

MODUL VI

PEMROGRAMAN pada FUNCTION dan PROCEDURE

TUJUAN

Setelah menyelesaikan modul ini, anda diharapkan dapat :

1. Mengenalkan mahasiswa tentang konsep pemrograman dengan fungsi dan *procedure/prosedur* pada mysql.
2. Mengenalkan mahasiswa tentang penggunaan pemrograman fungsi dan prosedur dalam basis data.

DASAR TEORI

Pada function dan procedure, kita dapat memasukkan logika pemrograman. Terdapat beberapa karakteristik pemrograman yang didukung oleh MySQL. Beberapa diantaranya adalah penggunaan variable, kendali kondisional dan perulangan.

Variable

Seperti pada pemrograman pada umumnya, kita dapat menggunakan variable local pada function dan procedure. Pendeklarasian variable memiliki sintaks sebagai berikut:

```
DECLARE var_name [, var_name] . . . type [DEFAULT value]
```

Nilai inialisasi variable dapat dilakukan menggunakan statement DEFAULT. Jika statement DEFAULT tidak digunakan, maka nilai inialisasi variable adalah NULL. Penamaan variable local bersifat case insensitive. Berikut adalah beberapa contoh deklarasi variable :

```
DECLARE total_sale INT
```

```
DECLARE x,y INT DEFAULT 0
```

Pemberian nilai ke sebuah variabel dilakukan dengan menggunakan statement SET. Hasil dari query juga dapat dimasukkan ke dalam variabel menggunakan SELECT . . . INTO. Berikut adalah contoh pemberian nilai ke variabel :

```
SET total_sale = 50;
```

```
SELECT COUNT(*) INTO numEmployee FROM employee;
```

Ruang lingkup variabel adalah diantara blok BEGIN – END dimana variabel tersebut didefinisikan. Variabel dapat diakses dari blok yang berada dalam blok dimana ia didefinisikan, kecuali pada blok

yang memiliki deklarasi nama variabel yang sama. Berikut adalah contoh penggunaan variabel dalam function dan stored procedure.

```
MariaDB [test]> DELIMITER ^-^
MariaDB [test]> CREATE FUNCTION addTax(salary FLOAT(8,2))
  -> RETURNS FLOAT (8,2)
  -> BEGIN
  -> DECLARE tax FLOAT DEFAULT 0.05;
  -> RETURN salary * (1-tax);
  -> END ^-^
Query OK, 0 rows affected (0.11 sec)

MariaDB [test]> DELIMITER ;
```

Pada contoh di atas, dibuat sebuah function dengan variabel bernama `tax`. Variabel ini diset memiliki nilai default 0.05 dan digunakan untuk mengubah nilai salary. Contoh di bawah ini menunjukkan penggunaan function `addTax`.

```
MariaDB [test]> DELIMITER ;
MariaDB [test]> SELECT first_name, addTax(salary) FROM employee;
+-----+-----+
| first_name | addTax(salary) |
+-----+-----+
| Jason      | 1172.83        |
| Alison     | 6328.69        |
| James      | 6217.54        |
| Celia      | 2227.54        |
| Robert     | 2218.04        |
| Linda      | 4106.64        |
| David      | 7502.89        |
| James      | 1171.14        |
| Caroline   | 7502.89        |
| Doe        | 3325.00        |
| John       | 950.00         |
+-----+-----+
11 rows in set (0.04 sec)
```

Nama variabel lokal seharusnya tidak sama dengan nama kolom dalam tabel database. Jika pada statement SQL seperti `SELECT` terdapat referensi ke kolom tabel dengan nama yang sama. MySQL mereferensikannya sebagai nama variabel. Berikut adalah contohnya :

```
MariaDB [test]> DELIMITER @@
MariaDB [test]> CREATE PROCEDURE checkScope()
  -> BEGIN
  -> DECLARE first_name VARCHAR(15) DEFAULT 'bob';
  -> SELECT id,first_name FROM employee WHERE first_name = first_name;
  -> END @@
Query OK, 0 rows affected (0.07 sec)

MariaDB [test]> DELIMITER ;
```

Memanggil procedure `checkScope` dengan perintah sebagai berikut :

```

MariaDB [test]> CALL checkScope();
+-----+
| id | first_name |
+-----+
| 1 | bob        |
| 2 | bob        |
| 3 | bob        |
| 4 | bob        |
| 5 | bob        |
| 6 | bob        |
| 7 | bob        |
| 8 | bob        |
| 9 | bob        |
| 10 | bob       |
| 11 | bob       |
+-----+
11 rows in set (0.00 sec)

```

Pada contoh di atas, ketika kita melakukan pemilihan SELECT untuk first_name, nilai yang ditampilkan adalah default dari variabel first_name yaitu 'bob'.

Kendali kondisional

Seperti layaknya pemrograman, kita juga dapat mendefinisikan kendali kondisional di dalam function dan procedure. Kendali kondisional yang disediakan dalam MySQL adalah IF dan CASE.

