UNIVERSITAS SANATA DHARMA
YOGYAKARTA

JURUSAN TEKNIK INFORMATIKA
PROGRAM HIBAH TIK-K1 2007

Pertemuan IV

CONDITION & HANDLER
CURSOR

TUJUAN

1. Mahasiswa dapat memahami tentang berbagai kondisi yang mungkin terjadi karena suatu kesalahan tertentu
2. Mahasiswa dapat menangani berbagai kondisi kesalahan.
3. Mahasiswa dapat memahami tentang konsep cursor dan pemakaiannya.

DECLARE Conditions

Terdapat kondisi-kondisi tertentu yang perlu untuk ditangani secara khusus. Kondisi-kondisi tersebut berhubungan dengan kesalahan, seperti aliran control dalam sebuah routine.

Syntak untuk DECLARE conditions

DECLARE *condition_name* **CONDITION FOR** *condition_value*

dimana:

condition_value :

SQLSTATE [VALUE] *sqlstate_value*

| *mysql_error_code*

Condition_value dapat berupa SQLSTATE value atau mysql_error_code.

DECLARE Handler

Syntak untuk DECLARE handlers

DECLARE *handler_type* **HANDLER FOR** *condition_value[,…] statement*

Dimana:

handler_type :

```
   CONTINUE
| EXIT
| UNDO
```

condition_value:

```
    SQLSTATE [VALUE] sqlstate_value
  | condition_name
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION
  | mysql_error_code
```

DECLARE Handler (cont)

- DECLARE .. HANDLER menangani satu atau banyak kondisi yang terjadi. Jika satu dari beberapa kondisi terjadi, maka statement akan dijalankan. *Statement* dapat berupa statement yang sederhana (sebagai contoh, SET var_name=value) atau dapat juga terdiri dari beberapa statement yang ditulis dengan menggunakan BEGIN dan END.
- Jika menggunakan handler CONTINUE, statement akan dijalankan setelah eksekusi dari statement handler walaupun kondisi tertentu menyebabkan kesalahan.
- Untuk handler EXIT, eksekusi akan dihentikan dan keluar dari statement BEGIN. . . . END dimana handler dideklarasikan.
- Jika suatu kondisi terjadi dan tidak ada handler yang dideklarasikan, maka default aksinya adalah EXIT.

DECLARE Handler (cont)

Condition_value dapat berupa:

- Nilai dari SQLSTATE atau MYSQL error code
- Nama kondisi yang sudah didefinisikan sebelumnya dalam DECLARE … CONDITION
- SQLWARNING adalah singkatan dari seluruh kode SQLSTATE yang diawali dengan 01.
- NOT FOUND adalah singkatan dari seluruh kode SQLSTATE yang diawali dengan 02.
- SQLEXCEPTION adalah singkatan dari seluruh kode SQLSTATE yang tidak dapat ditangani oleh SQLWARNING maupun NOT FOUND.

Kasus :

Jika kita hendak memasukan data ke dalam sebuah tabel dan pada salah satu fieldnya diset sebagai primary key. Maka data yang masuk tidak boleh sama untuk field yang diset sebagai primary key. Jika data tersebut dipaksakan masuk apa yang akan terjadi ??

Langkah yang dilakukan dengan penggunakan CONDITION ..HANDLER

1. Buat tabel contoh4_1 dengan 1 field diberi nama t, bertipe int diset sebagai primary key.
2. Buat prosedur dengan nama spTanpaHandler()

```
DELIMITER $$
CREATE PROCEDURE `pbd`.`spTanpaHandler` ()
BEGIN
     set @x=1;
     insert into pbd.latih4_1 value(1);
     set @x=2;
     insert into pbd.latih4_1 value(1);
     set @x=3;
END$$
DELIMITER ;
call spTanpaHandler();
select @x;
```

Pesan apa yang terjadi ? Berapa nilai @x??

Langkah .. (cont)

3. Buat prosedur dengan nama spContinue

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `pbd`.`spContinue`$$
CREATE PROCEDURE `spContinue`()
BEGIN
    DECLARE CONTINUE HANDLER FOR 1062 SET @x=5;
    set @x=1;
    insert into pbd.t value(2);
    set @x=2;
    insert into pbd.t value(2);
    set @x=3;
END$$
DELIMITER ;
call spContinue();
select @x;
```

Apakah terjadi error ??? Berapakah nilai @x??

Langkah .. (cont)

4. Buat prosedur dengan nama spExit

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `pbd`.`spExit`$$
CREATE PROCEDURE `spExit`()
BEGIN

    DECLARE CONTINUE HANDLER FOR 1062  SET @x=5;
    set @x=1;
    insert into pbd.t value(3);
    set @x=2;
    insert into pbd.t value(3);
    set @x=3;
END$$
DELIMITER ;
call spExit();
select @x;
```

Apakah terjadi error ??? Berapakah nilai @x??

