BAB VI
STORED FUNCTION

A. Tujuan

1. Mahasiswa dapat memahami dan menjelaskan perbedaan penggunaan antara *stored procedure* dan *stored function*.
2. Mahasiswa dapat memahami dan menjelaskan struktur sintaks pembuatan *stored function*.
3. Mahasiswa dapat mendemonstrasikan pembuatan *stored function* untuk penerapan formula perhitungan terhadap data.

B. Uraian Materi

MySQL menyediakan fungsi-fungsi *built-in* yang dapat digunakan untuk mengelola data di dalam basis data seperti fungsi *string*, fungsi *date time*, fungsi untuk perhitungan, fungsi untuk pemformatan data, dan lain sebagainya. Fungsi-fungsi tersebut dapat dipanggil di dalam perintah *query* DML seperti "SELECT", "INSERT", "UPDATE", dan "DELETE" (Kruckenberg & Pipes 2005). Selain fungsi-fungsi *built-in* tersebut, MySQL juga menawarkan pembuatan *user defined function* atau fungsi yang didefinisikan oleh pengguna. Seperti halnya *stored procedure*, *stored function* adalah program tersimpan di dalam *database*. Perbedaannya *stored function* dengan *stored procedure* adalah *stored function* harus memiliki nilai kembalian yang dapat dibaca melalui sintaks SQL, sehingga di dalam *stored function* harus dideklarasikan klausa "RETURN" beserta tipe data nilai kembaliannya (DuBois et.al 2005). Umumnya, *stored function* digunakan untuk mengimplementasikan perhitungan yang kompleks dan tidak dapat diselesaikan oleh perintah SQL serta di dalam perhitungan membutuhkan pernyataan kondisi dan logika perulangan (DuBois 2014). Selain itu, perhitungan yang dibuat di dalam *stored function* juga dapat digunakan berulang kali oleh beberapa aplikasi yang berbeda. Sebagai contoh, untuk mendapatkan nilai IPK, mendapatkan jumlah mata kuliah, jumlah mahasiswa yang mengikuti suatu mata kuliah dan lain sebagainya. Contoh lainnya adalah untuk menghitung

total penjualan, jumlah diskon yang diberikan berdasarkan nilai penjualan, dan lain sebagainya.

Stored function dapat menerima parameter yang akan dilewatkan melalui *stored function*. Semua parameter di dalam *stored function* dianggap sebagai parameter "IN", bukan parameter "OUT" atau "INOUT". Selain itu, yang harus dipertimbangkan dalam implementasi *stored function* adalah deklarasi tipe data nilai kembalian.

1. Sintaks *Stored Function*

Pembuatan *stored function* dapat dilakukan menggunakan perintah "CREATE FUNCTION" atau menggunakan perintah "CREATE OR REPLACE FUNCTION", jika *stored function* sudah ada sebelumnya. Struktur sintak *stored function* ditunjukkan pada gambar 6.1.

```
CREATE FUNCTION function_name ([parameter[,...])
       RETURNS datatype
    [LANGUAGE SQL]
    [ [NOT] DETERMINISTIC ]
    [ { CONTAINS SQL|NO SQL|MODIFIES SQL DATA|READS SQL DATA} ]
    [SQL SECURITY {DEFINER|INVOKER} ]
    [COMMENT comment_string]
    function_statements
```

Gambar 6.1. Struktur Sintaks *Stored Function*

Berdasarkan gambar 6.1., ditunjukkan pembuatan *stored function* menggunakan stateman "CREATE FUNCTION" dan diikuti klausa "RETURN", dimana harus ditentukan tipe data nilai kembaliannya. Klausa "RETURN" dan tipe data nilai kembalian ini harus ada di dalam pendeklarasian *stored function* (Harrison dan Feuerstein 2006). *Stored function* juga dapat melewatkan parameter, akan tetapi berbeda dengan *stored procedure*, parameter di dalam *stored function* adalah parameter masukan atau "IN". Selanjutnya di bagian *body function* harus terdapat statemen "RETURN".