1. Kendali IF

Sintaks dasar dari IF adalah sebagai berikut :

```

IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF;

```

Nilai search_condition dievaluasi. Jika bernilai true, maka statement_list setelah THEN akan dijalankan. Namun jika bernilai false, maka statement_list pada ELSE yang dijalankan. Penggunaan banyak kondisi dapat dilakukan dengan statement ELSEIF.

Berikut adalah contoh penggunaan IF :

```

MariaDB [test]> DELIMITER &&
MariaDB [test]> CREATE FUNCTION hideSalary(salary FLOAT (8,2))
  -> RETURNS VARCHAR(20)
  -> BEGIN
  -> DECLARE sal VARCHAR(20);
  -> IF salary < 4000 THEN SET sal = 'Low Salary';
  -> ELSE SET sal = 'High Salary';
  -> END IF;
  -> RETURN sal;
  -> END &&
Query OK, 0 rows affected (0.10 sec)

MariaDB [test]> DELIMITER ;

```

Pemanggilan function hideSalary adalah sebagai berikut :

```

MariaDB [test]> SELECT first_name, last_name, hideSalary(salary)
  -> FROM employee;

```

first_name	last_name	hideSalary(salary)
Jason	Martin	Low Salary
Alison	Mathews	High Salary
James	Smith	High Salary
Celia	Rice	Low Salary
Robert	Black	Low Salary
Linda	Green	High Salary
David	Larry	High Salary
James	Cat	Low Salary
Caroline	Grass	High Salary
Doe	NULL	Low Salary
John	Lenon	Low Salary

```

11 rows in set (0.00 sec)

```

2. Kendali CASE

Sintaks dari kendali CASE adalah sebagai berikut :

```

CASE case_value
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list] . . .
  [ELSE statement_list]
END CASE

```

Pada sintaks di atas, case_value dibandingkan dengan semua nilai when_value sampai ditemukan yang sesuai. Jika ditemukan, maka statement_list pada WHEN yang bersesuaian akan dijalankan. Jika tidak ada nilai when_value yang sesuai, maka statement_list pada ELSE yang akan dijalankan (jika ada).

Berikut ini adalah contoh penggunaan CASE :

```

MariaDB [test]> DELIMITER &&
MariaDB [test]> CREATE FUNCTION calcTax(job VARCHAR (20))
  -> RETURNS FLOAT(3,2)
  -> BEGIN
  -> DECLARE tax FLOAT(3,2) DEFAULT 0.05;
  -> CASE job
  -> WHEN 'Manager' THEN SET tax = 0.1;
  -> WHEN 'Programmer' THEN SET tax = 0.07;
  -> WHEN 'Tester' THEN SET tax = 0.06;
  -> ELSE SET tax = 0.05;
  -> END CASE;
  -> RETURN tax;
  -> END &&
Query OK, 0 rows affected (0.08 sec)

MariaDB [test]> DELIMITER ;

```

Pemanggilan function calcTax adalah sebagai berikut :

```

MariaDB [test]> SELECT first_name, last_name, calcTax(description) FROM employee;
+-----+-----+-----+
| first_name | last_name | calcTax(description) |
+-----+-----+-----+
| Jason      | Martin    | 0.07                 |
| Alison     | Mathews   | 0.06                 |
| James      | Smith     | 0.06                 |
| Celia      | Rice      | 0.10                 |
| Robert     | Black     | 0.06                 |
| Linda      | Green     | 0.06                 |
| David      | Larry     | 0.10                 |
| James      | Cat       | 0.06                 |
| Caroline   | Grass     | 0.06                 |
| Doe        | NULL      | 0.05                 |
| John       | Lenon     | 0.07                 |
+-----+-----+-----+
11 rows in set (0.00 sec)

```

Bentuk sintaks CASE yang lain adalah sebagai berikut:

```

CASE case_value
  WHEN search_condition THEN statement_list
  [WHEN search_condition THEN statement_list] . . .
  [ELSE statement_list]
END CASE

```

Pada sintaks di atas, search_condition pada setiap klausa WHEN dievaluasi hingga ditemukan klausa WHEN yang sesuai. Jika tidak ada klausa WHEN yang sesuai, maka klausa ELSE yang dijalankan. Jika tidak ada klausa ELSE ketika semua klausa WHEN tidak sesuai, maka akan terjadi case not found for CASE statement error. Berikut adalah contoh penggunaan sintaks CASE ... WHEN tersebut :

```

MariaDB [test]> DELIMITER >>
MariaDB [test]> CREATE FUNCTION calcTax2(job VARCHAR(20))
  -> RETURNS FLOAT(3,2)
  -> BEGIN
  -> DECLARE tax FLOAT(3,2);
  -> CASE
  -> WHEN job = 'Manager' THEN SET tax=0.1;
  -> WHEN job = 'Programmer' THEN SET tax=0.07;
  -> WHEN job = 'Tester' THEN SET tax=0.06;
  -> ELSE SET tax=0.05;
  -> END CASE;
  -> RETURN tax;
  -> END >>
Query OK, 0 rows affected (0.06 sec)

MariaDB [test]> DELIMITER ;

```

Pemanggilan function :

```

MariaDB [test]> SELECT first_name, last_name, calcTax2(description)
  -> FROM employee;

