

Nama : Afina Putri Dayanti
NIM : 825200049
Jurusan : Sistem Informasi
Mata Kuliah : Database Design and Management (Praktikum)

Vocabulary

Identify the vocabulary word for each definition below:

Visibility	Describes if a component can be seen, that is, referenced and used by other components or objects.
Private components	are declared only in the package body and can be referenced only by other (public or private) constructs within the same package body.
Public components	are declared in the package specification and can be invoked from any calling environment, provided the user has been granted EXECUTE privilege on the package.

Try It / Solve It

1. Create a package called hellofrom that contains three public procedures named proc_1, proc_2 and proc_3. Each of these procedures should use DBMS_OUTPUT.PUT_LINE() to display the message "Hello from Proc x" where "x" is 1 or 2 or 3, as appropriate.
Also, proc_1 should call proc_2 and proc_2 should call proc_3, so you need to include a reference to proc_2 from proc_1, and a reference to proc_3 from proc_2.
You will be making changes to this package specification and body, so make sure you save both the specification and body in your Application Express session.

Answer :

```
--1 spec
create or replace package hellofrom
is
  procedure proc_1;
  procedure proc_2;
  procedure proc_3;
end hellofrom;
/

-- body
create or replace package body hellofrom
is
  procedure proc_1 is
  begin
    dbms_output.put_line('Hello from Proc 1');
    proc_2;
  end proc_1;

  procedure proc_2 is
  begin
    dbms_output.put_line('Hello from Proc 2');
    proc_3;
  end proc_2;
```

```

end proc_2;

procedure proc_3 is
begin
    dbms_output.put_line('success');
end proc_3;
end hellofrom;

```

2. DESCRIBE the package to check that all three procedures are visible.

**) can't describe, because it doesn't have parameters so I use user procedure*

The screenshot shows the SQL Developer interface. The main window displays a query in the Worksheet:

```

begin
    hellofrom.proc_1;
end;
/
select * from user_procedures where object_name = 'HELLOFROM';

select * from user_source where name = 'HELLOFROM';
/

desc hellofrom;

```

The Query Result pane shows the following data:

OBJECT_NAME	PROCEDURE_NAME	OBJECT_ID	SUBPROGRAM_ID	OVERLOAD	OBJECT_TYPE	AGGREGATE	PIPELINED	IMPLTYPEOWNER	IMPLTYPEENAME	PARALLEL
HELLOFROM	PROC_1	26468	1 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK
HELLOFROM	PROC_2	26468	2 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK
HELLOFROM	PROC_3	26468	3 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK
HELLOFROM	(null)	26468	0 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK

Then, create and execute an anonymous block which displays all three messages with only one procedure call.

```

begin
    hellofrom.proc_1;
end;

```

The screenshot shows the SQL Developer interface. The main window displays an anonymous block in the Worksheet:

```

procedure proc_2 is
begin
    dbms_output.put_line('Hello from Proc 2');
    proc_3;
end proc_2;

procedure proc_3 is
begin
    dbms_output.put_line('Hello from Proc 3');
end proc_3;
end hellofrom;
/

DESC hellofrom;

begin
    hellofrom.proc_1;
end;

```

The Script Output pane shows the following output:

```

Task completed in 0.001 seconds

Hello from Proc 1
Hello from Proc 2
Hello from Proc 3

```

Modify the hellofrom package specification (not the body) so that only proc_1 is public and proc_2 and proc_3 are private.

```
create or replace package hellofrom
is
    procedure proc_1;
end hellofrom;
```

Recreate the package and then DESCRIBE it. What do you see?