Sebagai contoh, *stored function* berikut digunakan untuk menampilkan jumlah mahasiswa berdasarkan kode mata kuliah yang dimasukkan. Sintak *stored procedure* contoh di atas ditunjukkan pada gambar 6.2.

```
DELIMITER $$
CREATE FUNCTION sf_tampil_mahasiswa_mk (kd_mk VARCHAR(10))
RETURNS INT

BEGIN
        DECLARE jml INT;
        SELECT COUNT(*) AS jml_mhs INTO jml FROM krs WHERE Kode_MK = kd_mk;
RETURN jml;
END$$
```

Gambar 6.2. Contoh *Stored Function*

Berdasarkan gambar 6.2, ditunjukkan sebelum deklarasi pembuatan *stored* function, terlebih dahulu didefinisikan *delimiter* yang digunakan untuk mengatasi permasalah penggunaan tanda titik koma (;) di awal dan akhir blok sintaks *stored function*, yaitu "$$". Selanjutnya, sintaks *stored function* dimulai dengan perintah "CREATE FUNCTION" untuk memulai perintah pembuatan *function*. Diikuti parameter dan tipe datanya yang akan dilewatkan ke dalam *stored function* serta tipe data nilai yang dikembalikan dari *stored function*. Kode yang dieksekusi oleh *stored function* dituliskan pada blok "***BEGIN..END***". Penjelasan alur *stored function* pada gambar 6.2 adalah sebagai berikut :

a. CREATE FUNCTION sf_tampil_mahasiswa_mk. Digunakan untuk membuat *stored function* dengan nama "sf_tampil_mahasiswa_mk".
b. kd_mk VARCHAR (10). Digunakan untuk mendeklarasikan parameter masukan yang dilewatkan di dalam *stored function* dengan nama "kd_mk" bertipe VARCHAR dengan nilai maksimal adalah 10 (sepuluh) karakter, parameter ini digunakan untuk menyimpan kode mata kuliah,
c. RETURN INT digunakan untuk mendeklarasikan tipe nilai kembalian berupa *integer*.
d. DECLARE jml INT; digunakan untuk mendeklarasikan variabel "jml" bertipe *integer* untuk menampung nilai kembalian yang dihasilkan oleh sintaks pada *body function*.
e. Sintaks pada bagian *body function* menghitung jumlah baris berdasarkan jumlah mahasiswa yang mengikuti suatu perkuliahan melalui klausa "*COUNT*".
f. INTO jml. Digunakan untuk menyimpan hasil perhitungan dalam variabel "jml".
g. RETURN jml. Digunakan untuk menampilkan nilai kembalian *stored function*.

Seperti halnya *stored procedure*, di dalam *stored function* juga dapat diterapkan pembatasan untuk menentukan pengguna yang dapat mengeksekusi *stored function*. Pembatasan tersebut diaplikasi pada bagian "*SQL SECURITY*" dengan memilih opsi "*DEFINER*" atau "*INVOKER*". Pembahasan mengenai "*DEFINER*" atau "*INVOKER*" dapat dibaca pada BAB V "Stored Procedure". Sebagai contoh, *stored function* berikut digunakan untuk menampilkan nama mahasiswa berdasarkan NIM yang dimasukkan pada parameter. Sintak *stored function* ditunjukkan pada gambar 6.3.

```
DELIMITER $$
USE db_akademik$$
CREATE DEFINER=root@localhost FUNCTION TAMPILNAMA(NIM VARCHAR(9)) RETURNS varchar(25)
BEGIN
     DECLARE NAMA VARCHAR(25);
     SELECT Nama_Mhs FROM mahasiswa WHERE Nim = NIM LIMIT 1 INTO NAMA;
     RETURN NAMA;
END$$
```

Gambar 6.3. *Stored Function* dengan Pembatasan Akses

Sesuai sintaks pada gambar 6.3, *stored function* yang dibuat menerapkan pembatasan akses menggunakan perintah "DEFINER=root@localhost". Artinya *stored function* tersebut hanya dapat dieksekusi oleh pengguna *root* pada MySQL.