```

first_name	last_name	calcTax2(description)
Jason	Martin	0.07
Alison	Mathews	0.06
James	Smith	0.06
Celia	Rice	0.10
Robert	Black	0.06
Linda	Green	0.06
David	Larry	0.10
James	Cat	0.06
Caroline	Grass	0.06
Doe	NULL	0.05
John	Lenon	0.07

```

11 rows in set (0.00 sec)

```

Perulangan

Pada function dan procedure juga disediakan perulangan. Beberapa bentuk perulangan yang disediakan dalam MySQL adalah WHILE, REPEAT ... UNTIL dan LOOP.

a. Perulangan WHILE

Bentuk sintaks untuk perulangan WHILE adalah sebagai berikut :

```

WHILE search_condition DO
    Statement_list
END WHILE

```

Statement_list yang terdapat pada WHILE diulang selama search_condition bernilai true. Statement_list terdiri atas satu atau lebih statement SQL, setiap statemennya dipisahkan dengan delimiter titik koma (;). Berikut adalah contoh penggunaan WHILE :

```

MariaDB [test]> DELIMITER //
MariaDB [test]> CREATE PROCEDURE mod12(IN number INT(10))
  -> BEGIN
  -> WHILE number MOD 12 > 0 DO
  -> SET number = number + 1;
  -> END WHILE;
  -> SELECT number;
  -> END //
Query OK, 0 rows affected (0.08 sec)

MariaDB [test]> DELIMITER ;

```

Pemanggilan procedure sebagai berikut :

```

MariaDB [test]> CALL mod12(10);
+-----+
| number |
+-----+
|      12 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> CALL mod12(24);
+-----+
| number |
+-----+
|      24 |
+-----+
1 row in set (0.00 sec)

```

b. Perulangan REPEAT ... UNTIL

Sintaks dari REPEAT ... UNTIL dalah sebagai berikut :

```

REPEAT
    Statement_list
UNTIL search_condition
END REPEAT

```

Statement_list di dalam REPEAT dilakukan secara berulang hingga ekspresi search_condition bernilai true. Oleh karena itu, sebuah REPEAT memasuki perulangan paling sedikit sebanyak satu kali. Statement_list terdiri atas satu atau lebih statement, masing-masing dipisah dengan delimiter titik koma (;). Berikut adalah contoh penggunaan REPEAT ... UNTIL:

```

MariaDB [test]> DELIMITER //
MariaDB [test]> CREATE PROCEDURE repeatDemo(IN number INT(10))
  -> BEGIN
  -> REPEAT
  -> SET number = number + 1;
  -> UNTIL number MOD 12 = 0
  -> END REPEAT;
  -> SELECT number;
  -> END //
Query OK, 0 rows affected (0.09 sec)

MariaDB [test]> DELIMITER ;

```

```

MariaDB [test]> CALL repeatDemo(10);
+-----+
| number |
+-----+
|      12 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> CALL repeatDemo(14);
+-----+
| number |
+-----+
|      24 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

MariaDB [test]> CALL repeatDemo(38);
+-----+
| number |
+-----+
|      48 |
+-----+
1 row in set (0.00 sec)

```

c. Perulangan LOOP

Sintaks dari perulangan LOOP adalah sebagai berikut :

```

[begin_label:] LOOP
    Statement_list
END LOOP [end_label]

```

LOOP merupakan bentuk perulangan sederhana. Perulangan dilakukan terhadap `statement_list`, yang terdiri dari beberapa statement dengan dipisahkan tanda titik koma(;). Statement di dalam LOOP diulang sampai LOOP berakhir. Cara mengakhiri LOOP biasanya dilakukan dengan statement `LEAVE`. Tanda perulangan dilakukan menggunakan `ITERATE`. Berikut adalah contohnya :


```

MariaDB [test]> DELIMITER //
MariaDB [test]> CREATE PROCEDURE iterateDemo(number INT)
-> BEGIN
-> label1: LOOP
->     SET number = number + 1;
->     IF number MOD 12 > 0 THEN
->         ITERATE label1;
->     END IF;
->     LEAVE label1;
-> END LOOP label1;
-> SELECT number;
-> END //
Query OK, 0 rows affected (0.05 sec)

MariaDB [test]> DELIMITER ;
MariaDB [test]> CALL iterateDemo(10);
+-----+
| number |
+-----+
|      12 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

```

LATIHAN

1. Buatlah sebuah function bernama hitungDiskon yang menerima parameter harga INT, lalu mengembalikan nilai harga setelah diskon 10%.
2. Buatlah sebuah procedure bernama cekStatusGaji yang menerima parameter gaji INT. Jika gaji di atas 10 juta, tampilkan 'Gaji Tinggi', jika di bawah 5 juta tampilkan 'Gaji Rendah', selain itu tampilkan 'Gaji Menengah'.
3. Buatlah sebuah procedure bernama jumlahAngka yang menjumlahkan bilangan dari 1 hingga N (parameter input) menggunakan perulangan WHILE.
4. Buat sebuah procedure bernama ulangCetak yang mencetak 'Hello' sebanyak N kali menggunakan perulangan REPEAT ... UNTIL.