CURSOR

Tentang Cursor

Setiap kali perintah SQL dieksekusi oleh Server, maka akan memiliki sebuah cursor tertentu yang akan dihubungkan dengan perintah SQL tersebut. Terdapat dua jenis tipe kursor yaitu:

- Implisit cursor yang dideklarasikan untuk seluruh perintah DML dan perintah SELECT pada PL/SQL, termasuk query yang menghasilkan hanya satu record saja. Implisit cursor akan otomatis terbentuk.
- Eksplisit cursor adalah cursor yang dideklarasikan dan dinamai oleh pembuat cursor. Dapat digunakan untuk query yang menghasilkan lebih dari satu record. Dapat pula dimanipulasi melakukan statement tertentu dalam block eksekusi

Tentang Cursor (cont)

Cursor didukung di dalam store procedure dan function. Cursor memiliki sifat asensitive, read only dan non-scrolling. Asensitive berarti server bisa atau tidak bisa membuat copy ke dalam *result table*. Read only berarti bahwa hasil dari cursor hanya dapat dibaca. Non-scrolling berarti bahwa cursor hanya dapat bergerak ‘maju’ setelah sampai pada sebuah record tidak dapat ‘mundur’ lagi

Deklarasi cursor ditempatkan setelah mendeklarasikan varible, handler dan condition.

Eksplisit Cursor

Sekumpulan row yang dikembalikan oleh query multiple-row disebut dengan active set. Ukurannya adalah sejumlah record yang sesuai dengan kriteria.

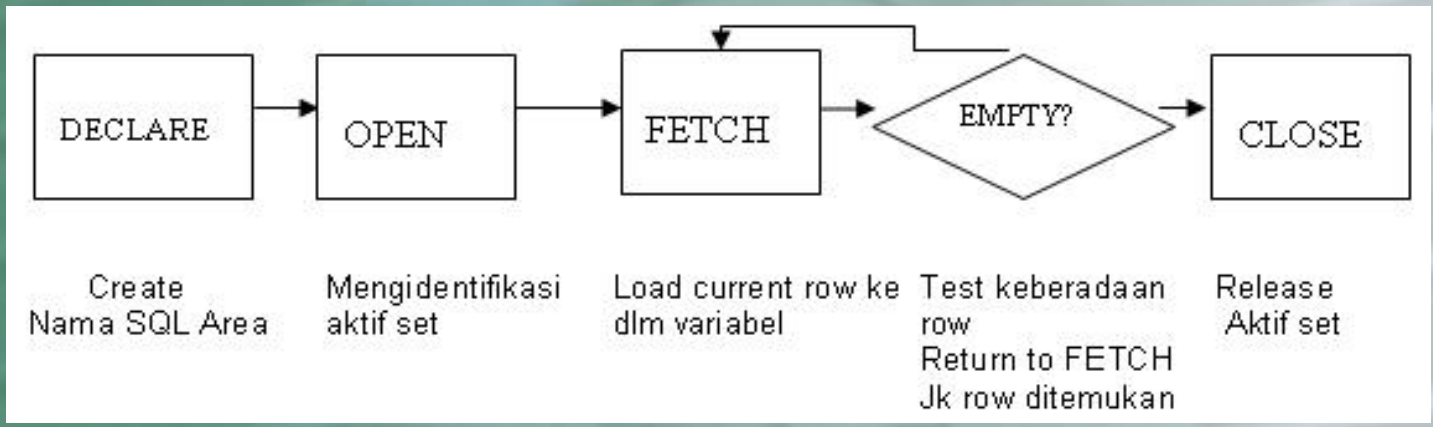
Store Procedure akan melakukan *open* cursor, memproses row yang diperoleh dari sebuah query dan kemudian *close* cursor..

Fungsi dari eksplisit kursor adalah :

a. Dapat memproses record yang dikembalikan oleh query, record per record.
b. Dapat men'track' record yang mana yang sedang diproses.
c. Memberikan keleluasaan kepada programmer untuk mengontrol eksplisit kursor secara manual dalam block store procedure.

