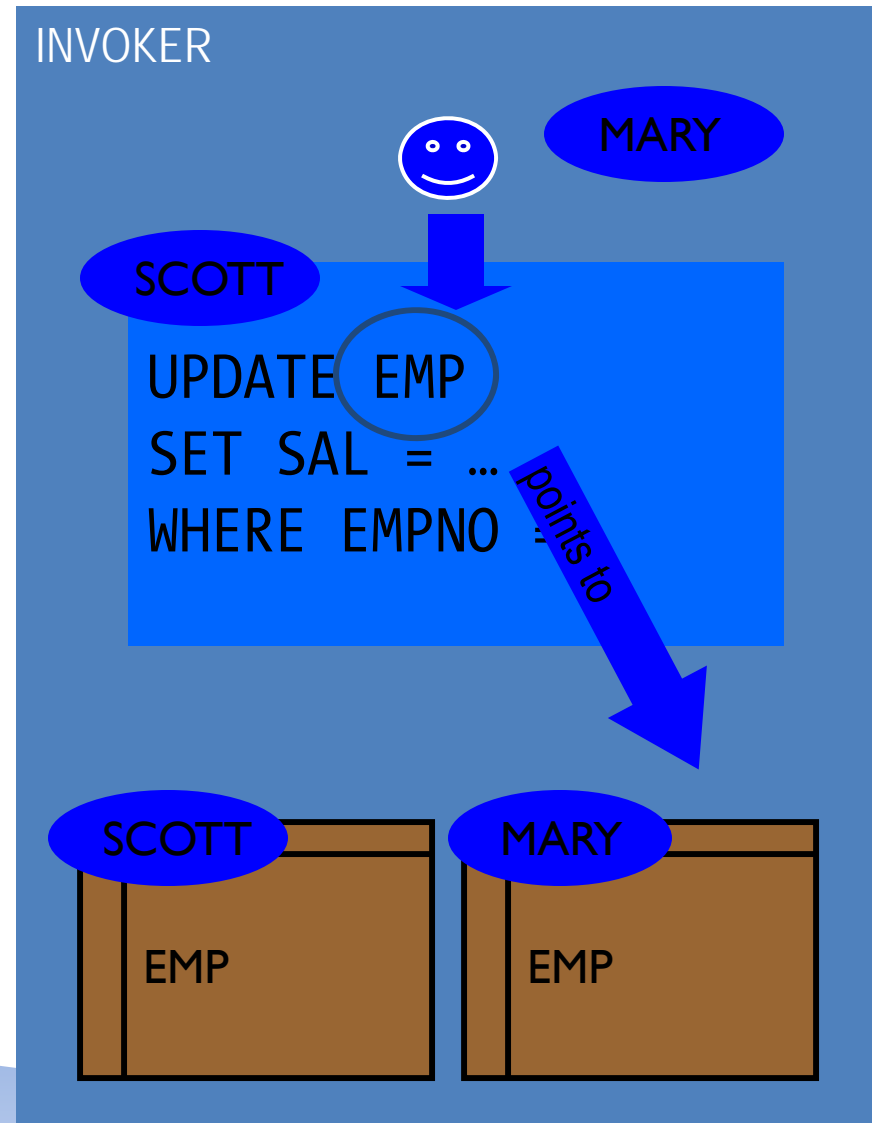
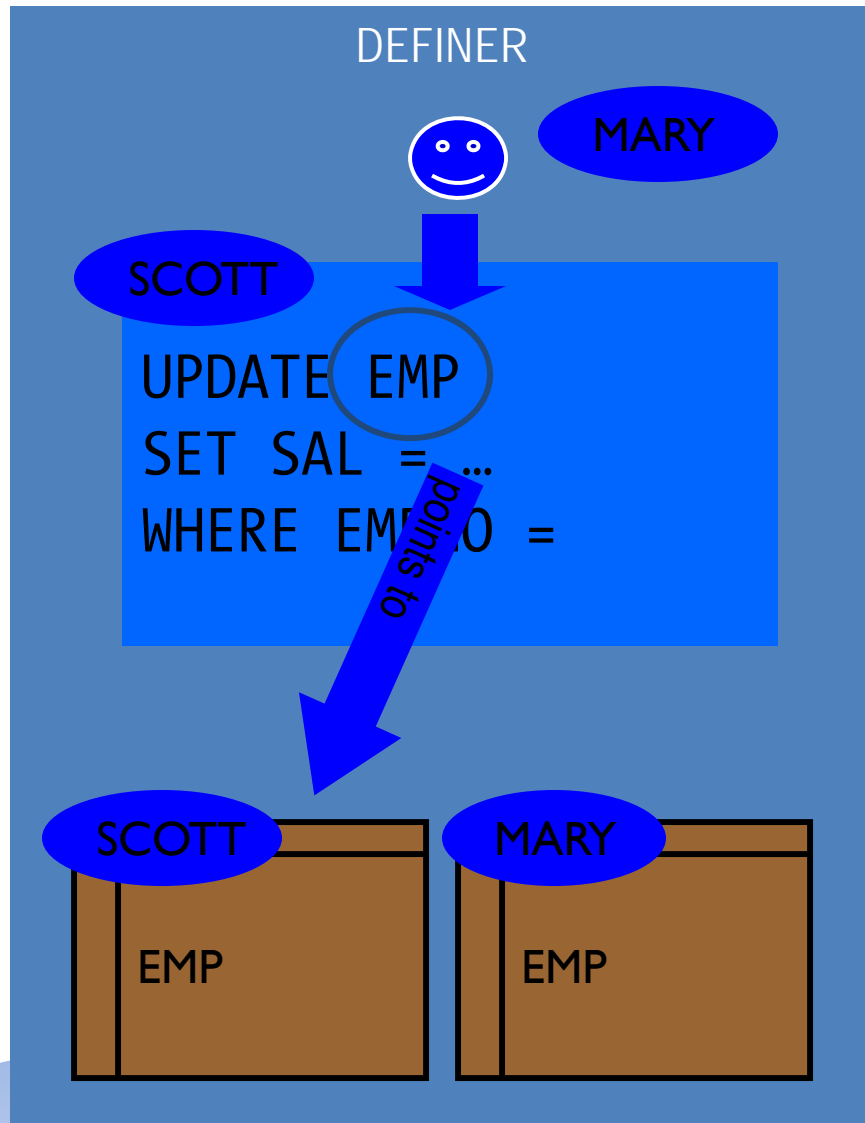
Name	Null?	Type
-----	-----	-----
ACCNO		NUMBER
ACCNAME		VARCHAR2(30)
SSN		VARCHAR2(9)
BIRTHDAY		DATE
PRINCIPAL		NUMBER
INTEREST		NUMBER
CREATED_DT		DATE

