

MODUL 10

PROSEDUR, FUNGSI, PAKET

1. DASAR TEORI

Salah satu alasan pemakaian basis data adalah kemampuan penerapan aturan bisnis pada basis data. Dengan demikian aturan bisnis akan berlaku bagi semua client yang mengakses basis data. Pada unit ini akan dipelajari pembuatan program (dalam hal ini adalah prosedur dan fungsi) yang tersimpan di basis data. Program ini disebut *stored-procedure* atau *stored-function*.

Program tersimpan (baik berupa *stored-procedure* atau *stored-function*) tersimpan di server basis data. Program tersebut dapat berupa:

- Prosedur (*stand-alone procedure*).
- Fungsi (*stand-alone function*).
- Paket (*package*), yang berisi prosedur dan fungsi.

Paket adalah suatu unit program yang mirip dengan pustaka prosedur dan fungsi. Selain itu, paket dapat juga menyimpan data referensi yang tidak berubah (konstanta).

Prosedur

Prosedur adalah sekumpulan SQL statement dan perintah-perintah PL/SQL yang dikompilasi dan disimpan dalam skema basis data Oracle. Prosedur bekerja sebagai sebuah unit atau obyek yang dapat mengerjakan sekumpulan tugas. Seperti halnya unit atau obyek lain yang berada dalam basis data Oracle (misal: *table*, *constraint*, *view*, *index* dan sebagainya), spesifikasi prosedur juga disimpan dalam kamus data.

Penulisan prosedur adalah sebagai berikut:

```
CREATE [OR REPLACE] PROCEDURE nama_prosedur IS
```

```
-- bagian deklarasi
```

```
BEGIN
```

```
--bagian program
```

```
EXCEPTION
```

```
--bagian exception handler (penanganan kesalahan)
```

```
END nama_prosedur;  
  
/
```

Prosedur terdiri dari tiga (3) bagian, yaitu bagian deklarasi, bagian program dan bagian exception.

Pernyataan CREATE digunakan hanya untuk membuat prosedur baru, sedangkan pernyataan CREATE OR REPLACE dapat digunakan untuk membuat ataupun mengganti prosedur yang sudah ada sebelumnya. Oleh karena itu, seterusnya digunakan pernyataan CREATE OR REPLACE.

Perintah untuk menghapus prosedur adalah:

```
DROP PROCEDURE nama_prosedur ;
```

Contoh :

Prosedur untuk menghitung jumlah item stock yang tersedia:

```
create or replace procedure jumlah_item is  
  
    banyak number;  
  
begin  
  
    select count(kd_stock) into banyak from d_stock;  
  
    dbms_output.put_line('Jumlah item stock yang tersedia  
  
    sebanyak : ' || banyak);  
  
end;  
  
/
```

Penjelasan contoh prosedur:

- **create** digunakan untuk membuat prosedur dan **replace** digunakan untuk mengganti isi prosedur yang mempunyai nama sama dengan prosedur ini.

- **Banyak number**, artinya kita mendeklarasikan sebuah *variable* bernama banyak dan bertipe number.
- **Select count(kd_stock) into banyak from d_stock;** Statement ini digunakan untuk menampilkan jumlah cacah item data yang tersedia dari table d_stock yang kemudian disimpan dalam *variabel* banyak.
- **dbms_output.put_line('Jumlah item stock yang tersedia sebanyak : ' || banyak);** Digunakan untuk menampilkan teks "Jumlah item stock yang tersedia sebanyak :" yang akan digabungkan dengan nilai variable banyak.

Cara memanggil prosedur:

```
execute nama_prosedur;
```

Untuk memanggil contoh prosedur diatas adalah:

```
execute jumlah_item;
```

→Jumlah item stock yang tersedia sebanyak : 5

Fungsi

Seperti halnya prosedur, pengertiannya tidak jauh berbeda dengan prosedur. Fungsi juga dikompilasi dan disimpan dalam skema basis data Oracle dan spesifikasinya juga disimpan dalam kamus data. Perbedaannya adalah fungsi selalu memiliki nilai balikan (*return values*).

