Praktikum PHP MySQL Part 5: Cara Menampilkan Pesan Kesalahan (Error) mysqli Extension

Jika dalam 2 Praktikum sebelumnya kita membahas tentang cara menampilkan data MySQL menggunakan procedural style mysqli dan object style mysqli, dalam Praktikum PHP MySQL kali ini kita akan mempelajari fungsi dan property yang bisa digunakan untuk menampilkan pesan kesalahan (error) di dalam mysqli.

Pesan kesalahan atau error yang dimaksud dalam Praktikum kali ini adalah pesan error dari MySQL yang bisa ditampilkan apabila terjadi kesalahan koneksi atau kesalahan penulisan query MySQL. Pesan error dari PHP kadang tidak mencukupi, atau tidak bisa ditampilkan langsung kepada pengguna.

Karena itu, kita perlu sebuah cara untuk mengetahui apa yang terjadi. Misalkan apakah MySQL Server sudah berjalan, atau apakah kita salah menuliskan password user, database belum ada, salah penulisan query, dll. Dalam Praktikum ini kita akan membahasnya dengan lebih dalam.

Karena mysqli memiliki 2 cara penulisan, yakni **procedural style** dan **object style**, saya akan membahas keduanya secara terpisah.

Cara Menampilkan Pesan Kesalahan (Error) Procedural Style mysqli

Untuk menampilkan pesan kesalahan dalam **procedural style mysqli**, kita akan menggunakan fungsi-fungsi. Beberapa diantaranya adalah fungsi **mysqli_connect_errno(), mysqli_connect_ error (), mysqli_errno(),** dan **mysqli_error()**. Agar mudah dipahami, kita akan langsung membahasnya menggunakan contoh kode program.

Pertama kali, saya akan mencoba membuat contoh kode program untuk menampilkan kesalahan pada saat proses koneksi dengan MySQL Server. Berikut adalah kode program PHPnya:

```
<?php

    // buat koneksi dengan MySQL, gunakan database:
```

```
universitas
     $link      =      mysqli_connect('localhost',      'root',
'','universitas');

     // cek koneksi
     if (!$link) {
          die('Koneksi  Error  :  '.mysqli_connect_errno()  .'  -
'.mysqli_connect_error());
     }

     // koneksi berhasil
     echo  'Koneksi  Berhasil  :'.mysqli_get_host_info($link)."<br
/>";

     // tutup koneksi
     mysqli_close($link);
?>
```

Pada kode program diatas, setelah proses koneksi dengan fungsi **mysqli_connect()**, saya memeriksa hasil koneksi dengan kondisi **if(!$link).** Fungsi **mysqli_connect()** akan mengembalikan link koneksi apabila koneksi ke MySQL sukses dilakukan. Tetapi jika koneksi gagal, fungsi ini akan mengembalikan nilai **FALSE**. Nilai kembalian inilah yang bisa kita manfaatkan untuk memeriksa apakah koneksi berhasil atau gagal. Karena kondisi IF baru akan berjalan jika kondisi bernilai TRUE, maka saya menambahkan tanda ! untuk membalik nilai **FALSE** menjadi **TRUE**. Kondisi **if(!$link)** baru akan dieksekusi ketika fungsi **mysqli_connect()** mengalami kegagalan.

Jika kondisi **if(!link)** menjadi **TRUE** (terdapat error), maka fungsi **die()** akan menghentikan proses PHP yang sedang berjalan. Selanjutnya, saya menampilkan pesan kesalahan menggunakan fungsi **mysqli_connect_errno()** dan **mysqli_connect_ error()**. Fungsi *mysqli_connect_errno()* akan menampilkan **nomor kode error**, sedangkan fungsi *mysqli_connect_error()* akan menampilkan pesan error.)

Sebagai contoh, jika saya sengaja mengubah **username** menjadi "*teman_root*" (yang memang tidak ada di MySQL), maka kita bisa melihat error yang terjadi:

```
1
2    Koneksi Error : 1044

     - Access denied for user ''@'localhost' to database
     'universitas'.
```

Atau jika database saya ganti jadi "*universitas_tetangga":*

```
1
     Koneksi Error : 1049 - Unknown database
     'universitas_tetangga'.
```

Pesan error yang dihasilkan ini selanjutnya bisa dikirim ke user agar bisa memperbaiki kesalahan tersebut.

