Function dan Stored Procedure

Function dan *Stored Procedure* merupakan fitur utama yang paling penting di MySQL 5. Function dan Stored Procedure merupakan suatu kumpulan perintah atau statement yang disimpan dan dieksekusi di server database MySQL.

Dengan SP (Stored Procedure), kita dapat menyusun program sederhana berbasis sintaks SQL untuk menjalankan fungsi tertentu. Hal ini menjadikan aplikasi yang kita buat lebih efektif dan efisien. Berikut ini beberapa keuntungan menggunakan Stored Procedure:

- **Lebih cepat**.

  Hal ini karena kumpulan perintah query dijalankan langsung di server. Berbeda dengan jika dijalankan secara sekuensial di bahasa pemrograman, akan lebih lambat karena harus *"bolak-balik"* antara client dan server.

- **Menghilangkan duplikasi proses, pemeliharaan yang mudah.**

  Pada dasarnya operasi yang terjadi di suatu aplikasi terhadap database adalah sama. Secara umum, di dalam aplikasi biasanya terdapat operasi untuk validasi data inputan, menambahkan record baru, mengubah record, menghapus record dan sebagainya. Dengan SP, mungkin kita dapat menghindari adanya duplikasi proses yang kurang lebih sama, sehingga pemeliharaannya juga jadi lebih mudah.

- **Meningkatkan keamanan database.**

  Dengan adanya SP, database akan lebih aman karena aplikasi yang memanggil SP tidak perlu mengetahui isi di dalamnya. Sebagai contoh, dalam proses menambahkan data (insert), kita membuat suatu SP khusus. Dengan demikian, saat client atau aplikasi akan menambahkan data (insert) maka tidak perlu tahu nama tabelnya, karena hanya cukup memanggil SP tersebut dengan mengirimkan parameter yang diinginkan. Selanjutnya, Stored Procedure dari segi bentuk dan sifatnya terbagi menjadi 2 (dua), yaitu `FUNCTION` dan

PROCEDURE. Perbedaan utama antara function dan procedure adalah terletak pada nilai yang dikembalikannya (di-return). Function memiliki suatu nilai yang dikembalikan (di-return), sedangkan procedure tidak.

Umumnya suatu procedure hanya berisi suatu kumpulan proses yang tidak menghasilnya value, biasanya hanya menampilkan saja. *Sebagai catatan bahwa dalam buku ini jika terdapat istilah SP (Stored Procedure) maka yang dimaksud adalah Function dan Procedure.*

HELLO WORLD!

Sebagai contoh sederhana, kita akan membuat suatu SP yang akan menampilkan string “Hello World!” di layar hasil. Berikut ini perintah query untuk membuat SP tersebut:

```
1    DELIMITER $$
2    CREATE PROCEDURE hello()
3    BEGIN
4    SELECT "Hello World!";
5    END$$
6    DELIMITER ;
```

Untuk memanggil procedure tersebut, gunakanlah CALL. Berikut ini contoh pemanggilan procedure dan hasil tampilannya:

```
1     CALL hello();
```

Hasilnya sebagai berikut:

```
1     +--------------+
2     | Hello World! |
3     +--------------+
4     | Hello World! |
5     +--------------+
```

MEMBUAT, MENGUBAH DAN MENGHAPUS SP

MEMBUAT SP

Untuk membuat SP baru, berikut ini bentuk umumnya:

```
CREATE
  [DEFINER = { user | CURRENT_USER }]
  PROCEDURE sp_name ([ proc_parameter[,...]])
  [ characteristic ...] routine_body

CREATE
  [DEFINER = { user | CURRENT_USER }]
  FUNCTION sp_name ([ func_parameter[,...]])
  RETURNS type
  [ characteristic ...] routine_body
```

Contoh 1. Procedure untuk menghitung jumlah pelanggan

```
DELIMITER $$
CREATE PROCEDURE jumlahPelanggan()
   BEGIN
      SELECT COUNT(*) FROM pelanggan;
   END$$
DELIMITER ;
```

Cara pemanggilan dari procedure di atas adalah dengan menggunakan **CALL jumlahPelanggan()**. Hasilnya akan ditampilkan jumlah record dari tabel pelanggan. Berikut ini bentuk lain dari contoh di atas:

```
DELIMITER $$
CREATE PROCEDURE jumlahPelanggan2(OUT hasil AS INT)
   BEGIN
     SELECT COUNT(*) INTO hasil FROM pelanggan;
END$$
DELIMITER ;
```