Penulisan fungsi adalah sebagai berikut:

```
CREATE [OR REPLACE] FUNCTION nama_fungsi
```

```
RETURN tipe_data_keluaran IS
```

```
-- bagian deklarasi
```

```
BEGIN
```

```
-- bagian program
```

```
RETURN nilai_keluaran ;
```

```
EXCEPTION
```

-- bagian exception handler (penanganan kesalahan)

```
END nama_fungsi ;
```

```
/
```

Penggunaan kata kunci CREATE OR REPLACE berfungsi sama dengan prosedur. Sedangkan **RETURN** merupakan kata kunci yang digunakan untuk mengembalikan nilai yang dihasilkan dari fungsi tersebut.

Perintah untuk menghapus fungsi adalah:

```
DROP FUNCTION nama_fungsi ;
```

Contoh:

Fungsi untuk mengetahui harga barang yang mempunyai kode '10100001'

```
create or replace function cari_harga  
  
return d_stock.h_jual%type is  
  
harga d_stock.h_jual%type;  
  
begin  
  
select h_jual into harga from d_stock  
  
where kd_stock = '10100001';  
  
return harga;  
  
end;  
  
/
```

Keterangan contoh fungsi:

- **return d_stock.h_jual%type**, sintak ini berarti bahwa nilai balikan dari fungsi tersebut bertipe sama dengan field tujuan dalam tabel d_stock.
- **Harga d_stock.h_jual%type**; sintak ini mendeklarasikan variabel harga dengan tipe yang sama dengan field tujuan dalam tabel d_stock.
- **return harga**; digunakan untuk mengembalikan nilai yang dihasilkan fungsi, yaitu harga dari barang yang berkode '10100001' yang telah tersimpan dalam variabel harga.

Cara memanggil fungsi diatas:

```
select distinct cari_harga from dual;
```

Parameter Prosedur dan Fungsi

Prosedur dan fungsi dapat mempunyai parameter. Parameter ini digunakan untuk melewati nilai. Parameter dideklarasikan setelah nama prosedur atau nama fungsi.

Parameter dapat berupa:

IN → parameter untuk melewati nilai masuk ke prosedur atau fungsi.

OUT → parameter untuk melewati nilai keluar ke prosedur atau fungsi.

IN OUT → parameter yang dapat melewati nilai masuk ataupun keluar.

Jika IN, OUT, atau IN OUT tidak disertakan, maka defaultnya adalah IN. Fungsi hanya mempunyai parameter masukan dan tidak mempunyai parameter keluaran. Karena fungsi telah mempunyai pernyataan sendiri untuk mengeluarkan nilai, yaitu RETURN.

Sehingga pernyataan nama_prosedur atau nama_fungsi diatas menjadi:

```
nama_prosedur [(parameter IN | OUT | IN OUT tipe_data,...)]
```

```
nama_fungsi [(parameter IN | OUT | IN OUT tipe_data,...)]
```

Eksekusi Prosedur dan Fungsi

Sebelum mengeksekusi prosedur atau fungsi, berikut dijelaskan terlebih dahulu mengenai variabel di SQL*Plus. Variabel digunakan sebagai parameter ataupun sebagai penerima keluaran dari fungsi. Variabel dapat diinisialisasi langsung.

Variabel-variabel SQL*Plus tidak dapat diakses dari program PL/SQL.

Perintah untuk mendeklarasikan variabel adalah:

```
VARIABLE nama_variabel tipe_data ;
```

Pemberian nilai ke variabel dilakukan dengan perintah :

```
EXECUTE :nama_variabel := nilai_yang_diberikan ;
```

Perhatikan!!!

Pada penugasan variabel harus diberikan tanda titik-dua sebelum nama_variabel.

Perintah untuk menampilkan isi variabel adalah:

```
PRINT nama_variabel ;
```

Dengan menggunakan variabel ini, prosedur dan fungsi dapat dievaluasi dengan perintah EXECUTE.

```
EXECUTE nama_prosedur [(parameter . . . . .)] ;
```

```
EXECUTE :nama_variabel := nama_fungsi [(parameter.....)] ;
```

Prosedur dan function dapat dipanggil melalui program PL/SQL dengan cukup menyebutkan nama prosedur atau fungsi beserta parameternya, jika ada.

