DIKTAT

Dasar-dasar Stored Procedure dan Function (berbasis MySQL)

Oleh : Cosmas Haryawan

Pendahuluan

Stored Procedure dan Function adalah rangkaian program yang disimpan dalam database dan dapat dipanggil oleh program lain atau melalui SQL Prompt

Stored procedured ditulis dalam bentuk suatu Script.

Keuntungan

- Cepat, kompilasi dilakukan di Database (kadang disebut “pre -compilation”) sehingga mengurangi traffic
- Adanya pemisahan antara database access logic dengan application logic sehingga program aplikasi menjadi lebih sederhana dan lebih ringkas (thin client concept)
- Berupa obyek dalam database, sehingga menghilangkan ketergantungan terhadap bahasa program yang digunakan
- Bersifat Portable, jika bisa berjalan di database tsb maka dipastikan jika database bi sa terinstall di manapun maka store procedure pasti bisa dijalankan

Bentuk Umum

Sintak untuk membuat :

Procedure

```
CREATE PROCEDURE sp_name ([proc_parameter [,...]])
    [characteristic ..]
    routine_body
```

Function

```
CREATE FUNCTION sp_name ([func_parameter [,...]])
    RETURNS type
    [characteristic ..]
    routine_body
```

Keterangan :

Proc_parameter:

    [IN | OUT | INOUT] param_name type

Func_parameter

    Param_name type

Type :

    Semua type data yang valid di MySQL.

Characteristic:

    LANGUAGE SQL

    [NOT] DETERMINISTIC

    {CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

SQL SECURITY {DEFINER | INVOKER }

COMMENT 'string'

Routine_body:

Statement SQL procedure yang valid.

Penjelasan

Secara default, routine berasosiasi dengan default database. Untuk berasosiasi secara eksplisit dengan database yang diberikan, secara spesifik diberi nama db_name.sp_name pada saat membuat routine. Jika tidak ada parameter maka empty parameter digunakan dengan menggunakan ( ). Setiap parameter memiliki IN parameter sebagai default. IN, OUT, INOUT parameter hanya valid untuk procedure, sedangkan untuk function hanya IN parameter saja.

RETURN type hanya berlaku untuk function.

ROUTINE_BODY berisi statement SQL yang valid. Dapat berisi statement sederhana seperti SELECT atau INSERT atau berisi gabungan beberapa statement yang dapat ditulis dengan menggunakan BEGIN .. END. Compound statement dapat berisi deklarasi, loop dan struktur kontrol yang lain.

Sintak untuk mengubah

Untuk mengubah stored procedure atau function pertama kali yang dilakukan adalah menghapus terlebih dahulu procedure / function nya kemudian baru dibuat kembali.

Sintak untuk menghapus

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

Contoh:

```
Drop Procedure spDafGaji;
```

Sintak untuk memanggil

```
CALL sp_name
```

Contoh:

```
Call spDafGaji();
```

Sintak untuk melihat daftar / list fungsi dan prosedur

```
SHOW CREATE {PROCEDURE|FUNCTION} sp_name;
```

Contoh:

```
Show create procedure spDafGaji;
```

Sintak untuk melihat daftar / list fungsi dan prosedur

```
SHOW {PROCEDURE|FUNCTION} status;
```

Contoh :

```
Show procedure status;
Show function status;
```

Contoh Pembuatan Stored Procedure

```
mysql> CREATE PROCEDURE sp_daftar_pegawai()
    -> select nip,namapeg,p.kodejabat,namajabat
    -> from pegawai p left join jabatan j
    -> on j.kodejabat = p.kodejabat;
```

Untuk Memanggil :

```
mysql> call sp_daftar_pegawai();
```

Contoh Stored Procedure dengan Ekspresi

```
mysql> CREATE PROCEDURE sp_detail_jual()
    -> select nojual,d.kodebrg,namabrg,qty,
    -> harga, qty * harga as subtotal
    -> from djual d left join barang b
    -> on b.kodebrg = b.kodebrg;
```

STORED PROCEDURE dengan PARAMETER

Parameter merupakan variabel memori yang digunakan untuk menerima suatu nilai dari pemangilnya.

Stored procedure dapat menggunakan parameter sehingga program aplikasi yang mem anggil stored procedure dapat mengakses database yang diperlukan sesuai dengan kondisi yang diminta yaitu dengan cara mengirimkan suatu nilai ke Stored Procedure melalui parameter.