# Result Caching

- PL/SQL Function
- Stores the results in result\_cache (a memory area)
- Returns the results when the function is called
  - (not re-executed)
  - When the underlying data hasn't changed

Get\_counts\_by\_birthyear\_v1.sql  
Get\_counts\_by\_birthyear\_v2.sql

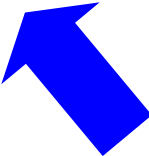
# Invoker Rights



# Invoker Rights

```
CREATE PROCEDURE
  UPDATE_EMP
  ( P_EMPNO IN NUMBER,
    P_SAL   IN NUMBER)
AUTHID DEFINER
IS
BEGIN
  UPDATE EMP
  SET SAL = P_SAL
  WHERE EMPNO = P_EMPNO;
END;
```

```
CREATE PROCEDURE
  UPDATE_EMP
  ( P_EMPNO IN NUMBER,
    P_SAL   IN NUMBER)
AUTHID CURRENT_USER
IS
BEGIN
  UPDATE EMP
  SET SAL = P_SAL
  WHERE EMPNO = P_EMPNO;
END;
```



# Result Cache + Invoker

```
create or replace function get_counts_by_birthyear
(
    p_year_of_birth in varchar2
)
return pls_integer
result_cache
authid current_user
as
    l_cnt pls_integer;
begin
    select count(1)
    into l_cnt
    from arup.accounts
    where to_char(birthday, 'yy') = p_year_of_birth;
    return l_cnt;
end;
/
```

*Schema  
qualification  
required*

*Get\_counts\_by\_birthyear\_v3.sql*

# Repeated Calculations

```
select
    accno,
    0.01 * principal +
    0.02 * interest +
    0.10 * (sysdate - created_dt) * 10 +
    0.05 * (sysdate - birthday) * 10
    as interest
from accounts
```



