11

STORED PROCEDURE

STORED PROCEDURE

- Stored procedure adalah sekumpulan perintah SQL yang disusun dalam sebuah procedure (mirip dengan di pemrograman biasa) yang mempunyai nama dan kegunaan tertentu.
- Keuntungan penggunaan stored procedure
  - Mempaketkan sekumpulan perintah SQL dalam sebuah objek. Hal ini akan mengurangi traffik jaringan.
  - Kalau ada perubahan aturan bisnis, bisa dilakukan di server, sehingga akan mempengaruhi semua client.

VARIABLE

JENIS-JENIS VARIABLE

- Variable System
  - Dapat diakses dengan menggunakan @@namavar
  - Contoh : **`SELECT @@port`**
- Variable yang didefinisikan oleh user (User Defined Variable)
  - Dapat diakses dengan menggunakan @namavariable atau namavariable (tanpa @)
  - @namavar dibuat dengan menggunakan SET
  - Namavar (tanpa @) dapat dibuat dengan menggunakan DECLARE dalam sebuah statement BEGIN END.

VARIABLE SYSTEM

- Untuk melihat variable system dan nilainya, dapat menggunakan perintah

```
SHOW VARIABLES
```

- Melihat nilai sebuah variable

```
SELECT @@lc_time_names
```

- Mengubah isi sebuah variable (jika tidak readonly0

```
SET @@lc_time_names='id_ID';

-- TEST HASIL PERUBAHAN VARIABLE
SELECT DAYNAME(now()),MONTHNAME(now()); -- KAMIS, DESEMBER
```

USER DEFINED VARIABLE (dengan @)

- Variable ini hanya dikenal per koneksi. Jika anda membuat variable dalam sebuah koneksi database, maka koneksi lain (yang dilakukanoleh anda atau orang lain) tidak akan mengenal variable tersebut. (SESSION variable)
- Untuk membuat variable tersebut dapat menggunakan perintah SET.

```
SET @nama='Shelly';
SET @umur=20;
SELECT @nama,@umur;
SET @umur=@umur+1;
SELECT @nama,@umur;
```

```


Shelly, 20

Shelly, 21
```

USER DEFINED VARIABLE (tanpa @)

- Variable tanpa @ digunakan dalam stored procedure yang menggunakan perintah DECLARE dalam sebuah statement yang diapit dengan BEGIN dan END. Digunakan untuk membuat variable local dalam stored procedure.
- Lingkup variablenya hanya berlaku di dalam BEGIN END saja.

```
CREATE PROCEDURE test()
BEGIN
  DECLARE var1 INT DEFAULT 0;
  SET var1=var1+5;
  SELECT var1;
END
```

STORED PROCEDURE

- Stored procedure adalah sekumpulan perintah SQL yang disusun dalam sebuah procedure (mirip dengan di pemrograman biasa) yang mempunyai nama dan kegunaan tertentu.
- Keuntungan penggunaan stored procedure
  - Mempaketkan sekumpulan perintah SQL dalam sebuah objek. Hal ini akan mengurangi traffik jaringan.
  - Kalau ada perubahan aturan bisnis, bisa dilakukan di server, sehingga akan mempengaruhi semua client.

STORED PROCEDURE

- Struktur Pendeklarasian Stored Procedure adalah

```
CREATE PROCEDURE
  nama_sp ([[ IN | OUT | INOUT ] nama_param tipe [,...]])
   isi_procedure;
```

CONTOH 1

- Persiapan
  - Buat dulu sebuah table untuk menyimpan data log user

```
CREATE TABLE log_user(
  waktu datetime,
  namauser varchar(100)
)
```

CONTOH 1

- Prosedure yang akan dibuat adalah sebuah procedure yang akan digunakan untuk menyimpan informasi login seorang user.

```
DELIMITER //
CREATE PROCEDURE catat_log_user()
   INSERT INTO log_user values(now(),user()) //
DELIMITER ;
```

- Memanggil stored procedure

```
CALL catat_log_user();
```

CONTOH 1

```
SELECT * FROM log_user;
```

waktu	namauser
2009-12-31 11:28:29	root@localhost

PARAMETER

- Parameter yang digunakan dalam stored procedure terdiri dari 3 jenis yaitu :
  - Parameter IN. Parameter jenis ini digunakan hanya untuk input saya. Jika parameter ini diubah nilainya di dalam stored procedure, maka tidak akan mempengaruhi nilai variablenya setelah stored procedure dieksekusi. Parameter ini boleh dari variable atau nilai langsung
  - Parameter OUT. Parameter jenisini digunakan untuk mengeluarkan hasil proses dalam stored procedure agar bisa diterima di luar procedure. Parameter ini hanya boleh menggunakan variable.
  - Parameter INOUT. Gabungan dari IN dan OUT. Dapat mengirim dan menerima nilai variable. Parameter ini hanya boleh menggunakan variable.

