MATERI V
STORED PROCEDURE

A. Tujuan

B. Dasar Teori

Eksekusi perintah SQL terkadang dilakukan secara berulang kali untuk memenuhi sebuah *task*. Sebagai contoh, ketika seorang pegawai baru bergabung di sebuah instansi, maka data pegawai tersebut harus dimasukkan ke dalam basis data. Data detil pegawai akan disimpan di beberapa table yang sesuai. Oleh karena itu, diperlukan beberapa kali eksekusi untuk perintah *insert* ke tiap-tiap tabel terkait. Pada skenario ini, beberapa perintah SQL dieksekusi secara bersamaan ke dalam SQL Server sebagai satu unit. Hal ini akan membantu mengurangi beban pada jalur komunikasi jaringan.

MySql menyediakan fitur untuk menangani eksekusi perintah SQL secara berulang melalui *Stored Procedure* dan *Function.* Perbedaan utama antara *function* dan *Stored Procedure* adalah terletak pada nilai yang dikembalikannya (*return value*).

Stored procedure adalah sebuah objek terkompilasi yang disimpan di dalam basis data. Di dalamnya dapat melibatkan perintah DDL dan DML. Umumnya suatu *Stored Procedure* hanya berisi suatu kumpulan proses yang tidak menghasilnya value, biasanya hanya menampilkan saja. Apabila terdapat nilai yang akan diberikan pada variabel yang dideklarasikan di dalam procedure pada saat *run time*, dapat dilakukan dengan melewatkan parameter ketika eksekusi *stored procedure* tersebut. Selain itu, sebuah *stored procedure* juga dapat memanggil *stored procedure* lainnya. Selain itu, melalui *Stored procedure*, SQL tidak akan melakukan loading seluruh tabel yang ter-relasi, tetapi langsung melakukan filtering berdasarkan query yang dimaksud sehingga dari sisi performa eksekusi, utilitas jaringan, dan keamanan dapat lebih terjaga.

1. Sintaks *Stored Procedure*

Stored procedure dibuat menggunakan perintah *CREATE PROCEDURE*. Format sintaks pembuatan *stored procedure* adalah sebagai berikut :

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

Keterangan :

- sp_name : nama *stored procedure*.
- proc_parameter : parameter input / output dari *stored procedure* tersebut (opsional).
- characteristic : menjelaskan karakteristik dari stored procedure (COMMENT, LANGUAGE SQL, dan lain-lain).
- routine_body : kumpulan perintah pada *stored procedure* tersebut.
- Jika DEFINER dispesifikasikan maka kita memutuskan *stored procedure* tersebut dijalankan hanya oleh user tertentu (dalam format penulisan user@host). Jika tidak dispesifikasikan, maka user yang melakukan perubahan (CURRENT_USER) adalah pilihan default.

Sebagai contoh, akan dibuat *stored procedure* untuk menampilkan NIP, Nama Dosen beserta Kode Mata Kuliah dan Nama Mata Kuliah yang diampu. Sintaks *stored procedure* untuk menangani kasus di atas ditunjukkan pada sintaks SQL berikut :

```
DELIMITER  //
CREATE PROCEDURE sp_dosen_mk()
BEGIN
        SELECT j.Nip,d.Nama_Dosen,j.Kode_MK,mk.Nama_MK
        FROM jadwal j
        JOIN dosen d ON j.Nip = d.Nip
        JOIN matakuliah mk ON j.Kode_MK=mk.Kode_MK;
END//
```

2. Eksekusi Stored Procedure

Eksekusi *Stored Procedure* dapat dilakukan dengan menggunakan perintah "CALL nama_stored_procedure (parameter *input* (*jika ada*))". Sebagai contoh, untuk mengesekusi *stored procedure* yang dibuat sebelumnya, yaitu "sp_dosen_mk", maka sintaks eksekusi *stored procedure*-nya adalah sebagai berikut : "CALL sp_dosen_mk()". Pada bagian parameter dikosongkan karena *stored procedure* yang dibuat sebelumnya tidak memerlukan parameter masukan atau luaran. Hasil eksekusi *stored procedure* ditunjukkan pada gambar berikut :

Nip	Nama_Dosen	Kode_MK	Nama_MK
2160898	Noven Indra, S.Kom.,M.Kom	130342217	Struktur Data dan Algoritma
2160887	Khatibul Umam, S.T.,M.Kom.	130342218	Sistem Cerdas
2160898	Noven Indra, S.Kom.,M.Kom	1403040202	Basis Data Lanjut
2160422	Bangun Wijayanto, S.T.,M.Cs.	1403042108	Teori Komputasi
2160577	Anton Sanjaya, S.Kom.,M.Kom.	1403042110	Manajemen Proyek
2160897	Irsyad Rizki, S.Kom.,M.Kom.	1403042111	Sistem Operasi
2160887	Khatibul Umam, S.T.,M.Kom.	1403043119	Komputasi Lunak
2160588	Teguh Wahyono, S.T.,M.T.	1403043122	Kecerdasan Bisnis

3. Perubahan dan Penghapusan Stored Precodure

Bagian *routine_body* pada sebuah *stored procedure* tidak dapat dimodifikasi menggunakan perintah ALTER PROCEDURE. Untuk mengubah *routine_body* pada sebuah *stored procedure* digunakan perintah "DROP procedure IF EXIST", kemudian deklarasikan kembali perintah pembuatan *stored procedure.* Sebagai contoh untuk merubah *stored procedure* "sp_dosen_mk ". Sintaks untuk merubah *stored procedure* "sp_dosen_mk" adalah sebagai berikut :

```
DELIMITER //
DROP PROCEDURE IF EXISTS `db_akademik`.`sp_dosen_mk`;

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_dosen_mk`()
BEGIN
        SELECT j.Nip,d.Nama_Dosen,j.Kode_MK,mk.Nama_MK,J.Hari
        FROM jadwal j
        JOIN dosen d ON j.Nip = d.Nip
        JOIN matakuliah mk ON j.Kode_MK=mk.Kode_MK;
END//
```