# Inline Functions

```
create or replace function get_bonus_amount
(
  p_accno in accounts.accno%type
)
return number
is
  l_bonus          number;
begin
  select
    0.01 * principal +
    0.02 * interest +
    0.10 * (sysdate - created_dt) * 10 +
    0.05 * (sysdate - birthday) * 10
  into l_bonus
  from accounts
  where accno = p_accno;
  return l_bonus;
end;
/
```

*Repeated. Better be out of the code in a function*

*Get\_bonus\_amount\_v1.sql  
Int1.sql*

# Inline Function

```
with
function get_bonus_amount
(p_accno in accounts.accno%type)
return number is
    l_bonus          number;
begin
    select
        0.01 * principal +
        0.02 * interest +
        0.10 * (sysdate - created_dt) * 10 +
        0.05 * (sysdate - birthday) * 10
    into l_bonus
    from accounts
    where accno = p_accno;
    return l_bonus;
end;
```

*Not a stored function*

```
select accno, get_bonus_amount (accno)
from accounts
where accno in (3,5,9);
```

*Get\_bonus\_amount\_v2.sql*

# Update

- SQL:  
update table  
set col1 = (with function ...)
- Will fail with ORA-32034: unsupported use of WITH clause
- Solution:  
update /\*+ WITH\_PLSQL \*/ table  
set col1 = (with function ...)

# Inline Functions

- No Schema changes required
  - Good for tightly controlled shops
- No SQL-PL/SQL context switching
- Function inlining
  - PL/SQL Optimization
- Watch out
  - No DETERMINISTIC function optimization

# Pragma UDF

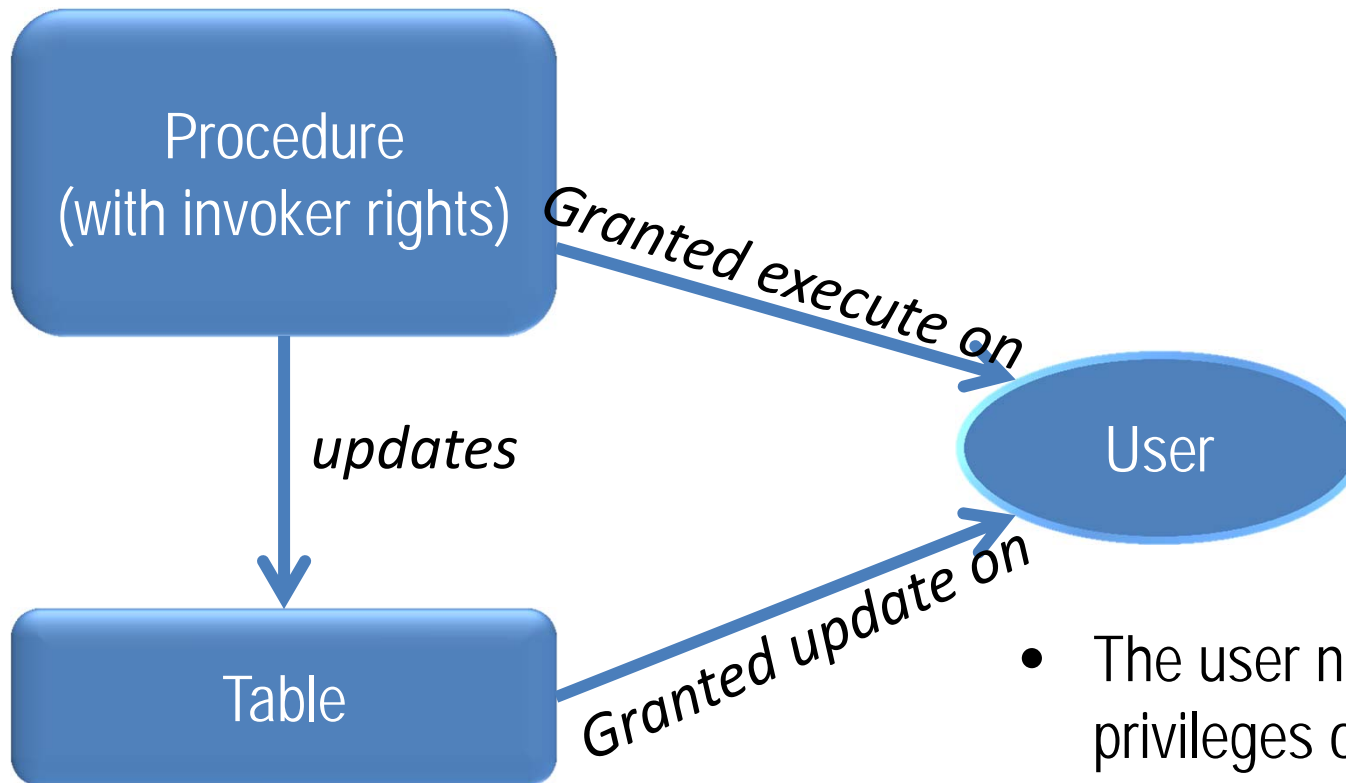
- Pragmas
  - pragma autonomous\_transaction;
  - pragma exception\_init (deadlock\_detected, -60);
- Pragma UDF (user defined function)

```
create or replace function get_bonus_amount  
(  
  p_accno in accounts.accno%type  
)  
return number  
is  
pragma UDF;  
  l_bonus      number;
```