Pada bentuk procedure yang kedua di atas (**jumlahPelanggan2**), kita menyimpan hasil dari procedure ke dalam satu variabel bernama **hasil** yang bertipe **INT**. Perbedaan dari kedua bentuk di atas adalah, pada bentuk kedua, kita dapat memanggil procedure dengan SELECT, sedangkan pada yang pertama tidak bisa. Berikut ini contoh pemanggilan untuk procedure yang kedua:

```
mysql> CALL jumlahPelanggan2(@jumlah);
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT @jumlah AS `Jumlah Pelanggan`;
```

Hasilnya menjadi sebagai berikut:

```
+------------------+
| Jumlah Pelanggan |
+------------------+
| 5                |
+------------------+
1 row in set (0.02 sec)
```

Contoh 2. Procedure untuk menghitung jumlah item barang yang pernah dibeli oleh satu pelanggan.

```
DELIMITER $$
CREATE PROCEDURE
    jumlahItemBarang (pelanggan VARCHAR(5))
    BEGIN
       SELECT SUM(detil_pesan.jumlah)
       FROM pesan, detil_pesan
       WHERE pesan.id_pesan=detil_pesan.id_pesan
       AND pesan.id_pelanggan=pelanggan;
   END$$
DELIMITER ;
```

Contoh 3. Function untuk menghitung jumlah produk yang tersedia (stock) untuk satu produk tertentu.

```
DELIMITER $$
CREATE FUNCTION jumlahStockBarang(produk VARCHAR(5))
      RETURNS INT

      BEGIN
          DECLARE jumlah INT;
          SELECT COUNT(*) INTO jumlah FROM produk
          WHERE id_produk=produk;
          RETURN jumlah;
      END$$
DELIMITER ;
```

Untuk memanggil suatu function, kita tidak menggunakan CALL, tetapi langsung dapat memanggil dengan SELECT. Berikut ini contoh pemanggilan untuk fungsi di atas.

```
SELECT jumlahStockBarang('B0001');
```

Dan berikut ini hasilnya:

```
+----------------------------+
| jumlahStockBarang('B0001') |
+----------------------------+
| 1                          |
+----------------------------+
```

MENGUBAH SP

Untuk mengubah SP yang sudah ada, berikut ini bentuk umumnya:

```
ALTER { PROCEDURE | FUNCTION} sp_name
[ characteristic ...]
```

MENGHAPUS SP

Untuk menghapus SP yang sudah ada, berikut ini bentuk umumnya:

```
DROP { PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

SINTAKS DASAR DALAM SP

SP dapat dikatakan sebagai bahasa pemrograman yang berada di dalam database. Oleh karena itu, tentunya terdapat sintaks-sintaks tertentu berhubungan dengan SP tersebut, misalnya bagaimana untuk mendeklarasikan variabel, penyeleksian kondisi, perulangan dsb. Pada bagian ini akan diuraikan beberapa sintaks dasar SP yang didukung oleh MySQL.

Variabel

Variabel digunakan untuk menyimpan suatu nilai secara temporer (sementara) di memory. Variabel akan hilang saat sudah tidak digunakan lagi. Variabel dalam MySQL sebelum dapat digunakan, pertama kali harus dideklarasikan terlebih dahulu. Berikut ini bentuk umum pendeklarasian suatu variabel di MySQL:

```
1     DECLARE variable_name DATATYPE [ DEFAULT value];
```

Contohnya:

```
1     DECLARE jumlah INT;
2     DECLARE kode VARCHAR(5);
3     DECLARE tgl_lahir DATE DEFAULT '1982-10-20';
```

Setelah dideklarasikan, suatu variabel dapat diisi dengan suatu nilai sesuai dengan tipe data yang didefinisikan saat pendeklarasian. Untuk mengisikan nilai ke dalam suatu variabel, digunakan perintah SET. Format umumnya sebagai berikut:

```
1     SET variable_name = expression|value;
```

Contohnya:

```
1     SET jumlah = 10;
2     SET kode = (SELECT id_pelanggan FROM pelanggan LIMIT 1);
3     SET tgl_lahir = now();
```

Berikut ini contoh function **hitungUmur()** untuk menghitung umur seseorang saat ini berdasarkan tahun kelahiran yang diberikan.

