MODUL VII
STORED PROCEDURE

A. TUJUAN

- Memahami konsep dasar stored procedure, kelebihan dan kekuranganya.
- Memahami implementasi stored procedure di dalam basis data.
- Mampu menyelesaikan operasi – operasi data spesifik dengan memanfaatkan stored procedure

B. PETUNJUK

- Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan – latihan praktikum dengan baik dan benar.
- Kerjakan tugas – tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal – hal yang kurang jelas.

C. DASAR TEORI

1. Stored Procedure

Stored Procedure adalah sebuah prosedur layaknya subprogram (subrutin) di dalam bahasa pemrograman reguler yang tersimpan di dalam katalog basis data. Beberapa kelebihan yang ditawarkan stored procedure antara lain : mengingkatakan performa, mereduksi trafik jaringan, reusable, dan meningkatkan kontrol sekuriti. Di balik kelebihan tersebut, stored procedure juga memiliki kekurangan. Di antaranya adalah berpotensi meningkatkan beban server dan penulisnaya tidak mudah (memerlukan pengetahuan yang spesifik).

Contoh sintaks stored procedure :

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

Untuk memanggil sotred preocedure, digunakan perintah CALL (beberapa DBMS ada yang menggunakan EXECUTE).

```
CALL sp_name
```

Dalam Implementasinya, penggunaan stored procedure sering melibatkan parameter. Di MySQL, parameter stored procedure dibedakan menjadi tiga mode : IN, OUT, dan INOUT.

A. **IN**

   Parameter yang merupakan mode default ini mengindikasikan bahwa sebuah parameter dapat di-*pass* ke dalam stored procedure tetapi nilainya tidak dapat diubah (dari dalam stored procedure)

B. OUT

   Mode ini mengindikasikan bahwa stored procedure dapat mengubah parameter dan mengirimkan kembali ke program pemanggil

C. INOUT

   Mode ini pada dasarnya merupakan kombinasi dari mode IN dan OUT.
   Sintaks pendefinisan parameter diperlihatkan sebagai berikut :

```
MODE param name param type(param size)
```

Stored procedure dapat mencerminkan beragam operasi data, misalnya seleksi, penambahan, pengubahan, penghapusan, dan juga operasi – oprasi DDL.

Seperti halnya procedure di dalam bahasa pemrograman, stored procedure juga dapat melibatkan variabel, pernyataan kondisional, dan pengulangan.

D. LATIHAN

1. Stored Procedure

Seperti halnya tabel, stored procedure diciptakan dengan menggunakan perintah CREATE sebagai contoh, buat stored procedure getMahasiswa() untuk menampilkan semua data mahasiswa.

1. Ketikkan pernyataan pembuatan stored procedure berikut :

```
DELIMITER //
CREATE PROCEDURE getMhs()
BEGIN
    SELECT * FROM mahasiswa;
END //
DELMITER;
```

Perintah DELIMITER digunakan untuk mengubah delimiter standar, misalnya di sini dari titik koma **(;)** menjadi slash ganda **(//).** Langkah ini umumnya dilakukan ketika isi stored procedure mengandung titik komayang merupakan delimiter standar di SQL.
Pernyataan di antara BEGIN dan END merupakan badan (*body)* stored procedure.
Perintah DELIMITER di akhir baris digunakan untuk mengembalikan delimiter ke karakter semula.

2. Eksekusi Query tersebut dengan memanggil procedure getMahasiswa().

```
CALL getMhs();
```

nim	nama	Jenis_Kelamin	alamat
101	Arif	L	Jl. Kenangan
102	Budi	L	Jl. Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Sari	P	Jl. Malang

2. Parameter IN

Stored procedure di contoh sebelumnya memperlihatkan bentuk default (tanpa parameter). Di sini kita juga bisa mendefinisikan parameter yang nantinya dapat digunakan olehh pernyataan di body stored procedure.

Sebagai contoh, kita bisa mendapatkan semua data matakuliah di semester tertentu.

```
DELIMITER //
CREATE PROCEDURE getMkBySemester(IN smt INT(2))
BEGIN
    SELECT * FROM matakuliah
    WHERE semester = smt;
END //
DELMITER;
```

Untuk memanggil stored procedure yang memiliki parameter, maka kita harus menspesifikasikan argumenya. Misalkan kita ingin mendapatkan data matakuliah di semester 3.

```
CALL getMkBySemester(3);
```

kd_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
PTI777	Sistem Informasi	2	3	99
TIK342	Praktikum Basis Data	1	3	11

Apabila pemanggilan stored procedure di atas mengabaikan argumen, DBMS akan merespon dengan pesan kesalahan.