Paket

Paket (*package*) terbentuk dari bagian spesifikasi dan bagian badan (*body*). Bagian spesifikasi digunakan untuk mendeklarasikan prosedur atau fungsi yang dikandungnya. Sedangkan bagian badan mendeklarasikan isi implemetasi dari prosedur atau fungsi yang dinyatakan dalam bagian spesifikasi paket.

Bagian spesifikasi paket dapat diakses baik dari dalam paket itu sendiri atau dari luar. Sedangkan bagian badan hanya berlaku untuk paket itu sendiri.

Penulisan bagian spesifikasi:

```
CREATE OR REPLACE PACKAGE nama_paket IS

-- deklarasi konstanta, tipe data, variable, dan sebagainya.

PROCEDURE nama_prosedur[(parameter . . . .)];

FUNCTION nama_fungsi[(parameter . . . .)]

RETURN tipe_data_keluaran ;

END nam_paket;

/
```

Penulisan bagian badan:

```
CREATE OR REPLACE PACKAGE BODY nama_paket IS

-- deklarasikan konstanta, tipe data, variable dan sebagainya.

PROCEDURE nama_prosedur [(parameter . . .)] IS

BEGIN

    -- implementasi prosedur.

END nama_prosedur;

FUNCTION nama_fungsi [(parameter . . .)]
```

```

RETURN tipe_data_keluaran IS

BEGIN

-- implementasi fungsi.

END nama_fungsi;

END nama_paket;

/

```

Untuk mengakses prosedur didalam paket, digunakan cara sebagai berikut:

```
execute nama_paket.nama_prosedur[(parameter . . .)];
```

Untuk mengakses fungsi didalam paket, digunakan cara berikut:

```
suatu_variabel:=nama_paket.nama_fungsi[(parameter..)];
```

atau dengan cara DBMS_OUPUT;

```

begin

dbms_output.put_line('Jenis          Mobil          :          ' ||
paket_jenis.jenis_mobil2('AB001'));

end;

/

```

Perintah untuk menghapus paket adalah:

```
drop package nama_paket;
```

Kamus Data

Kamus data yang tersedia untuk prosedur, fungsi dan paket adalah USER_SOURCE. User Source digunakan untuk mengetahui atau melihat kode program dari prosedur, fungsi atau paket. Pada USER_SOURCE terdapat beberapa kolom, antara lain:

Nama	Keterangan
------	------------

Kolom	
NAME	Nama prosedur, fungsi atau paket
TYPE	Menyatakan tipe obyek: prosedur, fungsi atau paket
TEXT	Menyatakan kode program
LINE	Nomor baris program yang terkandung dalam TEXT

Cara untuk melihat kode program suatu prosedur atau fungsi adalah:

```
SELECT text FROM USER_SOURCE
WHERE NAME = nama_prosedur | nama_fungsi
ORDER BY LINE
```

2. LANGKAH PRAKTIKUM

```
set serveroutput on;
```

Prosedur dan fungsi tanpa parameter.

Berikut ini prosedur untuk membuat teks.

```
SQL> set serveroutput on;
SQL> create or replace procedure selamat_datang is
2  begin
3  dbms_output.enable;
4  dbms_output.put_line('selamat datang');
5  dbms_output.put_line('di praktikum basis data pertemuan 10');
6  end;
7  /
```

Procedure created.