2. Eksekusi *Stored Function*

Cara eksekusi *stored function* berbeda dengan *stored procedure. Stored function* dapat dieksekusi menggunakan perintah "SELECT" diikuti nilai parameter yang akan dilewatkan (jika ada). Sebagai contoh, untuk mengeksekusi *stored function* pada gambar 6.2 digunakan perintah seperti ditunjukkan pada gambar 6.4 berikut

```
SELECT mk.Nama_MK, sf_tampil_mahasiswa_mk (130342218) as jumlah_mahasiswa
FROM matakuliah as mk
WHERE mk.Kode_MK='130342218'
```

Gambar 6.4. Eksekusi *Stored Function*

Parameter yang dilewatkan di *stored* function adalah kode mata kuliah "1303042218". Hasil eksekusi *stored function* menampilkan jumlah mahasiswa berdasarkan kode mata kuliah yang dimasukkan ditunjukkan pada gambar 6.5.

Nama_MK	jumlah_mahasiswa
Sistem Cerdas	3

Gambar 6.5. Hasil Eksekusi *Stored Function*

3. Perubahan dan Penghapusan *Stored Function*

Struktur atau karakteristik *stored function* dapar diubah menggunakan perintah "ALTER FUNCTION nama_*stored_function*" (DuBois et.al 2005). Sebagai contoh, untuk mengubah hak akses *stored function* dengan menambahkan klausa "*SQL SECURITY INVOKER / DEFINER*" dan klausa "*COMMENT*". Contoh sintaks untuk mengubah karakteristik atau struktur *stored function* ditunjukkan pada gambar 6.6.

```
ALTER FUNCTION f
  SQL SECURITY INVOKER
  COMMENT 'this function has invoker security';
```

Gambar 6.6. Mengubah Karakteristik *Stored Function*

Sintaks pada gambar 6.6 digunakan untuk mengubah karakteristik *stored function*, yaitu menambahkan *SQL SECURITY INVOKER* dan komentar dengan nilai "*this function has invoker security*". Sama halnya dengan *stored procedure*, bagian *routine_body* pada sebuah *stored function* tidak dapat dimodifikasi menggunakan perintah ALTER FUNCTION. Untuk mengubah *routine_body* pada sebuah *stored function* digunakan perintah "DROP procedure IF EXIST", kemudian deklarasikan kembali perintah pembuatan *stored function* (Kruckenberg dan Pipes 2005).

4. Menggunakan Parameter pada *Stored Function*

Seperti sudah dibahas pada bagian sebelumnya, *stored function* hanya dapat memiliki parameter masukan yang dituliskan di dalam pasangan tanda "(" dan ")". Parameter masukan ini nantinya akan diproses oleh *stored function* dan hasilnya disimpan sebagai nilai kembalian suatu *stored function*. Sebagai contoh, *stored function* untuk menghasilkan NIM mahasiswa baru berdasarkan tahun masuk dan urutan NIM-nya. Sintaks *stored function* ditunjukkan pada gambar 6.7.

```
DELIMITER $$
CREATE FUNCTION NIM_Baru(TAHUN VARCHAR(4))
RETURNS VARCHAR(12)
BEGIN
     DECLARE kodebaru CHAR(12);
                DECLARE urut INT;
                DECLARE urut_baru INT;
                DECLARE angkatan CHAR(2);
                DECLARE jumlah INT;

                SELECT SUBSTR(TAHUN,3,2) INTO angkatan;
                IF (angkatan ='20') THEN
                BEGIN
                              SELECT MAX(SUBSTR(Nim,LOCATE("0",Nim,6),3)) INTO urut
                              FROM mahasiswa WHERE NIM LIKE "20%";

                              SET urut_baru = urut+1;
                              SET kodebaru = CONCAT('200304',LPAD(urut_baru,3,'0'));
                END;
                ELSEIF (angkatan ='21') THEN
                BEGIN
                              SELECT COUNT(Nim) INTO jumlah FROM mahasiswa WHERE NIM LIKE "21%";
                              IF (jumlah=0) THEN
                              BEGIN
                                            SET urut_baru = 1;
                                            SET kodebaru = CONCAT('210304',LPAD(urut_baru,3,'0'));
                              END;
                              ELSEIF (jumlah >1) THEN
                              BEGIN
                                            SELECT MAX(SUBSTR(Nim,LOCATE("0",Nim,6),3)) INTO urut
                                            FROM mahasiswa WHERE NIM LIKE "21%";
                                            SET urut_baru = urut+1;
                                            SET kodebaru = CONCAT('210304',LPAD(urut_baru,3,'0'));
                              END;
                              END IF;
                END;
                END IF;
                RETURN kodebaru;
END$$
```