Bergantung kebutuhan, pendefinisian parameter pada stored procedure juga bisa lebih dari satu. Sebagai contoh, buat stored procedure dengan dua buah parameter seperti berikut :

```
DELIMITER //
CREATE PROCEDURE getMkBySemSks(
    IN Smt INT(2),
    IN Sks INT(2))
BEGIN
    SELECT * FROM matakuliah
    WHERE semester = Smt
    AND sks = Sks ;
END //
DELMITER;
```

Pemanggilan stored procedure di atas tentunya akan memerlukan dua buah argumen.

```
CALL getMkBySemSks(3,2);
```

kd_mk	nama_mk	sks	semester	kode_dos
PTI777	Sistem Informasi	2	3	99

Variabel

Di MySQL, kita juga bisa mendeklarasikan variabel global – ruang lingkup session – dengan menggunakan perintah SET dan notasi @. Sebagai contoh, perintah berikut akan mendeklarasikan variabel bernama **smt** dan diinisialisasi dengan nilai **3**. Dan untuk memeriksa nilai variabel, gunakan perintah select

```
SET @smt = 3;
SELECT @smt;
```

@smt
3

Langkah selanjutnya, kita bisa memanfaatkan variabel yang telah dideklarasikan untuk operasi – operasi lain, misalnya sebagai argumen stored procedure.

```
CALL getMHSBySemester(@smt);
```

kd_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
PTI777	Sistem Informasi	2	3	99
TIK342	Praktikum Basis Data	1	3	11

Penambahan Data

Pada operasi penambahan, data – data terkait diisikan melaui argumen. Selanjutnya, isi stored procedure akan memasukkan data ke dalam tabel.

Berikut adalah contoh stored procedure untuk menambahkan data pada tabel dosen

```
DELIMITER //
CREATE PROCEDURE AddDosen(
    IN kode_dos VARCHAR(10),
    IN nama_dos VARCHAR(50),
    IN alamat_dos VARCHAR(100)
)
BEGIN
    INSERT INTO dosen VALUES(
        kode_dos,nama_dos,alamat_dos);
END //
DELMITER;
```

Lakukan eksekusi terhadap procedure tersebut

```
call AddDosen('212','Gunawan','Jl. Ambarawa');
```

Selanjutnya lakukan pengecekan data pada tabel dosen.

```
select * from dosen;
```

kode_dos	nama_dos	alamat_dos
10	Suharto	Jl. Jombang
11	Martono	Jl. Kalpataru
12	Rahmawati	Jl. Jakarta
13	Bambang	Jl. Bandung
14	Nurul	Jl. Raya Tidar
212	Gunawan	Jl. Ambarawa

Operasi – operasi manipulasi data lainya bisa anda coba sendiri, dan tidak jauh berbeda dengan pernyataan SQL reguler

3. Parameter OUT

Dalam konteks bahasa pemrograman, parameter OUT analog dengan *passing-by-reference*. Dengan demikian, parameter ini nilainya bisa diubah oleh stored procedure.

```
DELIMITER //
CREATE PROCEDURE JumlahDosen(
    OUT jumlah_dos INT(3)
)
BEGIN
    SELECT COUNT(kode_dos)
    INTO jumlah_dos FROM dosen;
END //
DELMITER;
```

Untuk mengeksekusi stored procedure dengan parameter OUT, dibutuhkan argumen yang spesifik.

```
call JumlahDosen(@jumlah_dosen);
```

Perhatikan, argumen harus menggunakan notasi @, yang mengindikasikan sebagai suatu parameter OUT.

Langkah selanjutnya, untuk mendapatkan nilai variabel, gunakan pernyataan

SELECT

```
select @jumlah_dosen;
```

@jumlah_dosen
6

Parameter mode OUT juga bisa dikombinasikan dengan mode IN.

4. Parameter INOUT

Pada parameter dengan mode INOUT ini, kita bisa mengirimkan parameter kedalam stored procedure dan mendapatkan nilai kembalian yang baru dari stored procedure yang didefinisikan.