PARAMETER

```
-- PEMBUATAN PROCEDURE
CREATE PROCEDURE tambahkan(IN a int,IN b int, OUT c int)
   SET c=a+b;
```

```
-- MEMANGGIL PROCEDURE
SET @c=0; -- Deklarasi var c
CALL tambahkan(100,50,@c);
SELECT @c; -- Menghasilkan nilai 150
```

SELECT ... INTO

- SELECT INTO digunakan untuk mengambil data dari sebuah query, dan menyimpan hasil querynya ke suatu variable.

```
CREATE PROCEDURE AmbilData(OUT terbesar double,
                           OUT ratarata double)
  SELECT MAX(buyprice),  AVG(buyprice)
    INTO     terbesar,      ratarata
     FROM classicmodels.products;
```

```
CALL AmbilData(@terbesar,@ratarata);
SELECT @terbesar,@ratarata; -- 103.42, 54.3951818181818
```

PERCABANGAN

- Pernyataan IF
- Pernyataan CASE

Pernyataan IF

- Struktur statement

```
IF Kondisi THEN
   statement_list
[ELSEIF Kondisi THEN
        statement_list] ...
[ELSE
     statement_list]
END IF
```

Pernyataan IF

- Contoh procedure pencarian index nilai dari nilai akhir

```
DELIMITER //
CREATE PROCEDURE CariIndeks(na double, OUT indeks char(1))
BEGIN
  IF na>=80 THEN SET indeks='A';
  ELSEIF na>=68 THEN SET indeks='B';
  ELSEIF na>=56 THEN SET indeks='C';
  ELSEIF na>=45 THEN SET indeks='D';
  ELSEIF na>=0 THEN SET indeks='E';
  ELSE SET indeks='T';
  END IF;
END//
DELIMITER ;
```

Pernyataan IF

- Penggunaan Procedure CariIndeks

```
CALL CariIndeks(78,@idx);
SELECT @idx; -- Menghasilkan B
```

Pernyataan CASE

- Struktur statement

```
CASE case_value
    WHEN when_value THEN statement_list
   [WHEN when_value THEN statement_list] ...
   [ELSE statement_list]
END CASE
```

Pernyataan CASE

- Struktur penggunaan CASE. Mencari keterangan berdasarkan indeks. Procedure CariKeterangan

```
DELIMITER //
CREATE PROCEDURE CariKeterangan(indeks char(1), OUT ket varchar(20))
BEGIN
     SET indeks=UPPER(indeks);
     CASE indeks
          WHEN 'A' THEN SET ket='Sangat Baik';
          WHEN 'B' THEN SET ket='Baik';
          WHEN 'C' THEN SET ket='Cukup';
          WHEN 'D' THEN SET ket='Kurang';
          WHEN 'E' THEN SET ket='Sangat Kurang';
          ELSE SET ket='Tidak Valid';
     END CASE;
END //
DELIMITER ;
```

Pernyataan CASE

- Penggunaan Procedure CariKeterangan

```
CALL CariKeterangan('a',@ket);

select @ket -- Menghasilkan Sangat Baik
```

PERULANGAN

- Statement REPEAT
- Statement WHILE
- Statement LOOP

Untuk contoh perulangan akan dibuat procedure untuk menghitung jumlah deret (1 + 2 + 3 + …. + N)

PERULANGAN DENGAN REPEAT

- Akan melakukan perulangan selama kondisi masih bernilai FALSE. Perulangan akan berhenti ketika kondisi bernilai TRUE.
- Struktur REPEAT

```
REPEAT
     perintah-perintah …
UNTIL kondisi_loop END REPEAT
```

PERULANGAN DENGAN REPEAT

- Procedure JumlahDeret

```
DELIMITER //
DROP PROCEDURE JumlahDeret//
CREATE PROCEDURE JumlahDeret(N int, OUT Hasil Int)
BEGIN
    DECLARE ctr int DEFAULT 0;
    SET Hasil=0;
    REPEAT
          SET Hasil=Hasil+ctr;
          SET ctr=ctr+1;
    UNTIL ctr>N END REPEAT;
END//
```

PERULANGAN DENGAN REPEAT

- Penggunaan Fungsi JumlahDeret

```
CALL JumlahDeret(5,@jumlah);
SELECT @jumlah; -- Menghasilkan 1+2+3+4+5 = 15
```

PERULANGAN DENGAN WHILE

- Akan melakukan perulangan selama kondisi masih bernilai TRUE. Perulangan akan berhenti ketika kondisi bernilai FALSE.
- Struktur WHILE

```
WHILE kondisi DO
   perintah-perintah …
END WHILE
```

PERULANGAN DENGAN WHILE

- Procedure JumlahDeret2

```
DELIMITER //
CREATE PROCEDURE JumlahDeret2(N int, OUT Hasil Int)
BEGIN
    DECLARE ctr int DEFAULT 0;
    SET Hasil=0;
    WHILE ctr<=N DO
          SET Hasil=Hasil+ctr;
          SET ctr=ctr+1;
    END WHILE;
END//
```