Terdapat 3 mode parameter yaitu :IN, OUT dan INOUT :

- IN (default) → akan mempassingkan nilai konstan dari memori ke stored procedure
- OUT → akan mengambil nilai dari prosedur
- IN OUT → akan mempassingkan nilai dari memori ke dalam procedure dan memungkinkan nilai yang berbeda dari prosedur dikembalikan ke memori dengan menggunaka n parameter yang sama.

Secara default stored procedure / function memiliki parameter IN, sehingga Untuk Parameter IN, kata IN tidak perlu ditambahkan sebelum nama parameter.

Contoh dengan Parameter IN

Dimisalkan dibuat SP untuk menampilkan data pegaw ai untuk jabatan tertentu saja, sehingga saat dipanggil diperlukan parameter berupa kode jabatan yang akan ditampilkan

```
mysql> CREATE PROCEDURE sp_peg_jabatan(kdjabat char(2))
    -> select nip,namapeg,p.kodejabat,namajabat
```

```
    -> from pegawai p left join jabatan j
    -> on j.kodejabat = p.kodejabat
    -> WHERE p.kodejabat = kdjabat;
```

Untuk Memanggil :

```
mysql> call sp_peg_jabatan('01');
```

→ parameter berupa kode jabatan berupa nilai '01'

Contoh dengan parameter OUT

Dimisalkan dibuat Stored Procedure untuk mengetahui jumlah Pegawai

Diperlukan parameter OUT untuk menampung hasil perhitungan jumlah pegawai

```
mysql> CREATE PROCEDURE sp_jum_peg (OUT jum int)
    -> SELECT count(*) INTO jum FROM pegawai;
Query OK, 0 rows affected (0.00 sec)
mysql> call sp_jum_peg(@n);
mysql> select @n
+-----+
| @n  |
+-----+
|   4 |
+-----+
1 row in set (0.08 sec)
```

Contoh dengan parameter INOUT

Dibuat Stored Procedure untuk merubah tampilan No Telepon, dibutuhkan parameter masukan notelp yang sekaligus akan digunakan sebagai hasil keluaran

```
mysql> CREATE PROCEDURE
    ->  sp_telpon (INOUT notelp varchar(20))
    ->    SELECT CONCAT("(",left(notelp,3),") ",
    ->    substring(notelp,4,3),"-",substring(notelp,7))
    ->    INTO notelp;

mysql> SET @tlp = '021234567';

mysql> call sp_telpon(@tlp);

mysql> select @tlp;
+----------------+
| @tlp           |
+----------------+
| (021) 234-567  |
+----------------+
```

Contoh dengan parameter IN dan OUT

Akan dibuat Stored Procedure untuk mengetahui Jumlah Pegawai untuk jabatan tertentu, maka dibutuhkan parameter yang dikirimkan berupa kode jabatan (kdjabat) dan juga parameter untuk menampung hasil perhitungan (jum)

```
mysql> CREATE PROCEDURE
    ->   sp_jum_peg_jabat(IN kdjabat char(2), OUT jum int)
    ->    SELECT count(*) INTO jum FROM pegawai
    ->    WHERE kodejabat = kdjabat;
mysql> call sp_jum_peg_jabat('02',@n);
mysql> select @n;
+------+
| @n   |
+------+
| 3    |
+------+
```

Begin – End Syntax

Kadangkala dalam suatu stored routines dan trigger dibutuhkan untuk menulis beberapa buah statement sekaligus.

Gabungan statement / compound statement tersebut diawali dengan BEGIN dan diakhiri dengan END. Diantara BEGIN ... END, dapat terdiri dari satu atau banyak statement dan masing -masing statement harus diakhiri dengan tanda semicolon (;),

Karena setiap statement harus diakhiri dengan semikolon (;) maka diperlukan untuk mengganti delimiter dari ; menjadi delimiter yang diinginkan, misalnya dengan menggunakan | atau // atau $$. Pengubahan delimiter ini akan membuat setiap statement dalam stor ed routine dapat menggunakan ;

Contoh :

Untuk memudahkan penjelasan akan dibuat tabel kelompok :

```
mysql> create table kelompok (
    -> kode char(3) not null primary key,
    -> nama varchar(20) not null);
```

Kemudian akan dibuat Stored Procedure untuk mengisikan data kelompok sehingga dibutuhkan parameter kode kelompok dan nama kelompok