...

[get\\_bonus\\_amount\\_pragma\\_udf\\_v1.sql](#)

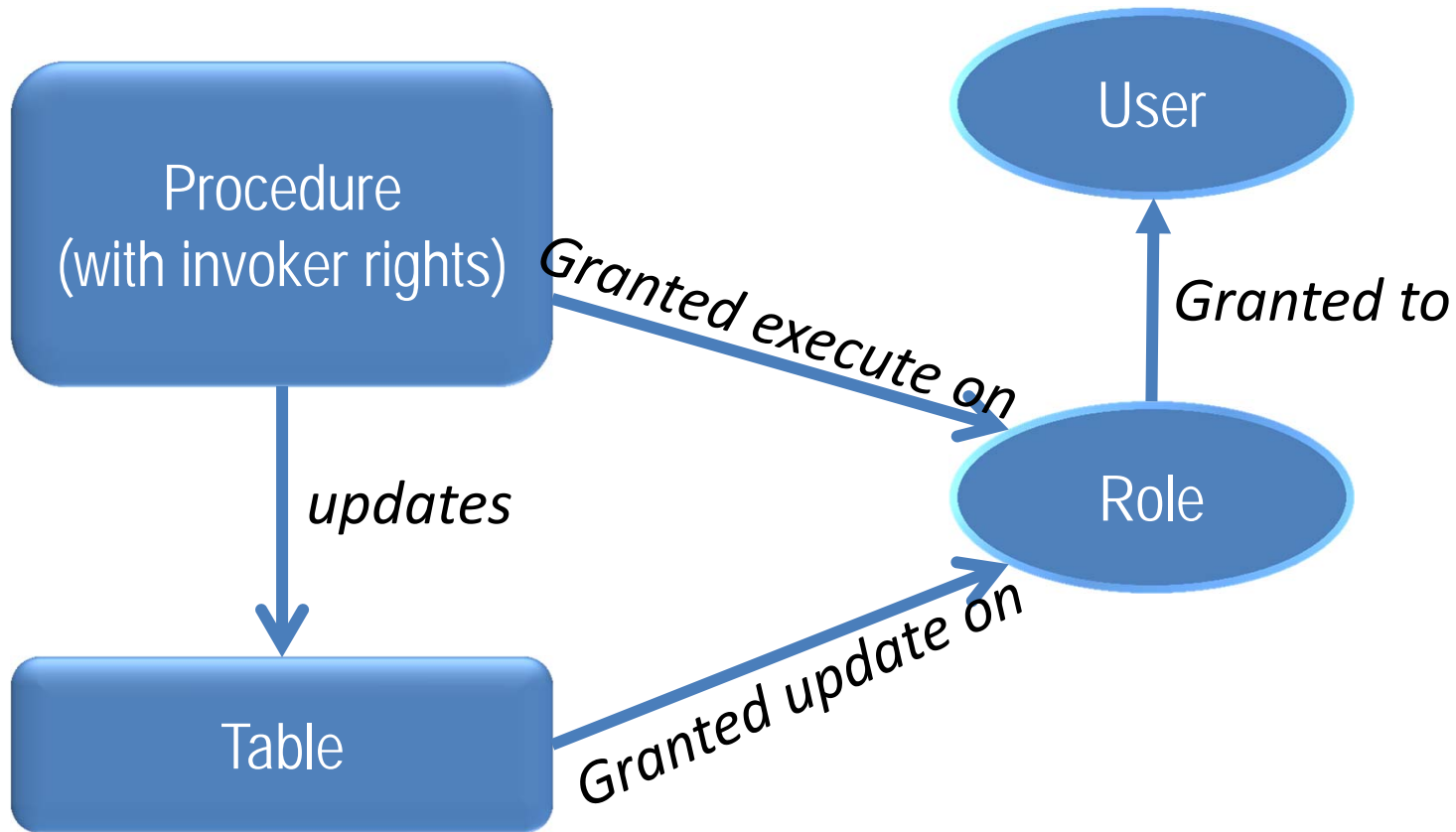
# Program Security