Cara memanggil prosedur:

```
execute selamat_datang;
```

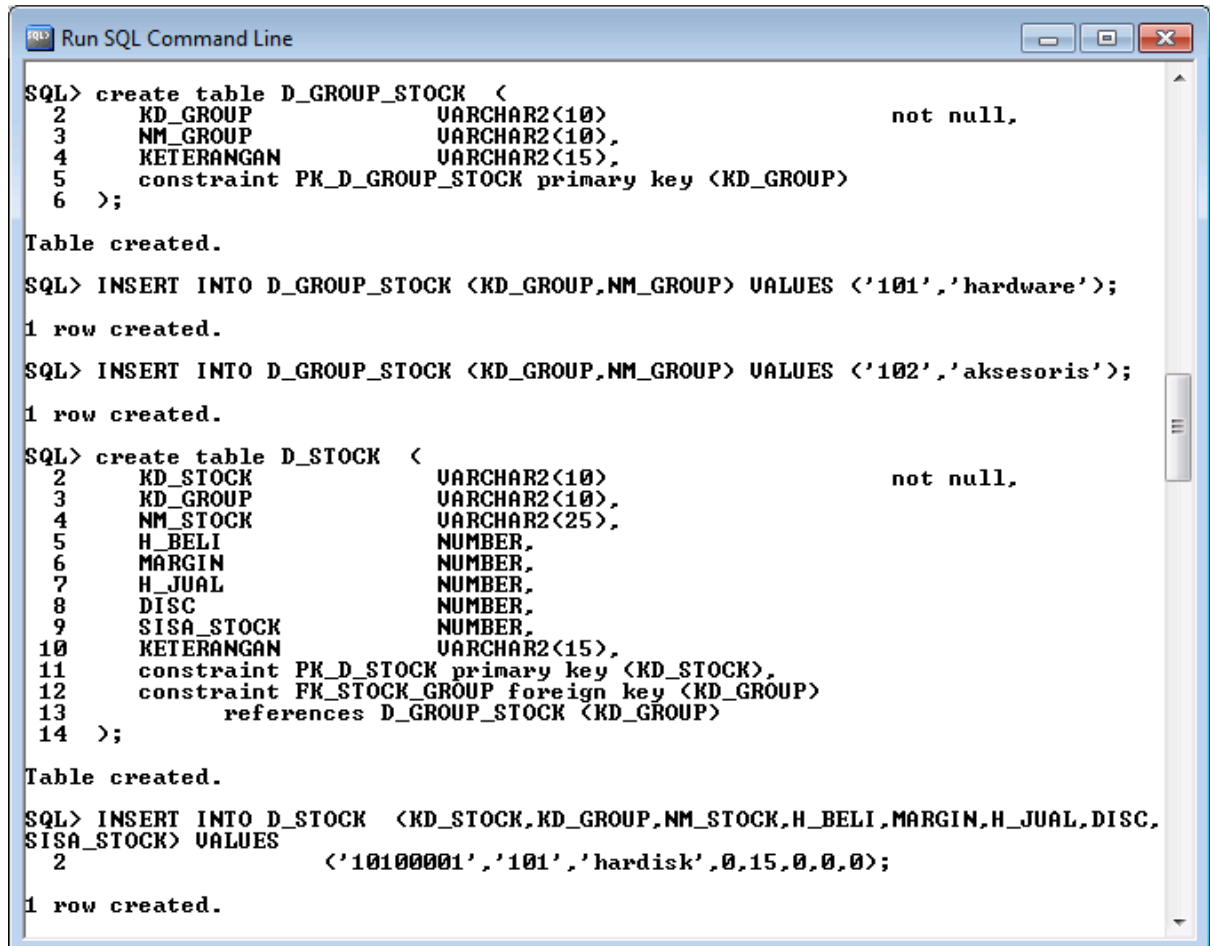
```

SQL> execute selamat_datang;
selamat datang
di praktikum basis data pertemuan 10

PL/SQL procedure successfully completed.

SQL>

```



```

Run SQL Command Line

SQL> create table D_GROUP_STOCK (
2   KD_GROUP          VARCHAR2(10)                not null,
3   NM_GROUP          VARCHAR2(10),
4   KETERANGAN        VARCHAR2(15),
5   constraint PK_D_GROUP_STOCK primary key (KD_GROUP)
6 );

Table created.

SQL> INSERT INTO D_GROUP_STOCK (KD_GROUP,NM_GROUP) VALUES ('101','hardware');

1 row created.

SQL> INSERT INTO D_GROUP_STOCK (KD_GROUP,NM_GROUP) VALUES ('102','aksesoris');

1 row created.

SQL> create table D_STOCK (
2   KD_STOCK          VARCHAR2(10)                not null,
3   KD_GROUP          VARCHAR2(10),
4   NM_STOCK          VARCHAR2(25),
5   H_BELI            NUMBER,
6   MARGIN            NUMBER,
7   H_JUAL            NUMBER,
8   DISC              NUMBER,
9   SISA_STOCK        NUMBER,
10  KETERANGAN        VARCHAR2(15),
11  constraint PK_D_STOCK primary key (KD_STOCK),
12  constraint FK_STOCK_GROUP foreign key (KD_GROUP)
13     references D_GROUP_STOCK (KD_GROUP)
14 );

Table created.

SQL> INSERT INTO D_STOCK (KD_STOCK,KD_GROUP,NM_STOCK,H_BELI,MARGIN,H_JUAL,DISC,
SISA_STOCK) VALUES
2   ('10100001','101','hardisk',0,15,0,0,0);

1 row created.