Gambar 6.7. Sintaks *Stored Function* NIM Mahasiswa

Sintak *stored function* pada gambar 6.7 memiliki parameter masukan dengan nama "TAHUN", tipe data pada parameter masukan tersebut adalah VARCHAR dengan jumlah maksimum karakter adalah 4 (empat). Parameter "TAHUN" digunakan untuk menyimpan nilai masukan, yaitu tahun. Berdasarkan nilai masukan tersebut akan dihasilkan NIM baru sesuai urutan NIM yang sudah ada. *Stored function* tersebut dipanggil menggunakan perintah "*SELECT NIM_Baru_tahun('2020') as NIM_Baru*" dengan nilai masukan adalah "2020". Hasil eksekusi *stored function* pada gambar 6.7 ditunjukkan pada gambar 6.8.

NIM_Baru
200304004

Gambar 6.8. Hasil Pembuatan NIM Menggunakan *Stored Function*

Pemanggilan *stored function* juga dapat digabungkan dengan perintah *query* lainnya, misalkan dengan perintah *"INSERT", "SELECT", "UPDATE"* atau *"DELETE"*. Sebagai contoh, untuk menambahkan data mahasiswa baru dengan NIM dihasilkan dari hasil eksekusi *stored function* pada gambar 6.7. Sintaks perintah *query*

dengan pemanggilan *stored function* ditunjukkan pada gambar 6.9. Sedangkan hasil eksekusi perintah *query* ditunjukkan pada gambar 6.10.

```
INSERT INTO mahasiswa (Nim,Nama_Mhs,Tgl_Lahir,Alamat,Jenis_Kelamin)
VALUES (NIM_Baru_tahun('2020'),'Anto Kurniawan','1995-08-02',
'JL. MASJID AL KAUSAR RT. 3 RW. 5 DONDONG KESUGIHAN CILACAP 53274',
'Laki-laki')
```

Gambar 6.9. Perintah *Query* dengan Pemanggilan *Stored Function*

190304001	Susi Susanti	1998-09-01	Banteran Sumbang Banyumas	Perempuan
200304001	Riza Habibi	1999-10-02	Purbalingga	Laki-laki
200304002	Anang Kukuh Adi Susilo	1999-08-03	Banyumas	Laki-laki
200304003	Hendrik Prasetyo	1999-06-10	Cilacap	Laki-laki
200304004	Anto Kurniawan	1995-08-02	JL. MASJID AL KAUSAR RT. 3 RW. 5 DONDONG KESUGIHAN CILACAP 5	Laki-laki

Gambar 6.10. Hasil Eksekusi *Query* dengan Pemanggilan *Stored Function*

5. Mengembalikan nilai pada *Stored Function*

Sebelum mengembalikan nilai pada *stored function*, terlebih dahulu harus didefinisikan tipe data kembaliannya menggunakan perintah "*RETURN <Tipe_Data>*". Kemudian perlu ditetapkan pula deklarasi variabel local untuk menyimpan nilai kembalian dari *stored function*. Variabel lokal dapat dideklarasikan menggunakan perintah "*DECLARE <nama_variabel> <tipe_data>;*". Sebagai contoh "`DECLARE kodebaru CHAR(12);`".