**) can't describe, because it doesn't have parameters so I use user procedure*

The screenshot shows the SQL Developer interface with the following SQL code in the worksheet:

```
begin
    hellofrom.proc_1;
end;
/
select * from user_procedures where object_name = 'HELLOFROM';

select * from user_source where name = 'HELLOFROM';
/

desc hellofrom;
```

The Query Result pane displays the following table:

OBJECT_NAME	PROCEDURE_NAME	OBJECT_ID	SUBPROGRAM_ID	OVERLOAD	OBJECT_TYPE	AGGREGATE	PIPELINED	IMPLTYPEOWNER	IMPLTYPEOWNER	PARALLEL
HELLOFROM	PROC_1	26468	1 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK
HELLOFROM	(null)	26468	0 (null)	PACKAGE	NO	NO	(null)	(null)	NO	NK

3. What will happen if you try to run hellofrom.proc_1 now, before you make any changes to the body? Try it.

Answer : error could not find program unit being called: "SYSTEM.HELLOFROM"

The screenshot shows the SQL Developer interface with the following SQL code in the worksheet:

```
end proc_3;
end hellofrom;
/

begin
    hellofrom.proc_1;
end;
/
```

The Script Output pane displays the following error report:

```
end;
Error report -
ORA-04063: package body "SYSTEM.HELLOFROM" has errors
ORA-06508: PL/SQL: could not find program unit being called: "SYSTEM.HELLOFROM"
ORA-06512: at line 2
04063. 00000 - "ks has errors"
Cause: Attempt to execute a stored procedure or use a view that has errors. For stored procedures, the problem could be syntax errors or references to other, non-existent procedures. For views, the problem could be a reference in the view's defining query to a non-existent table. Can also be a table which has references to non-existent or inaccessible types.
Action: Fix the errors and/or create referenced objects as necessary.
```

4. What changes can you make to the package body in order for proc_1 to run successfully? Make the changes and recreate the body.

Answer :

```
create or replace package body hellofrom
is
  procedure proc_2;
  procedure proc_3;
  procedure proc_1 is
  begin
    dbms_output.put_line('Hello from Proc 1');
    proc_2;
  end proc_1;

  procedure proc_2 is
  begin
    dbms_output.put_line('Hello from Proc 2');
    proc_3;
  end proc_2;

  procedure proc_3 is
  begin
    dbms_output.put_line('Hello from Proc 3');
  end proc_3;
end hellofrom;
```

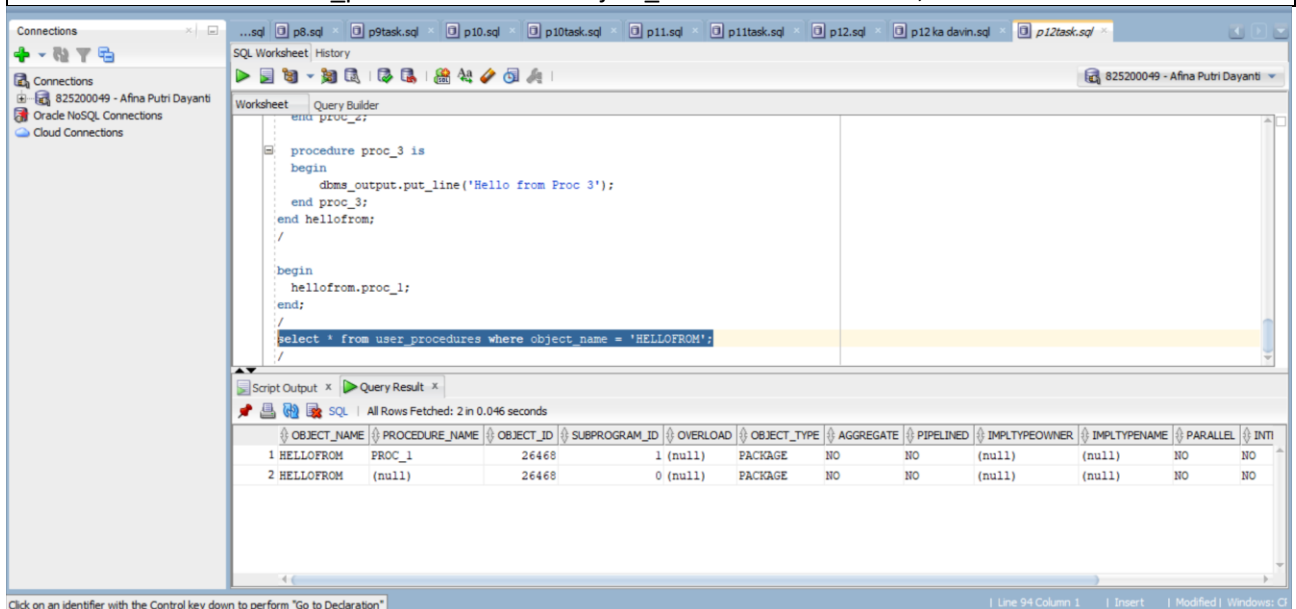
5. Try to call each of the three procedures from anonymous blocks. What happens? Explain why some of these calls fail.