4. Menggunakan Parameter pada *Stored Procedure*

Stored procedure dapat berisi parameter, yaitu nilai yang akan dilewatkan pada stored procedure secara run time saat stored procedure tersebut dieksekusi. Setiap parameter memiliki nama, tipe data, variabel dimana nilai tersebut akan disimpan, dan nilai awal. Sebagai contoh *stored procedure* berikut untuk menampilkan mata kuliah beserta dosen pengampu pada hari tertentu. Sintak *stored procedure* tersebut ditunjukkan pada sintaks berikut :

```
DELIMITER//
CREATE PROCEDURE sp_mk_hari
(
        IN hari_kuliah VARCHAR(12)
)
BEGIN
        SELECT j.Nip,d.Nama_Dosen,j.Kode_MK,mk.Nama_MK,j.Hari
        FROM jadwal j
        JOIN dosen d ON j.Nip = d.Nip
        JOIN matakuliah mk ON j.Kode_MK=mk.Kode_MK
        WHERE j.Hari=hari_kuliah;
END//
```

Eksekusi *stored procedure* dengan parameter masukan dapat dilakukan dengan sintaks berikut ;

```
CALL sp_mk_hari('Selasa');
```

Parameter masukan pada *stored procedure* dapat digunakan untuk menambahkan data, mengubah, atau menghapus data pada suatua tabel. Sebagai contoh *stored procedure* berikut digunakan untuk menambahkan record baru di tabel dosen.

```
DELIMITER //
CREATE PROCEDURE tambah_dosen (
        IN NIP_Dosen VARCHAR(12),
        IN NAMA_DOSEN VARCHAR(50)
)

BEGIN
        INSERT INTO dosen (Nip,Nama_Dosen)
        VALUES (NIP_Dosen,NAMA_DOSEN);
END//
```

Penggunaan *stored procedure* untuk menambahkan data pada tabel dosen dapat dilakukan dengan memanggil *stored procedure* beserta parameter masukkannya, yaitu NIP,dan NAMA_DOSEN. Sintaks pemanggailan *stored procedure* ditunjukkan sebagai berikut :

```
call tambah_dosen ('2160587','Suyanto, S.T.,M.T.')
```

5. Mengembalikan nilai pada *Stored Procedure*

Sama halnya dengan pemberian nilai masukan pada *stored procedure* saat *run time*, *stored procedure* juga dapat digunakan untuk mengembalikan nilai sebagai luaran dari *stored procedure*. Nilai dapat dikembalikan melalui parameter OUT. Untuk menentukan parameter sebagai sebuah luaran, dapat digunakan keyword OUT pada bagian deklarasi parameter. Sebagai contoh, *stored procedure* berikut digunakan untuk menghitung jumlah mata kuliah yang diikuti tiap mahasiswa. Jumlah mata kuliah yang

diikuti merupakan parameter luaran. S*tored procedure* tersebut ditunjukkan pada sintaks berikut :

```
DELIMITER//
CREATE PROCEDURE sp_mahasiswa_jml_mk
(
                IN NIM_mhs VARCHAR (12),
                OUT NIM VARCHAR (12),
                OUT NAMA VARCHAR (25),
                OUT jumlah_mk SMALLINT
)
BEGIN
                SELECT k.NIM,m.Nama_Mhs,COUNT(k.Kode_MK) INTO NIM,NAMA,jumlah_mk
                FROM krs k
                JOIN mahasiswa m ON k.NIM=m.Nim
                WHERE k.NIM=NIM_mhs;
END//
```

Pemanggilan *stored procedure* yang menggunakan OUT parameter dapat dilakukan menggunakan sintaks berikut :

```
CALL sp_mahasiswa_jml_mk('100304001',@NIM,@NAMA,@jumlah_mk);
SELECT @NIM as NIM,@NAMA as Nama,@jumlah_mk AS jumlah_MK;
```

Hasil eksekusi pemanggilan *stored procedure* tersebut ditunjukkan pada gambar berikut :

NIM	Nama	jumlah_MK
100304001	DIANA BUDI	5

C. Evaluasi dan Pertanyaan

1. Buatlah *stored procedure* dengan menggunakan *subquery* untuk menambahkan ecord pada table jadwal dengan nama dosen 'Irsyad Rizki, S.Kom.,M.Kom.', nama mata kuliah 'Manajemen Proyek', Hari 'Selasa', Ruang "Lab. RPL", dan Jam "13.00-15.00" !
2. Buatlah *stored procedure* untuk menampilkan NIM, nama mahasiswa, dan IP Semester tiap tiap mahasiswa !
3. Buatlah *stored procedure* untuk menampilkan kode mata kuliah, nama mata kuliah, nama dosen pengampu, dan jumlah mahasiswa tiap yang mengikuti tiap mata kuliah !
4. Buatlah *stored procedure* untuk mengubah (*update*) tabel jadwal untuk kode MK =”1403043119”dengan ketentuan sebagai berikut :
   - Pengampu dengan Nip =”2160887” menjadi “2160587”
   - Hari =”Rabu” menjadi “Jum’at”
   - Ruang =”Lab. Cerdas” menjadi “Lab. RPL”