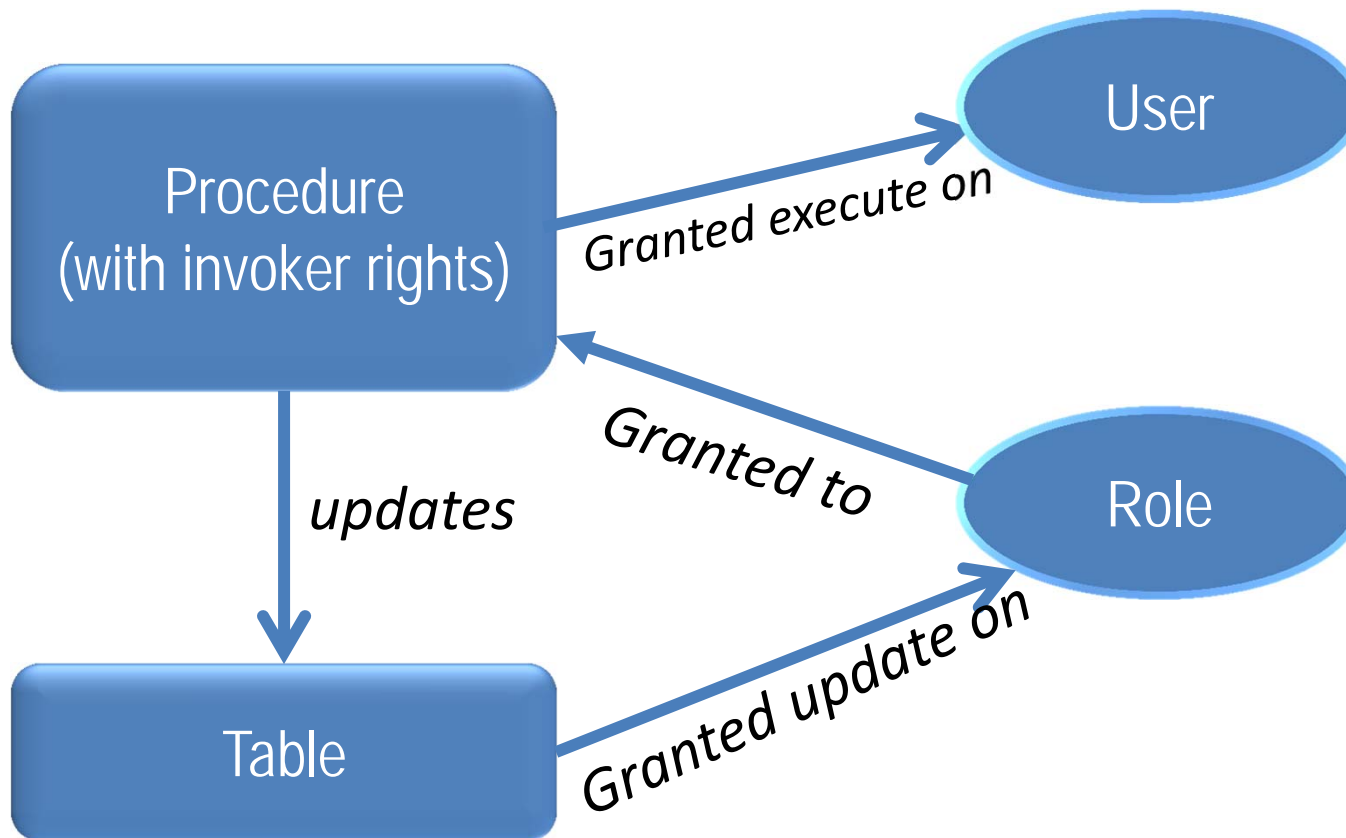
- The user needs update privileges on the table
- Can update anything; not just what the procedure is capable of