Sebagai contoh, definisikan stored procedure seperti berikut :

```
DELIMITER //
CREATE PROCEDURE CountBySks(
    INOUT var INT(3)
)
BEGIN
    SELECT COUNT(kd_mk)
    INTO var
    FROM matakuliah
    WHERE sks=var;
END //
DELMITER;
```

Lakukan eksekusi pada procedure tersebut untuk mencari jumlah matakuliah yang memiliki sks = 2. Dengan mendeklarasikan variabel @sks dengan nilai 2 terlebih dahulu.

```
set @sks=2;
```

@sks
2

```
call CountBySks(@sks);
```

Lakukan pengecekan pada variabel sks setelah dilakukan eksekusi pada stored procedure tersebut.

```
select @sks;
```

@sks
2

Pendekatan INOUT juga dapat direpresentasikan dengan memisah parameter IN dan OUT

Contoh penggunaanya, misal untuk mendapatkan jumlah mahasiswa yang jenis kelaminya adalah L.

```
DELIMITER //
CREATE PROCEDURE CountByGender(
    IN gender VARCHAR(3),
    OUT total INT(3)
)
BEGIN
    SELECT COUNT(nim)
    INTO total
    FROM mahasiswa
    WHERE Jenis_Kelamin=gender;
END //
DELMITER;
```

Kemudian lakukan eksekusi pada procedure tersebut.

```
CALL CountByGender('L',@total);
select @total;
```

Maka akan didapat jumlah dari mahasiswa yang berjenis kelamin L adalah :

@total
4

5. Pencabangan dan Pengulangan

Penggunaan pernyataan – pernyataan percabangan ataupun pengulangan dapat dilakukan di dalam stored procedure.Sehingga dapat digunakan untuk menghasilkan suatu procedure yang lebih kompleks.

Contoh berikut merupakan penggunaan dari pernyataan IF di dalam stored procedure.

```
DELIMITER //
CREATE PROCEDURE DemoIF(
    IN bil INT(3)
)
BEGIN
    DECLARE str VARCHAR(50);
    IF(bil<0) THEN
        SET str = 'BILANGAN NEGATIF';
    ELSE
        SET str = 'BILANGAN POSITIF';
    END IF;

    SELECT str;
END //
```

Di dalam procedure tersebut membutuhkan beberapa variabel tambahan seperti str, sehingga di dekalrasikan variabel str dengan tipe varchar. Variabel ini digunakan sebagai keluaran dari procedure teresebut.

Kemudian eksekusi stored procedure tersebut dengan memberikan parameter bilangan berapapun.

```
CALL DemoIF(7);
```

str
BILANGAN POSITIF

Contoh tersebut merupakan contoh untuk pernyataan kondisi didalam stored procedure. Dan berikut merupakan contoh penggunaan perulangan atau LOOPING pada stored procedure.

```
DELIMITER //
CREATE PROCEDURE DemoLOOP(
    IN bil INT(3)
)
BEGIN
    DECLARE str VARCHAR(50);
    DECLARE i INT(3);
    SET i=1;
    SET str='';
    WHILE i <= bil DO
        SET str=CONCAT(str,i,',');
        set i=i+1;
    END WHILE;

    SELECT str;
END //
DELMITER;
```

Lakukan eksekusi pada procedure tersebut.

```
CALL DemoLOOP(9);
```

str
1,2,3,4,5,6,7,8,9,

E. TUGAS PRAKTIKUM

1. Definisikan stored procedure untuk meng-*update* data pada tabel mahasiswa!
2. Definisikan stored procedure untuk mengetahui apakah nim sembarang mahasiswa sedang mengambil matakuliah atau tidak. Jika sedang mengambil, set status “ADA’, dan jika tidak mengambil, set status “TIDAK ADA”.
3. Definisikan stored procedure untuk menampilkan nilai genap saja, dengan inputan yang ditentukan!

F. TUGAS RUMAH

Tabel Buku

id_buku	id_penulis	nama_buku	genre	tahun
102	190	Spring In London	Romance	2007
111	129	Good Fifht	Action	2010
142	178	Kata Hati	Romance	2008
153	128	Marmut Merah Jambu	Comedy	2012
188	128	Koala Kumal	Comedy	2015
196	129	Pillow Talk	Romance	2008
322	190	In Blue Moon	Action	2007

Tabel Penulis

id_penulis	nama_penulis
128	Raditya Dika
129	Bernard Batubara
178	Christian simamora
190	Ilana Tan

Tabel detail_buku

id_buku	harga	stok
102	75000	20
142	92000	12
196	84000	27
111	63000	21
322	129000	7
153	89000	14
188	93000	26

1. Definisikan Store prosedure untuk menambahkan data pada tabel mahasiswa, jika data sudah ada, set status " Data Mahasiswa Sudah Terdaftar". dan jika data belum ada, set status "Data Mahasiswa berhasil ditambahkan" dan langsung ditampilkan!
2. Definisikan suatu Stored procedure yang berfungsi untuk menambahkan data pada tabel penulis, detail_buku, dan buku.
3. Definisikan Stored Procedure **INOUT** pilih salah satu dari tabel buku, penulis, atau detail_buku!