Answer : The proc_2 and proc_3 are failed, because the procedures are private and cannot be called from external calls.

6. Look in USER_PROCEDURES for the hellofrom package. What do you see?

Answer : one defined procedure for hellofrom

```
select * from user_procedures where object_name = 'HELLOFROM';
```

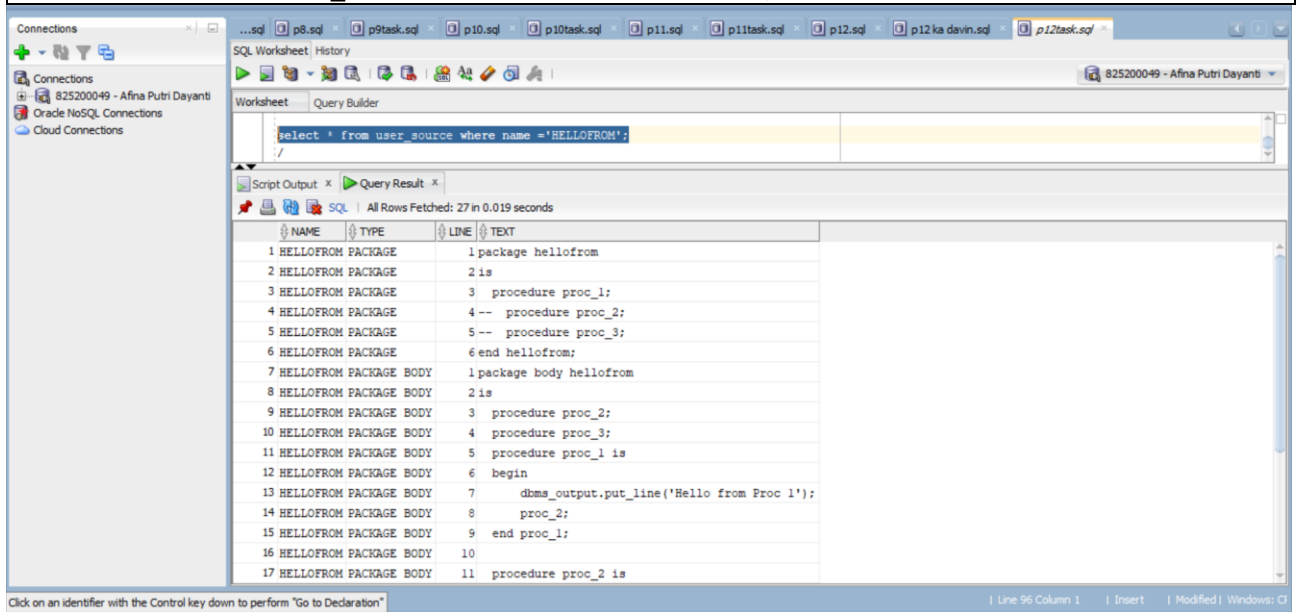


The screenshot shows the SQL Developer interface with the query results displayed in the Query Result tab. The query executed was `select * from user_procedures where object_name = 'HELLOFROM';`. The results table has the following data:

	OBJECT_NAME	PROCEDURE_NAME	OBJECT_ID	SUBPROGRAM_ID	OVERLOAD	OBJECT_TYPE	AGGREGATE	PIPELINED	IMPLTYPEOWNER	IMPLTYPEOWNER	PARALLEL	INTI
1	HELLOFROM	PROC_1	26468	1 (null)		PACKAGE	NO	NO	(null)	(null)	NO	NO
2	HELLOFROM	(null)	26468	0 (null)		PACKAGE	NO	NO	(null)	(null)	NO	NO

7. Also have a look in USER_SOURCE. Unlike USER_PROCEDURES, USER_SOURCE shows the entire body, including the private procedures.

```
select * from user_source where name='HELLOFROM';
```



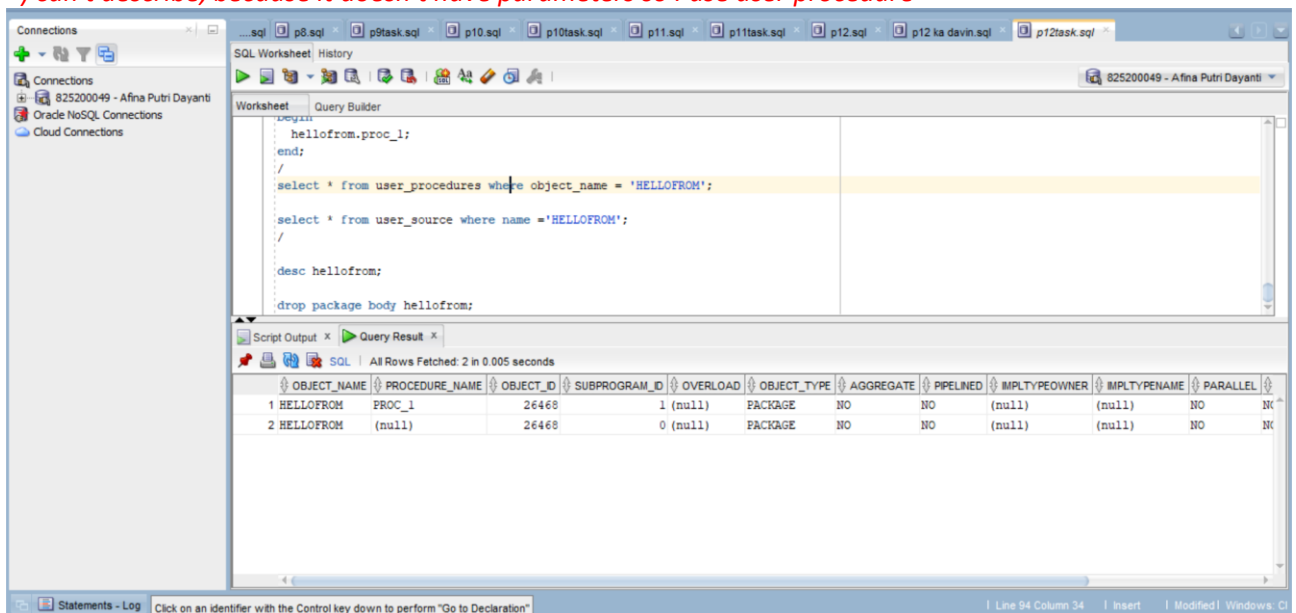
NAME	TYPE	LINE	TEXT
HELLOFROM	PACKAGE	1	package hellofrom
HELLOFROM	PACKAGE	2	is
HELLOFROM	PACKAGE	3	procedure proc_1;
HELLOFROM	PACKAGE	4	-- procedure proc_2;
HELLOFROM	PACKAGE	5	-- procedure proc_3;
HELLOFROM	PACKAGE	6	end hellofrom;
HELLOFROM	PACKAGE BODY	1	package body hellofrom
HELLOFROM	PACKAGE BODY	2	is
HELLOFROM	PACKAGE BODY	3	procedure proc_2;
HELLOFROM	PACKAGE BODY	4	procedure proc_3;
HELLOFROM	PACKAGE BODY	5	procedure proc_1 is
HELLOFROM	PACKAGE BODY	6	begin
HELLOFROM	PACKAGE BODY	7	dbms_output.put_line('Hello from Proc 1');
HELLOFROM	PACKAGE BODY	8	proc_2;
HELLOFROM	PACKAGE BODY	9	end proc_1;
HELLOFROM	PACKAGE BODY	10	
HELLOFROM	PACKAGE BODY	11	procedure proc_2 is

8. Make sure you have saved the hellofrom package body code from question 1. Then try to drop just the body. What happens? What do you see when you DESCRIBE helloform?