roles\_to\_proc\_trad.sql

# Will a role help?



# 12c Way



roles\_to\_proc\_12c.sql



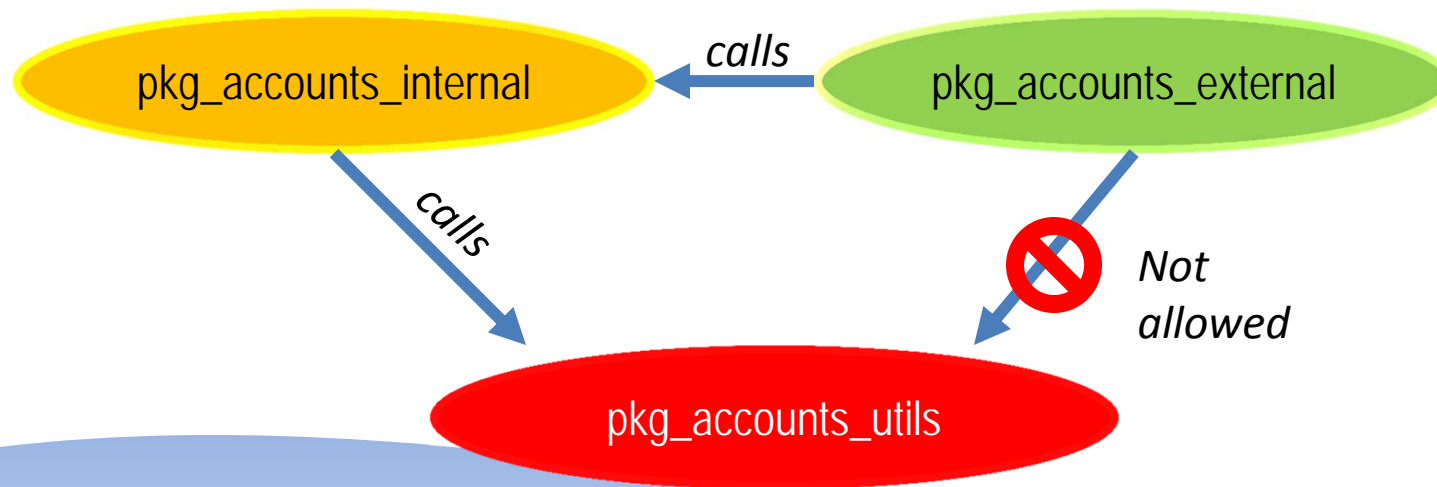
# Program Security

Traditional Approach	New Approach
Grant execute on procedure to SCOTT	Grant execute on procedure to SCOTT
Grant update on table to SCOTT	
	Grant update on table to role
	Grant role to procedure

```
grant <role> to procedure <proc>
```

# Code Whitelisting

- You have a utility package
  - pkg\_accounts\_utils
  - contains sensitive procedures
- You have an internal package that calls these utils
  - pkg\_accounts\_internal
- External package called by others
  - pkg\_accounts\_external



# Accessible By

- The package specification should have  
create or replace package pkg\_accounts\_utils  
**accessible by (pkg\_accounts\_internal)**

is

```
procedure update_int (p_accno in accounts.accno%type,  
p_int_amt in accounts.interest%type);  
end;
```

- Others will get

PLS-00904: insufficient privilege to access object  
PKG\_ACCOUNTS\_UTILS

```
pkg_accounts_utils_pks_v1.sql  
pkg_accounts_utils_pkb_v1.sql  
pkg_accounts_internal_pks_v1.sql  
pkg_accounts_internal_pkb_v1.sql  
pkg_accounts_external_pks_v1.sql  
pkg_accounts_external_pkb_v1.sql
```

# Collections in PL/SQL

- Using collections

```
function get_avg_int_accnolist  
(  
    p_int_table in <Collection>  
)  
return number is
```

- Parameter p\_int\_table had to be a database schema object

- Varray
- Nested Table

## PL/SQL Constructs

Records

PL/SQL Table Indexed by  
pls\_integer

# Package Collections

- Create a package spec (body not required)

```
create or replace package pkg_acc is
  type ty_rec_int is record (
    accno      number,
    interest   number
  );
  type ty_tab_int is table of ty_rec_int index by pls_integer;
  rec_int      ty_tab_int;
end;
/
```