Apabila koneksi berhasil, fungsi **mysqli_get_host_info()** bisa digunakan untuk menampilkan keterangan mengenai jenis koneksi apa yang saat ini diakses, apakah dari localhost atau alamat IP.

Jika kita menggunakan fungsi mysqli_connect('localhost', 'root', '', 'universitas'), maka hasil pemanggilan fungsi **mysqli_get_host_info()** adalah:

```
1
    Koneksi berhasil : localhost via TCP/IP
```

Jika saya mengubahnya menjadi mysqli_connect('127.0.0.1', 'root', '', 'universitas'), hasilnya menjadi:

```
1
    Koneksi berhasil : 127.0.0.1 via TCP/IP
```

Selanjutnya, bagaimana cara menampilkan pesan error yang terjadi pada saat **query** di jalankan? Kita bisa menggunakan fungsi **mysqli_errno()** dan **mysqli_error()**. Berikut adalah contoh cara penggunaannya:

```
<?php
    // buat koneksi dengan MySQL, gunakan database:
universitas
    $link = mysqli_connect('localhost', 'root',
'','universitas');

    // cek koneksi
    if (!$link) {
        die('Koneksi Error : '.mysqli_connect_errno().' -
'.mysqli_connect_error());
    }

    // koneksi berhasil
    echo 'Koneksi Berhasil :'.mysqli_get_host_info($link)."<br
/>";

    // jalankan query
    $result = mysqli_query($link,"SELECT * FROM
mahasiswa_ilkom");

    // cek hasil query
    if (!$result) {
        die('Query Error : '.mysqli_errno($link). ' -
'.mysqli_error($link));
    }
```

```
    // tampilkan query
    while ($row=mysqli_fetch_row($result)) {
        echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
        echo "<br />";
    }

    // tutup koneksi
    mysqli_close($link);
?>
```

Dalam kode program diatas saya melakukan pengecekan apakah query berjalan sukses atau tidak dengan kondisi **if (!$result).** Jika di dalam query terdapat kesalahan, maka fungsi **die()** akan dijalankan. Sama seperti fungsi mysqli_connect(), fungsi mysqli_query() juga akan mengembalikan nilai **FALSE** jika query gagal dijalankan.

Fungsi **mysqli_errno()** digunakan untuk menampilkan **nomor kode error**, dan fungsi **mysqli_error()** digunakan untuk menampilkan **pesan error** yang terjadi.

Sebagai contoh, jika saya mengganti query menjadi *SELECT * FROM mahasiswa_mipa*, maka hasilnya adalah sebagai berikut:

```
1
    Query Error : 1146 - Table 'universitas.mahasiswa_mipa'
    doesn't exist
```

Dimana dalam database **universitas** yang saya gunakan, memang tidak terdapat tabel **mahasiswa_mipa**. Pesan kesalahan ini sangat berguna terutama jika kita membuat program dimana user bisa menginput query sendiri.

Dalam 2 contoh kode program diatas, saya menampilkan pesan kesalahan menggunakan mysqli dengan *procedural style*. Bagaimana dengan object style mysqli? Mari kita lihat cara penulisannya.

Cara Menampilkan Pesan Kesalahan (Error) Object Style mysqli

Untuk menampilkan pesan kesalahan dengan menggunakan **object style mysqli**, kita tidak lagi menggunakan fungsi, tetapi dengan memeriksa **property** error dari objek **mysqli**. Property yang digunakan untuk menampilkan pesan kesalahan ini memiliki nama yang mirip dengan fungsi yang digunakan pada *procedural style mysqli.*