Answer :

```
drop package body hellofrom;
```

**) can't describe, because it doesn't have parameters so I use user procedure*



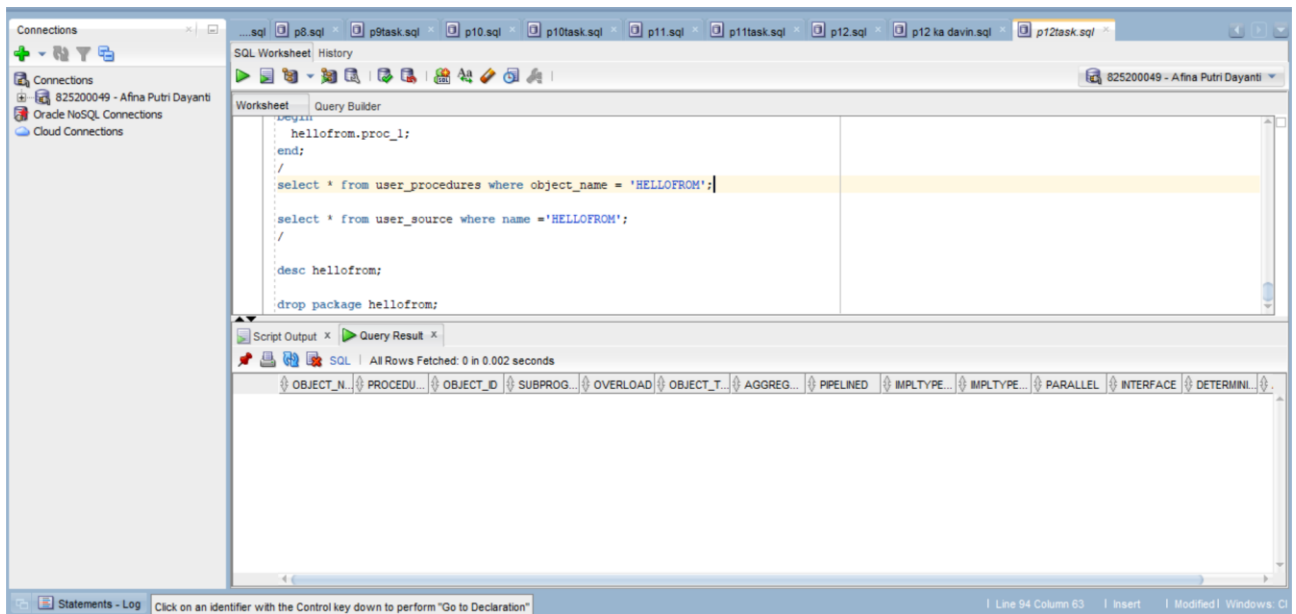
OBJECT_NAME	PROCEDURE_NAME	OBJECT_ID	SUBPROGRAM_ID	OVERLOAD	OBJECT_TYPE	AGGREGATE	PIPELINED	IMPLTYPEOWNER	IMPLTYPE	NAME	PARALLEL
HELLOFROM	PROC_1	26468	1	(null)	PACKAGE	NO	NO	(null)	(null)	NO	NK
HELLOFROM	(null)	26468	0	(null)	PACKAGE	NO	NO	(null)	(null)	NO	NK

9. Re-create the body from your saved code. Now try to drop just the specification. What happens?

Answer :

```
drop package hellofrom;
```

**) can't describe, because it doesn't have parameters so I use user procedure*



10. Name at least two advantages of enclosing these three procedures in a package.
Answer : It makes interfaces between packages that would use these procedures simple and you have the option of making whichever procedure you want either private or public.