Langkah berikutnya adalah menyimpan nilai yang dihasilkan oleh perintah di dalam *stored function* ke dalam variabel lokal. Penyimpanan ke variabel lokal dapat menggunakan perintah "`SET <nama_variabel_lokal> = <nilai_yang_dihasilkan>;`". Sebagai contoh "*SET kodebaru = CONCAT('200304',LPAD(urut_baru,3,'0'));*". Atau menggunakan perintah "*INTO <nama_variabel_lokal>*", sebagai contoh "*SELECT COUNT(*) AS jml_mhs INTO jml FROM krs WHERE Kode_MK = kd_mk;*".

Setelah nilai yang dihasilkan oleh *stored function* disimpan di dalam variabel lokal, maka nilai tersebut dapat diatur sebagai nilai kembalian *stored function* menggunakan perintah "*RETURN <nama_variabel_lokal>;*". Sebagai contoh "*RETURN kodebaru;*".

C. Rangkuman

1. *Stored function* merupakan jenis *stored program* di dalam *database* yang digunakan untuk menangani perhitungan yang kompleks dan tidak dapat diselesaikan oleh perintah SQL serta di dalam perhitungan membutuhkan pernyataan kondisi dan logika perulangan.
2. *Stored function* dapat dipanggil di dalam perintah *query* dengan perintah DML, seperti seperti "SELECT", "INSERT", "UPDATE", dan "DELETE".
3. Perbedaan *stored function* dengan *stored procedure* yaitu : 1) *stored function* memiliki nilai kembalian, sedangkan *stored procedure* tidak memiliki nilai kembalian, 2) *stored function* hanya dapat memiliki parameter masukan, sedangkan *stored procedure* dapat memiliki parameter masukan dan luaran.

D. Tugas/Latihan

1. Buatlah *stored function* untuk menampilkan jumlah mahasiswa yang mengikuti mata kuliah tertentu berdasarkan nama mata kuliahnya !
2. Buatlah *stored function* untuk menghitung nilai Indeks Prestasi Semester (IPS) seorang mahasiswa (input parameternya adalah NIM) !
3. Pemanggilan *stored function* pada *query.* Buatlah *query* untuk menampilkan NIM, Nama Mahasiswa dan IPS-nya, nilai didapatkan pemanggilan stored function pada nomor 2 !
4. Buatlah *stored function* untuk penomoran NIM Mahasiswa sesuai tahun masuknya dengan format sebagai contoh, jika tahun 2020, maka format NIM adalah "200304" + nomor_urut, sedangkan apabila tahun 2021, maka format NIM adalah "210304"+ nomor_urut dan seterusnya.
    a. Gunakan fungsi *substr* untuk mengekstrak nomor urut dari data NIM yang sudah ada di tabel mahasiswa
    b. Gunakan fungsi *MAX* untuk mendapatkan nomor NIM terbesar dari hasil fungsi *substr*
    c. NIM baru dapat dibuat menggunakan formula dan function berikut : `CONCAT(angkatan,'0304',LPAD(URUT_BARU,3,0);`

d. Lebih lanjut tentang fungsi LPAD, dapat dipelajari pada tautan berikut :*https://www.w3schools.com/sql/func_mysql_lpad.asp* dan *https://www.w3resource.com/mysql/string-functions/mysql-lpad-function.php*

5. Diskusikan, apakah *stored function* dapat mengembalikan nilai lebih dari satu nilai ? Berikan contoh sintaks *stored function* untuk mengembalikan nilai lebih dari satu nilai !

E. Daftar Pustaka

1. DuBois, Paul., 2014, *MySQL CookBook 3rd edition,* USA : O'Reilly Media.
2. DuBois, P., Hinz, S., Pedersen, C., 2005, MySQL 5.0 Certification Study Guide, USA : MySQL Press.
3. Kruckenberg, M., dan Pipes, J., 2005, Pro MySQL, USA : Apress.