PEMBUATAN FUNGSI

Tujuan Pembelajaran :

- Memahami penggunaan fungsi
- Dapat memanggil fungsi
- Dapat menghapus fungsi
- Memahami perbedaan antara prosedur dan fungsi

11.1. Pendahuluan

Suatu fungsi adalah block PL/SQL yang memiliki sebuah nama dan mengembalikan sebuah nilai. Fungsi dapat disimpan di database sebagai schema object agar dapat dipanggil dan dijalankan berulang kali. Fungsi dapat menjadi bagian dari suatu ekspresi.

11.2. Sintak untuk Membuat Fungsi

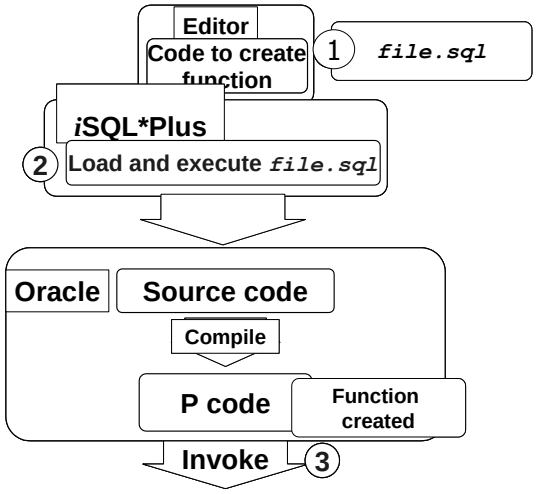
Sintak untuk membuat fungsi :

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1,
  parameter2 [mode2] datatype2,
  . . .)]
RETURN datatype
IS|AS
PL/SQL Block;
```

PL/SQL Block pada sintak diatas sedikitnya harus memiliki satu statement RETURN.

11.3. Membuat Fungsi dengan SQL*PLUS

Tahapan dalam pembuatan fungsi :



Berikut ini tahapan dalam pembuatan fungsi (stored function) :

- Tulis sintak : Masukkan kode untuk membuat fungsi dalam text editor dan simpan sebagai file script SQL.
- Compile kode tersebut : dengan menggunakan iSQL*PLUS, upload dan jalankan file script SQL. Source code kemudian dikompile ke dalam P code. Fungsi telah dibuat.
- Setelah itu, fungsi yang telah dibuat tersebut dapat dipanggil dari PL/SQL block.

Tambahkan klausa RETURN dengan tipe data sebagai header dari sebuah fungsi, dan sedikitnya satu statement RETURN pada bagian executable. Meskipun bisa lebih dari satu statement RETURN yang bisa ditulis dalam bagian executable, tapi hanya satu saja yang dijalankan.

11.4. Pembuatan Fungsi (Stored Function) dengan iSQL*PLUS

Untuk membuat fungsi dengan iSQL*PLUS :

- Masukkan kode fungsi yang didahului dengan Statement CREATE FUNCTION pada text editor kemudian simpan sebagai file script.
- Jalankan file script untuk menyimpan source code dan meng-compile fungsi
- Gunakan SHOW ERRORS untuk melihat kesalahan kompilasi
- Jika fungsi sukses di-compile, maka fungsi tersebut bisa dipanggil.

`get_salary.sql`

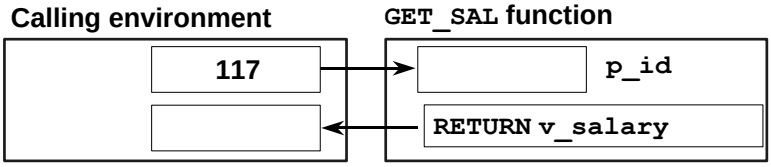
```
CREATE OR REPLACE FUNCTION get_sal
      (p_id  IN employees.employee_id%TYPE)
      RETURN NUMBER
IS
      v_salary employees.salary%TYPE :=0;
BEGIN
      SELECT salary
      INTO   v_salary
      FROM   employees
      WHERE  employee_id = p_id;
     RETURN v_salary;
END get_sal;
/
```

Contoh pembuatan fungsi :

11.5. Menjalankan Fungsi

Fungsi dapat dipanggil sebagai bagian dari suatu ekspresi, atau sebagai suatu nilai yang ditampung dalam sebuah variabel.

Ilustrasi dari menjalankan sebuah fungsi :



1. Panggil dan jalankan file `get_salary.sql` untuk membuat fungsi

```
VARIABLE g_salary NUMBER

EXECUTE :g_salary := get_sal(117)

PRINT g_salary
```

PL/SQL procedure successfully completed.

G_SALARY
2800

11.6. Penggunaan User-Defined Function pada Ekspresi SQL

Berikut ini penggunaan User-Defined Function pada ekspresi SQL :

- Menyederhanakan Statement SQL yang terlalu kompleks
- Dapat meningkatkan efisiensi pada saat perlu menggunakan klausa WHERE untuk menyaring data
- Dapat memanipulasi string karakter.

11.7. Memanggil Fungsi pada Ekspresi SQL