```

Run SQL Command Line

```
SQL> select*from d_stock;
```

KD_STOCK	KD_GROUP	NM_STOCK	H_BELI	MARGIN	H_JUAL

	DISC	SISA_STOCK	KETERANGAN		

10100001	101	hardisk	0	15	0
	0	0			
10100002	101	flasdisk	0	55	0
	0	0			
10100003	102	screen_guard	0	10	0
	0	0			
KD_STOCK	KD_GROUP	NM_STOCK	H_BELI	MARGIN	H_JUAL

	DISC	SISA_STOCK	KETERANGAN		

10100004	102	mouse	0	31	0
	0	0			

Berikut ini prosedur untuk melihat sisa barang yang berkode '10100001'.

Run SQL Command Line

```
SQL> create or replace procedure sissa_barang is
2  sisa d_stock.sisa_stock%type;
3  begin
4  select sisa_stock into sisa from d_stock
5  where kd_stock='10100001';
6  dbms_output.put_line('sisa barang dengan kode stok 10100001 adalah sebanyak
'|| sisa);
7  end;
8  /

Procedure created.
```

Cara memanggil prosedur:

Run SQL Command Line

```
SQL> execute sissa_barang;
sisa barang dengan kode stok 10100001 adalah sebanyak0
PL/SQL procedure successfully completed.
```

Berikut ini fungsi untuk melakukan perkalian.

```
create or replace function perkalian return number is

begin

return (111 * 3);
```

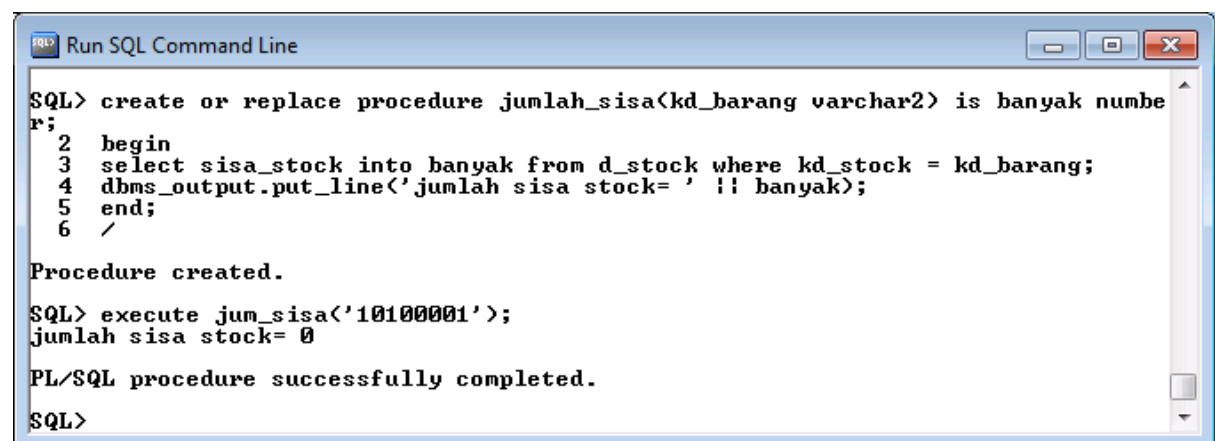
```
end;  
  
/
```

Cara memanggil fungsi:

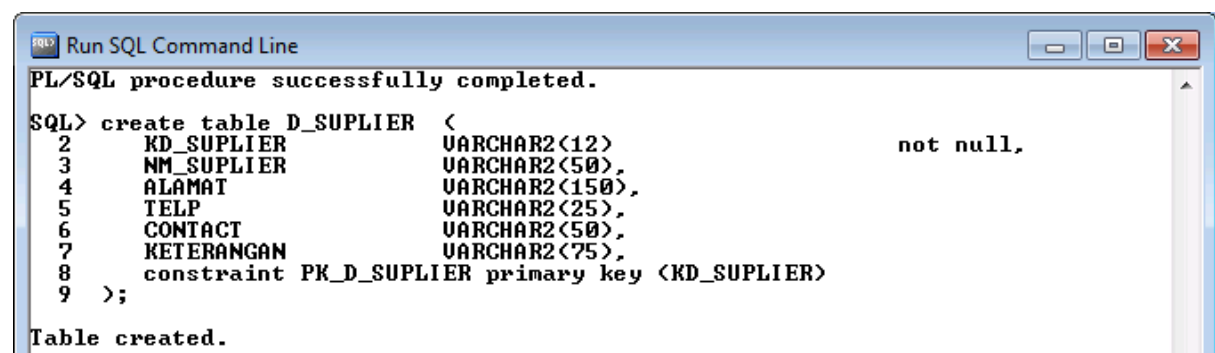
```
variable keluaran number;  
  
execute :keluaran := perkalian;  
  
print keluaran;
```

Prosedur dan fungsi dengan parameter.

Berikut ini fungsi untuk melihat jumlah sisa stock dari sebuah kode barang tertentu.



```
SQL> create or replace procedure jumlah_sisa(kd_barang varchar2) is banyak number;  
2 begin  
3 select sisa_stock into banyak from d_stock where kd_stock = kd_barang;  
4 dbms_output.put_line('jumlah sisa stock= ' || banyak);  
5 end;  
6 /  
  
Procedure created.  
  
SQL> execute jum_sisa('10100001');  
jumlah sisa stock= 0  
  
PL/SQL procedure successfully completed.  
  
SQL>
```



```
PL/SQL procedure successfully completed.  
  
SQL> create table D_SUPLIER (  
2 KD_SUPLIER VARCHAR2(12) not null,  
3 NM_SUPLIER VARCHAR2(50),  
4 ALAMAT VARCHAR2(150),  
5 TELP VARCHAR2(25),  
6 CONTACT VARCHAR2(50),  
7 KETERANGAN VARCHAR2(75),  
8 constraint PK_D_SUPLIER primary key (KD_SUPLIER)  
9 );  
  
Table created.
```

Berikut ini fungsi untuk mencari alamat supplier berdasar kode supplier.

```
SQL> create or replace function cari_alamat_spl (kode_suplier d_suplier
.kd_suplier%type)
2 return d_suplier.alamat%type is
3 alamat_spl d_suplier.alamat%type;
4 begin
5 select alamat into alamat_spl from d_suplier
6 where kd_suplier = kode_suplier;
7 return alamat_spl;
8 end;
9 /

Function created.
```

Cara memanggil fungsi tersebut:

```
SQL> variable alamat varchar2(150);
SQL> execute :alamat := cari_alamat_spl ('sp_002');

PL/SQL procedure successfully completed.

SQL> print alamat;

ALAMAT
-----
semarang

SQL> _
```

Keterangan: tipe data dari variable harus sama dengan tipe data nilai balikan, beserta juga ukurannya.

Paket.

Berikut ini paket yang berisi prosedur untuk melihat sisa stock barang berdasarkan kode stock.

Cara penulisan paket bagian spesifikasi:

```
SQL> create or replace package paket_stock is
2 procedure cek_sisa (kode_barang varchar2);
3 end paket_stock;
4 /

Package created.

SQL>
```

Cara penulisan paket bagian badan:

```
SQL> Run SQL Command Line

SQL> create or replace package body paket_stock is
  2  procedure cek_sisa (kode_barang varchar2) is
  3  sisa d_stock.sisa_stock%type;
  4  begin
  5      select sisa_stock into sisa from d_stock
  6  where kd_stock = kode_barang;
  7  dbms_output.put_line('Sisa stock = '|| sisa);
  8  exception
  9      when NO_DATA_FOUND then
 10      dbms_output.put_line('Data stok tidak ditemukan');
 11  end cek_sisa;
 12  end paket_stock;
 13  /

Package body created.

SQL>
```

Cara memanggil prosedur dalam paket:

```
execute paket_stock.cek_sisa('10100001');
```

Cara menghapus paket:

```
drop package paket_kapasitas;
```