```
1      DELIMITER $$
2      CREATE FUNCTION hitungUmur (lahir DATE)
3           RETURNS INT
4           BEGIN
5               DECLARE thn_sekarang, thn_lahir INT;
6               SET thn_sekarang = YEAR(now());
7               SET thn_lahir = YEAR (lahir);
8               RETURN thn_sekarang - thn_lahir;
9           END$$
```

```
10      DELIMITER ;
```

PENYELEKSIAN KONDISI

Dengan adanya fasilitas penyeleksian kondisi, kita dapat mengatur alur proses yang terjadi dalam database kita. Di MySQL, penyeleksian kondisi terdiri dari IF, IF…ELSE dan CASE. Berikut ini bentuk umum ketiga perintah tersebut:

```
1       IF kondisi THEN
2            perintah-jika-benar;
3       END IF;
4       IF kondisi THEN
5           perintah-jika-benar;
6       ELSE
7          perintah-jika-salah;
8       END IF;
9       CASE e x p r e s s i o n
10         WHEN v a l u e THEN
11            s t a t e m e n t s
12        [WHEN value THEN
13            s t a t e m e n t s ...]
14        [ELSE
15            s t a t e m e n t s]
16      END CASE;
```

Berikut ini contoh penggunaan perintah IF dalam fungsi **cekPelanggan**() dimana fungsi ini memeriksa apakah pelanggan sudah pernah melakukan transaksi pemesanan barang. Jika sudah pernah, tampilkan pesan berapa kali melakukan pemesanan, jika belum tampilkan pesan belum pernah memesan.

```
1       DELIMITER $$
2       CREATE FUNCTION cekPelanggan (pelanggan varchar(5))
3              RETURNS VARCHAR (100)
4
5              BEGIN
6                  DECLARE jumlah INT;
7                  SELECT COUNT(id_pesan) INTO jumlah FROM pesan
8                     WHERE id_pelanggan=pelanggan;
9                  IF (jumlah > 0) THEN
10                    RETURN CONCAT("Anda sudah bertransaksi sebanyak ",
11                    jumlah, " kali");
12                 ELSE
13                    RETURN "Anda belum pernah melakukan transaksi";
14                 END IF;
15           END$$
16      DELIMITER ;
```

Dan berikut ini contoh penggunaan perintah CASE dalam fungsi **getDiskon**() di mana fungsi ini menentukan diskon berdasarkan jumlah pesanan yang dilakukan.

```
DELIMITER $$
CREATE FUNCTION getDiskon(jumlah INT) RETURNS int(11)
     BEGIN
       DECLARE diskon INT;
       CASE
           WHEN (jumlah >= 100) THEN
              SET diskon = 10;
           WHEN (jumlah >= 50 AND jumlah < 100) THEN
              SET diskon = 5;
           WHEN (jumlah >= 20 AND jumlah < 50) THEN
              SET diskon = 3;
           ELSE SET diskon = 0;
      END CASE;
      RETURN diskon;
   END$$
DELIMITER ;
```

PERULANGAN

Selain penyeleksian kondisi, MySQL juga mendukung adanya perulangan dalam querynya. Perulangan biasanya digunakan untuk mengulang proses atau perintah yang sama. Dengan perulangan, perintah akan lebih efisien dan singkat.

Berikut ini bentuk-bentuk perintah perulangan:

```
[label:] LOOP
     statements
END LOOP [label];

[label:] REPEAT
     statements
UNTIL expression
END REPEAT [label]

[label:] WHILE expression DO
statements
END WHILE [label]
```

Contoh perulangan dengan LOOP

```
SET i=1;
ulang: WHILE i<=10 DO
    IF MOD(i,2)<>0 THEN
        SELECT CONCAT(i," adalah bilangan ganjil");
    END IF;
    SET i=i+1;
END WHILE ulang;
```

Cara menampilkan Triger, Store Procedure dan Function

```
SELECT ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE="PROCEDURE" AND ROUTINE_SCHEMA="penjualan";

SELECT ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE="FUNCTION" AND ROUTINE_SCHEMA="penjualan";

SELECT TRIGGER_NAME FROM INFORMATION_SCHEMA.TRIGGERS
WHERE TRIGGER_SCHEMA="penjualan";
```

Referensi:

Achmad Solichin, 2010, MySQL 5 : Dari Pemula Hingga Mahir