```
CREATE OR REPLACE FUNCTION tax(p_value IN NUMBER)
 RETURN NUMBER IS
BEGIN
   RETURN (p_value * 0.08);
END tax;
/
SELECT employee_id, last_name, salary, tax(salary)
FROM   employees
WHERE  department_id = 100;
```

Function created.

EMPLOYEE_ID	LAST_NAME	SALARY	TAX(SALARY)
108	Greenberg	12000	960
109	Faviet	9000	720
110	Chen	8200	656
111	Sciarra	7700	616
112	Urman	7800	624
113	Popp	6900	552

6 rows selected.

Slide berikut ini memperlihatkan fungsi yang dipanggil dari statement SELECT.:
Fungsi menerima parameter NUMBER dan mengembalikan pajak setelah nilai parameter dikalikan dengan 0.08. Pada iSQL*PLUS, fungsi TAX dipanggil dalam query yang menampilkan employee ID, nama, salary dan tax.

11.8. Bagian-bagian yang bisa Memanggil Fungsi

Berikut ini bagian-bagian yang bisa memanggil sebuah fungsi :

- Setelah statement SELECT
- Bagian dari kondisi yang ditulis pada klausa WHERE atau HAVING
- Pada klausa CONNECT BY, START WITH, ORDER BY dan GROUP BY
- Klausa VALUES yang ada pada perintah INSERT
- Klausa SET dari perintah UPDATE

11.9. Batasan dalam Pemanggilan Fungsi pada SQL Statement

Agar bisa dipanggil dari suatu ekspresi SQL, suatu fungsi harus :

- Disimpan sebagai stored function (schema object dalam database)
- Hanya menerima parameter IN
- Hanya menerima tipe data SQL yang valid, bukan tipe yang spesifik sebagai parameter fungsi
- Mengembalikan tipe data SQL yang valid, bukan tipe yang spesifik sebagai parameter fungsi
- Fungsi yang dipanggil dari ekspresi SQL tidak berisi statement DML.
- Fungsi yang dipanggil dari statement UPDATE/DELETE pada table T tidak berisi DML pada tabel yang sama yaitu tabel T.
- Fungsi yang dipanggil dari statement UPDATE/DELETE pada table T tidak melakukan query pada table yang sama yaitu tabel T.

Contoh berikut ini menunjukkan sebuah fungsi yang memiliki DML Statement untuk memasukkan record baru ke dalam tabel EMPLOYEES. Fungsi ini dipanggil dalam statement UPDATE yang memodifikasi nilai salary dari employee 170. Maka terdapat kesalahan yang menyebutkan bahwa *table is mutating.*

```
CREATE OR REPLACE FUNCTION dml_call_sql (p_sal NUMBER)
   RETURN NUMBER IS
BEGIN
   INSERT INTO employees(employee_id, last_name, email,
                         hire_date, job_id, salary)
        VALUES(1, 'employee 1', 'emp1@company.com',
                  SYSDATE, 'SA_MAN', 1000);
   RETURN (p_sal + 100);
END;
/
```

Function created.

```
UPDATE employees SET salary = dml_call_sql(2000)
 WHERE employee_id = 170;
```

```
UPDATE employees SET salary = dml_call_sql(2000)
                              *
ERROR at line 1:
ORA-04091: table PLSQL.EMPLOYEES is mutating, trigger/function may not see it
ORA-06512: at "PLSQL.DML_CALL_SQL", line 4
```

11.10. Menghapus Fungsi

Sintak untuk menghapus fungsi :

```
DROP FUNCTION function_name
```

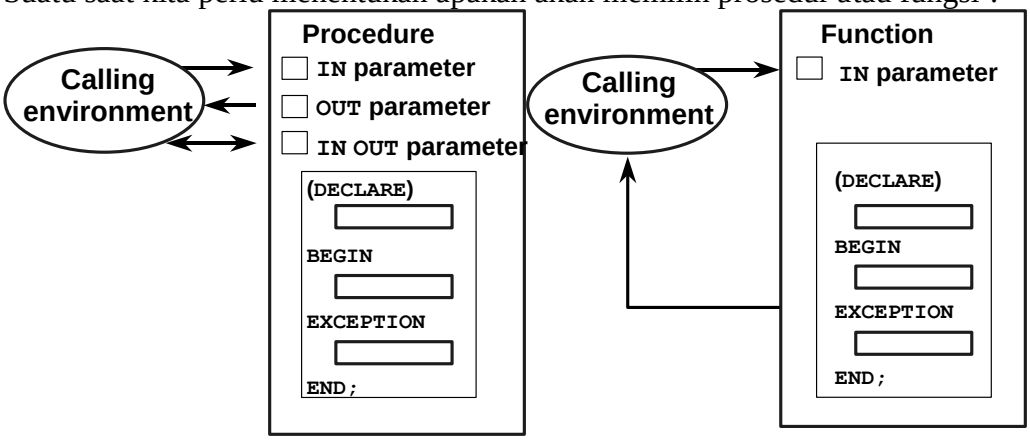
Contoh penghapusan fungsi :