Pkg\_acc\_v1.sql

# Use Collections in Binds

- Function to compute average interest for a group of accounts

```
create or replace function get_avg_int_accnolist
  (p_int_table in pkg_acc.ty_tab_int)
return number is
  l_tot_int      number := 0;
  l_avg_int      number := 0;
  cnt            pls_integer := 0;
begin
  cnt := p_int_table.count;
  for i in 1..cnt loop
    l_tot_int := p_int_table(i).interest + l_tot_int;
  end loop;
  l_avg_int := l_tot_int / cnt;
  return l_avg_int;
End;
```

[get\\_avg\\_int\\_accnolist\\_v1.sql](#)

# More Binds

- Trying to find the average interest for 65 years and older customers
- Populate

```
select accno, interest
bulk collect into pkg_acc.rec_int
from accounts
where sysdate - birthday > 65*365;
```
- Example 1

```
select get_avg_int_accnolist(pkg_acc.rec_int)
into l_avg_int from dual;
```
- Example 2

```
execute immediate 'select get_avg_int_accnolist (:var1) from dual'
into l_avg_int
using pkg_acc.rec_int;
```
- Example 3

```
select avg(interest), count(1)
into l_avg_int, l_tot_accs
from table(pkg_acc.rec_int);
```

[show\\_avg\\_int\\_for\\_65yrs\\_v1.sql](#)  
[show\\_avg\\_int\\_for\\_65yrs\\_v2.sql](#)

# Boolean Binding

```
create or replace procedure show_boolean
(
    p_truth in boolean
) is
    l_val  varchar2(5);
begin
    l_val :=
        case p_truth
        when TRUE      then 'TRUE'
        when FALSE     then 'FALSE'
        else           'NULL'
        end;
    dbms_output.put_line ('The input was ' || l_val);
end;
/
```

Show\_boolean\_v1.sql



# Boolean Binding

```
create or replace procedure show_truth
is
  c_null constant boolean := null;
begin
  execute immediate 'begin show_boolean(:param); end;' using true;
  execute immediate 'begin show_boolean(:param); end;' using false;
  execute immediate 'begin show_boolean(:param); end;' using
c_null;
end;
/
```

[show\\_truth\\_v1.sql](#)

# Predefined Directives

- In previous versions only a few directive were made available to you to display:
  - \$\$PLSQL\_UNIT
  - \$\$PLSQL\_LINE
- Now you can select two additional directives
  - \$\$PLSQL\_UNIT\_TYPE
  - \$\$PLSQL\_UNIT\_OWNER

demo\_proc1\_v1.sql  
demo\_proc2\_v1.sql

# Nested Codes

```
create or replace procedure depth_proc1 as
  procedure depth_proc2 as
    procedure depth_proc3 as
      procedure depth_proc4 as
        procedure depth_proc5 as
          begin
            some_code_comes_here;
          end;
        begin
          depth_proc5;
        end;
      begin
        depth_proc4;
      end;
    begin
      depth_proc3;
    end;
  begin
    depth_proc2;
  end;
```



Need to place  
call stack data  
here

# Call Stack

- Before 12c  
create or replace procedure  
display\_call\_stack  
is  
begin  
    dbms\_output.put\_line(dbms\_utility.format  
\_call\_stack);  
end;  
/

depth\_proc\_demo\_v1.sql  
display\_call\_stack\_v1.sql

# 12c Way

```
create or replace procedure display_call_stack
is
    l_dynamic_depth      pls_integer := utl_call_stack.dynamic_depth;
begin
    -- dbms_output.put_line(dbms_utility.format_call_stack);
    for i in reverse 1..l_dynamic_depth loop
        dbms_output.put_line (
            lpad(to_char(i,'99'),3)
            ||' '||
            utl_call_stack.owner(i)
            ||'.'||
            utl_call_stack.concatenate_subprogram (utl_call_stack.subprogram (i))
            ||' ('||
            utl_call_stack.unit_line(i)
            ||') '
        );
    end loop;
end;
```

[display\\_call\\_stack\\_v2.sql](#)



# *Thank You!*

**My Blog: [arup.blogspot.com](http://arup.blogspot.com)**

**My Tweeter: [arupnanda](#)**