Eksplisit Cursor (cont)

Berikut adalah langkah-langkah yang dilakukan untuk memahami konsep cursor :



Eksplisit Cursor (cont)

a. Melakukan deklarasi dengan memberi nama kursor dan mendefiniskan struktur dari query yang digunakan.
b. Membuka kursor, o*pen Cursor*. Dengan menggunakan statement OPEN akan mengeksekusi query dan membindingnya dengan variabel-variabel yang mereferensinya. Record diidentifikasi oleh query yang dipanggil oleh aktif set dan record akan tersedia untuk diFETCH.(diambil)
c. FETCH data dari cursor. Data yang difetch akan dapat diperoses record per record. Setelah melakukan FETCH akan dicek apakah cursor tersebut berisi record. Jika tidak ada record maka cursor harus ditutup.
d. CLOSE cursor. Menggunakan statement CLOSE untuk merelease aktif set dari row. Memungkin untuk membuka kembali kursor dan merefresh aktif set.

Jadi akan ada 3 statement yang digunakan untuk mengontrol eksplisit cursor yaitu OPEN, FETCH dan CLOSE.

Sintak untuk Cursor

a. Untuk deklarasi cursors

```
DECLARE cursor_name CURSOR FOR select_statement
```

- Tidak boleh menggunakan clausa INTO pada select_statement cursor. Jika pemrosesan record membutuhkan urutan tertentu maka dapat menggunakan clause ORDER BY.
- Banyak deklarasi cursor diijinkan dalam sebuah store procedure, tetapi setiap cursor haruslah memiliki nama yang unik.

Sintak untuk Cursor (cont)

b. Untuk statement OPEN cursor

```
OPEN cursor_name
```

- Statement ini digunakan untuk membuka cursor setelah dideklarasikan.
- OPEN statement akan mengeksekusi query yang sesuai dengan deklarasi cursor, mengidentifikasi result set dan menentukan posisi cursor sebelum record pertama.

Sintak untuk Cursor (cont)

Operasi yang dilakukan pada saat mengeksekusi statement OPEN adalah:

a. Mengalokasikan memori secara dinamik.
b. Memparsing statement SELECT
c. Membinding variabel input
d. Mengidentifikasi aktif set
e. Memposisikan pointer segera sebelum record pertama pada aktif set.

Sintak untuk Cursor(cont)

c. Untuk statement FETCH cursors

```
FETCH cursor_name INTO var_name [, var_name] ...
```

Statement ini digunakan untuk melakukan fetch / mengambil record berikutnya (jika ada record yang ditemukan) dengan menggunakan open cursor dan advance cursor pointer.

Petunjuk:

1. Jumlah variabel yang dimasukan ke dalam klausa INTO dari statement FETCH harus sama dengan kolom dalam perintah SELECT, dan pastikan bahwa tipe data juga sesuai.
2. Cocokan setiap variabel dengan kolomnya, posisi harus sama.
3. Ujilah kursor apakah berisi record atau tidak. Jika tidak ada nilai, maka fetch akan meninggalkan proses dalam aktif record dan tidak ada record yang direkam.

Sintak untuk Cursors (cont)

Cara kerja FETCH statement:

a. Membaca data dari record tertentu ke dalam variabel output PL/SQL.
b. Advances pointer ke record berikutnya.

FETCH ini digunakan untuk menampilkan nilai dari record tertentu ke dalam variabel output. Setelah dilakukan fetching maka data dalam variabel tersebut dapat dimanipulasi.

Sintak untuk Cursor (cont)

d. Untuk statement CLOSE cursor

```
CLOSE cursor_name
```

- Statement ini digunakan untuk menutup cursor yang telah terbuka. Statement CLOSE akan membuat kursor menjadi disable, dan aktif set menjadi undefined. Gunakanlah statement ini setelah selesai melakukan proses dengan perintah SELECT.
- Dapat dilakukan reopen kembali kursor, jika memang dibutuhkan. Setelah statement CLOSE maka tidak dapat melakukan FETCH data kembali.
- Walaupun memungkinkan untuk tidak menutup block routine dengan closing cursor, akan tetapi jadikan hal ini sebagai sebuah kebiasaan untuk menutup kursor sehingga akan bebas.

Contoh

```
DELIMITER $$
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE a CHAR(16);
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM
   pbd.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM pbd.t2;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET
   done = 1;
  OPEN cur1;
  OPEN cur2;
```

Contoh (cont)

```
REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
          INSERT INTO pbd.t3 VALUES (a,b);
       ELSE
          INSERT INTO pbd.t3 VALUES (a,c);
       END IF;
    END IF;
  UNTIL done END REPEAT;
  CLOSE cur1;
  CLOSE cur2;
END
$$
DELIMITER ;
```

Contoh (cont)

Jika diketahui data untuk

T1 :

Data1	1
Data2	7
Data3	2

T2 :

2
5
3
8

Berapakah hasil t3 ??