Stored Procedure tersebut akan menghandle, jika kode sudah dimasukkan maka otomatis akan dilakukan peng-update-an, tetapi jika belum dimasukkan maka akan dilakukan penyisipa n data baru

Pembuatan Stored Procedure

```
mysql> DELIMITER |
mysql> CREATE PROCEDURE
    ->   sp_isi_kelompok ( kd char(3), nm varchar(20))
    ->   BEGIN
```

```
    ->     IF (EXISTS(select kode from kelompok where kode= kd))
    ->     THEN
    ->       UPDATE kelompok SET nama = nm WHERE kode = kd;
    ->     ELSE
    ->       INSERT INTO kelompok (kode,nama) VALUES (kd,nm);
    ->     END IF;
    ->   END;
    -> |
mysql> DELIMITER ;
```

Hasil :

```
mysql> call sp_isi_kelompok('01','FOOD');
Query OK, 1 row affected (0.09 sec)

mysql> select * from kelompok;
+------+------+
| kode | nama |
+------+------+
| 01   | FOOD |
+------+------+
1 row in set (0.00 sec)

mysql> call sp_isi_kelompok('01','MAKANAN');
Query OK, 1 row affected (0.05 sec)

mysql> select * from kelompok;
+------+---------+
| kode | nama    |
+------+---------+
| 01   | MAKANAN |
+------+---------+
1 row in set (0.00 sec)
```

Function

Perbedaan Utama Function dan Stored Procedure adalah :

- Function bisa mengembalikan suatu nilai balik (return Value)
- Pada Function, parameter yang diperbolehkan hanya parameter IN
- Function bisa langsung dipanggil dari perintah SELECT SQL

Contoh :

```
mysql> delimiter |
mysql> CREATE FUNCTION f_jum_peg (kdjabat char(2))
    -> RETURNS int
    -> BEGIN
    ->   DECLARE jum int;
    ->   select count(*) into jum from pegawai
    ->    where kodejabat = kdjabat;
```

```
    ->   RETURN jum;
    -> END;
    -> |
mysql> delimiter ;

mysql> select kodejabat, namajabat, f_jum_peg(kodejabat)
    ->   from jabatan;
+-----------+-----------+----------------------+
| kodejabat | namajabat | f_jum_peg(kodejabat) |
+-----------+-----------+----------------------+
| 01        | DIREKTUR  |                    1 |
| 02        | KABAG     |                    3 |
+-----------+-----------+----------------------+
2 rows in set (0.00 sec)
```

Contoh :

```
mysql> delimiter |
mysql> CREATE FUNCTION f_jurusan (kode char(2))
    -> RETURNS varchar(30)
    -> BEGIN
    ->   DECLARE namajur varchar(30);
    ->   CASE kode
    ->     WHEN 'TI' THEN
    ->        SET namajur = 'TEKNIK INFORMATIKA';
    ->     WHEN 'SI' THEN
    ->        SET namajur = 'SISTEM INFORMASI';
    ->     WHEN 'MI' THEN
    ->        SET namajur = 'MANAJEMEN INFORMATIKA';
    ->     WHEN 'KA' THEN
    ->        SET namajur = 'KOMPUTERISASI AKUNTANSI';
    ->     WHEN 'TK' THEN
    ->        SET namajur = 'TEKNIK KOMPUTER';
    ->     ELSE SET namajur = 'KODE JURUSAN SALAH';
    ->   END CASE;
    ->   RETURN namajur;
    -> END;
    -> |
```

Hasil :

```
mysql> delimiter ;
mysql> select f_jurusan('SI');
+------------------+
| f_jurusan('SI')  |
+------------------+
| SISTEM INFORMASI |
+------------------+
1 row in set (0.00 sec)
```

```
mysql> select nim,nama,f_jurusan(jurusan)
    ->  from mahasiswa;
+-----+-------+----------------------+
| nim | nama  | f_jurusan(jurusan)   |
+-----+-------+----------------------+
| 001 | TOTOK | TEKNIK INFORMATIKA   |
| 002 | TITIK | SISTEM INFORMASI     |
| 003 | TATAK | MANAJEMEN INFORMATIKA |
| 004 | TUTUK | SISTEM INFORMASI     |
+-----+-------+----------------------+
4 rows in set (0.00 sec)
```