```
DROP FUNCTION get_sal;
```

Function dropped.

11.11. Pilih Prosedur atau Fungsi ?

Suatu saat kita perlu menentukan apakah akan memilih prosedur atau fungsi ?



Prosedur dapat berisi 0,1, atau lebih dari satu parameter yang ditranfer ke dan dari calling environment, tapi prosedur tidak bisa mengembalikan suatu nilai.

Sedangkan fungsi dibuat pada saat kita perlu untuk melakukan komputasi pada suatu nilai yang harus dikembalikan ke calling environment. Fungsi dapat berisi 0,1, atau lebih dari satu parameter yang ditransfer dari calling environment. Fungsi harus mengembalikan satu nilai tunggal dimana nilai

tersebut dikembalikan melalui statement RETURN. Fungsi yang digunakan dalam SQL Statement tidak perlu memiliki mode parameter OUT atau IN OUT. Karena akan muncul pesan kesalahan jika fungsi dengan parameter OUT digunakan dalam SQL Statement.

11.12. Pembandingan Prosedur dan Fungsi

Procedures	Functions
Menjalankan PL/SQL statement	Dipanggil sebagai bagian dari ekspresi
Tidak berisi klausa `RETURN` pada header	Harus berisi klausa `RETURN` pada header
Dapat mengembalikan 0, 1 atau > 1 nilai	Harus mengembalikan sebuah nilai tunggal
Dapat berisi statement `RETURN`	Harus berisi sedikitnya satu statement `RETURN`

Berikut ini perbandingan antara Prosedur dan fungsi :

11.13. Keuntungan Penggunaan

Berikut ini keuntungan dari penggunaan Prosedur atau Fungsi :

- Meningkatkan unjuk kerja / performansi
- Memudahkan pemeliharaan (maintenance)
- Meningkatkan keamanan dan integritas data
- Meningkatkan kejelasan dari kode yang ditulis.

11.14. Ringkasan

Pada bab ini telah dipelajari :

- Fungsi sebagai block bernama pada PL/SQL yang mengembalikan sebuah nilai
- Fungsi dibuat dengan sintak CREATE FUNCTION
- Fungsi dipanggil sebagai bagian dari ekspresi
- Fungsi yang disimpan dalam database dapat dipanggil dalam SQL Statement
- Fungsi dapat dihapus dari database dengan menggunakan sintak DROP FUNCTION
- Secara umum kita gunakan prosedur untuk melakukan suatu tindakan dan fungsi untuk menghitung suatu nilai.

11.15. Latihan Soal

LATIHAN PRAKTIKUM

1. Buatlah fungsi untuk mencari nama pegawai dari tabel pegawai, berdasarkan gaji yang lebih dari 2000. Kemudian jalankan dengan perintah ‘select’?

2. Buatlah fungsi untuk menghitung gaji menggunakan PL/SQL,

   Gaji < 1000 => Low
   1000 ≤ Gaji < 3000 => Medium
   Gaji ≥ 3000 => High
   Gaji > 5000 atau Gaji <0 => Undifined

   Kemudian jalankan hasilnya dengan men-SELECT fungsi tersebut

3. Buat fungsi fn_monthHire untuk mendapatkan informasi mengenai bulan mulai bekerja karyawan. Dengan catatan bahwa hanya Januari saja yang ditampilkan, selain itu beri informasi Bukan Januari. Kemudian tampilkan dalam query SELECT untuk menampilkan semua data kode karyawan, nama, hire_date dan monthHire dari tabel employees. Contoh Tabel output :

EMPLOYEE_ID	LAST_NAME	HIRE_DATE	MONTH_HIRE
198	OConnell	21-Jun-07	Bukan Januari
199	Grant	13-Jan-08	Januari
200	Whalen	17-Sep-03	Bukan Januari
201	Hartstein	17-Feb-04	Bukan Januari

4. Buat fungsi fn_bulan untuk mendapatkan informasi mengenai bulan mulai bekerja karyawan. Dengan catatan bahwa hanya April dan Juni saja yang ditampilkan, selain itu beri informasi Bukan April dan Juni . Kemudian tampilkan dalam query SELECT untuk menampilkan semua data kode karyawan, nama, hire_date dan monthHire dari tabel emp.

5. Buat fungsi fn_info yang akan menampilkan informasi prosentase kenaikan, gaji lama dan gaji baru. Fungsi memiliki 1 parameter untuk menampung kode karyawan.
   Syarat kenaikan :
   - untuk kode departemen 10, 50, 110 diberikan kenaikan 5%
   - untuk kode departemen 60 diberikan kenaikan 10%
   - untuk kode departemen 20 dan 60 diberikan kenaikan 15%
   - untuk kode departemen lainnya tidak diberikan kenaikan

Kemudian tampilkan dalam query SELECT untuk menampilkan semua data kode karyawan, nama, dan info dari tabel employees. Contoh output:

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	INFORMASI
198	OConnell	50	kenaikan 5% , gaji awal 2600 gaji sekarang 2730
199	Grant	50	kenaikan 5% , gaji awal 2600 gaji sekarang 2730
204	Baer	70	tidak ada kenaikan gaji