PERULANGAN DENGAN WHILE

- Penggunaan Fungsi JumlahDeret

```
CALL JumlahDeret2(5,@jumlah);
SELECT @jumlah; -- Menghasilkan 1+2+3+4+5 = 15
```

PERULANGAN DENGAN LOOP

- LOOP merupakan bentuk perulangan. Kondisi perulangan biasanya dilakukan dengan membuat sebuah pernyataan IF dan digabung dengan LEAVE untuk keluar dari LOOP, atau ITERATE untuk melakukan perulangan lagi.
- Struktur LOOP

```
[nama_label:]
LOOP
    [IF kondisi THEN {LEAVE/ITERATE} namal_label]
    perintah-perintah
    [IF kondisi THEN {LEAVE/ITERATE} namal_label]
END LOOP [nama_label]
```

PERULANGAN DENGAN LOOP

- Procedure JumlahDeret3

```
DELIMITER //
CREATE PROCEDURE JumlahDeret3(N int, OUT Hasil Int)
BEGIN
    DECLARE ctr int DEFAULT 0;
    SET Hasil=0;
    AWAL:
    LOOP
          SET Hasil=Hasil+ctr;
          SET ctr=ctr+1;
          IF ctr>N THEN
             LEAVE AWAL;
          END IF;
    END LOOP AWAL;
END//
```

PERULANGAN DENGAN LOOP

- Penggunaan Fungsi JumlahDeret4

```
CALL JumlahDeret3(5,@jumlah);
SELECT @jumlah; -- Menghasilkan 1+2+3+4+5 = 15
```

PERULANGAN DENGAN LOOP

- Procedure JumlahDeret4

```
DELIMITER //
CREATE PROCEDURE JumlahDeret4(N int, OUT Hasil Int)
BEGIN
    DECLARE ctr int DEFAULT 0;
    SET Hasil=0;
    AWAL:
    LOOP
        SET Hasil=Hasil+ctr;
        SET ctr=ctr+1;
        IF ctr<=N THEN
            ITERATE AWAL;
        END IF;
        LEAVE AWAL;
    END LOOP AWAL;
END//
```

PERULANGAN DENGAN LOOP

- Penggunaan Fungsi JumlahDeret

```
CALL JumlahDeret4(5,@jumlah);
SELECT @jumlah; -- Menghasilkan 1+2+3+4+5 = 15
```

Contoh Real Stored Procedure

- Buatlah suatu procedure yang digunakan untuk melakukan proses transfer antar rekening di suatu bank dengan ketentuan :
  - Saldo pengirim harus mencukupi
  - Rekening pengirim dan penerima harus ada.
  - Update masing-masing rekening sesuai besar transfer.

Contoh Aplikasi Stored Procedure

- Persiapan Tabel Rekening

```
-- STRUKTUR TABEL
CREATE TABLE rekening(
   NO INT primary key,
   NAMA VARCHAR(30) NOT NULL,
   Saldo DOUBLE NOT NULL DEFAULT 0
)ENGINE=INNODB;
```

```
INSERT INTO rekening VALUES
   (1,'Orang 1',500),(2,'Orang 2',50)
```

Contoh Aplikasi Stored Procedure

```
DELIMITER //
CREATE PROCEDURE Transfer(norek1 INT, norek2 INT, besar DOUBLE, OUT Status VARCHAR(50))
BEGIN
     DECLARE norek1_ada INT DEFAULT 0;
     DECLARE norek2_ada INT DEFAULT 0;
     DECLARE saldo_rek1 DOUBLE DEFAULT 0;
     SELECT count(*) INTO norek1_ada FROM rekening where no=norek1;
     IF norek1_ada>0 THEN
        SELECT saldo INTO saldo_rek1 FROM rekening where no=norek1;
        IF saldo_rek1>besar THEN
            SELECT count(*) INTO norek2_ada FROM rekening where no=norek2;
            IF norek2_ada>0 THEN
               UPDATE rekening SET saldo=saldo-besar WHERE no=norek1;
               UPDATE rekening SET saldo=saldo+besar WHERE no=norek2;
               SET status='Sukses';
            ELSE
                SET status=CONCAT_WS(' ','Rekening',norek2,'Tidak Ditemukan');
            END IF;
        ELSE
            SET status=CONCAT_WS(' ','Saldo Rekening',norek1,'Tidak Mencukupi');
        END IF;
     ELSE
         SET status=CONCAT_WS(' ','Rekening',norek1,'Tidak Ditemukan');
     END IF;
END//
```

Contoh Aplikasi Stored Procedure

```
CALL Transfer(3,2,40,@status);
SELECT @status; -- Rekening 3 Tidak Ditemukan

CALL Transfer(1,5,40,@status);
SELECT @status; -- Rekening 5 Tidak Ditemukan

CALL Transfer(1,2,4000,@status);
SELECT @status; -- Saldo Rekening 1 Tidak Mencukupi

CALL Transfer(1,2,40,@status);
SELECT @status; -- Sukses
SELECT * FROM REKENING; -- Rek 1 : 460, Rek 2 : 90
```