Berikut adalah contoh kode program untuk menampilkan kesalahan MySQL menggunakan mysqli dengan object style:

```
<?php
     // buat koneksi dengan MySQL, gunakan database: 
universitas
     $mysqli = new mysqli("localhost", "root", 
"","universitas");

     // cek koneksi
     if ($mysqli->connect_errno) {
          die('Koneksi Error: '.$mysqli->connect_errno.' - '. 
$mysqli->connect_error);
     }

     // koneksi berhasil
     echo 'Koneksi Berhasil : '.$mysqli->host_info."<br />";

     // jalankan query
     $result = $mysqli->query("SELECT * FROM mahasiswa_ilkom");

     // cek hasil query
     if ($mysqli->errno) {
          die('Query Error : '.$mysqli->errno.'-'.$mysqli-
>error);
     }

     // tampilkan query
     while ($row= $result->fetch_row()) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
          echo "<br />";
     }

     // tutup koneksi
     $mysqli->close();
?>
```

Perhatikan bahwa untuk mengecek apakah suatu koneksi dan proses query berhasil atau tidak, kita bisa mengujinya dengan mengecek “isi” dari property **$mysqli->connect_errno** dan **$mysqli->errno**. Kedua property ini tidak akan berisi apa-apa jika tidak ada kesalahan, dan baru akan berisi nilai jika terdapat error pada koneksi

MySQL atau query MySQL.

Dengan kata lain, untuk memeriksa apakah pada saat koneksi MySQL terjadi kesalahan, kita bisa mengujinya dengan kondisi **if ($mysqli->connect_errno)**. Sedangkan untuk mengecek query, bisa menggunakan **if ($mysqli->errno)**.

Anda bisa menguji kode diatas dengan melakukan "*kesalahan*", dan melihat apakah kode program diatas bisa menangkap error yang terjadi.

Cara penanganan error dalam contoh *object style mysqli* diatas sebenarnya belum murni "*objek*". Menggunakan **kondisi IF** dan fungsi **die()** untuk menangani kesalahan merupakan "*cara procedural styl*e". Di dalam OOP, kita sebaiknya menggunakan kondisi **TRY…CATCH** untuk penanganan kesalahan (kita akan melihat contohnya saat membahas tentang PDO dalam Praktikum mengenai PDO PHP).

Dalam Praktikum ini saya tidak menggunakannya agar contoh kita menjadi lebih sederhana.

Dalam Praktikum belajar PHP MySQL kali ini kita telah membahas cara menangani dan menampilkan pesan kesalahan jika terjadi error pada saat koneksi MySQL maupun pada saat query dijalankan. Pesan error ini akan berguna jika kita membuat program dimana user bisa menginput query sendiri. Sehingga jika terjadi salah penulisan query, kita bisa menampilkan pesan yang sesuai.

Praktikum PHP MySQL Part 6: Pengertian dan Cara Penggunaan Prepared Statements mysqli

Salah satu fitur baru yang tersedia di dalam **mysqli extension** yang tidak ada pada **mysql extension** adalah dukungan untuk **prepared statements** MySQL. Dalam Praktikum belajar PHP MySQL kali ini kita akan membahas tentang pengertian prepared statements dan cara penggunaan prepared statements dengan mysqli.

PHP mendukung fitur **prepared statements** pada **mysqli extension** dan **PDO**. Fitur ini juga umum digunakan di dalam pemrograman PHP lanjutan seperti *framework* PHP. Karena hal ini, penting juga bagi kita untuk memahami cara penggunaannya.

Pengertian Prepared Statements MySQL

Prepared statements adalah sebuah fitur yang disediakan MySQL (dan juga beberapa aplikasi database lainnya), dimana kita bisa mengirim **query** (perintah) secara terpisah antara query inti dengan “data” dari query. Tujuannya, agar query menjadi lebih aman dan cepat (jika perintah yang sama akan digunakan beberapa kali).

Sebagai perbandingan, untuk menampilkan data MySQL menggunakan fungsi **mysqli_query()**, kita membuat seluruh query dalam 1 string dan langsung mengirimkannya ke MySQL Server, sebagai berikut:

```
1
2   $result=mysqli_query("SELECT * FROM mahasiswa_ilkom WHERE
    nama='Neil Situmorang'");
```

Dengan **prepared statements**, query tersebut akan dipisah antara perintah query: “SELECT…” dengan ‘data’-nya yakni “*Neil Situmorang*”.

Proses Pembuatan Prepared Statement MySQL

Proses pembuatan *prepared statements* membutuhkan 3 langkah: **Prepared, Bind,** dan **Execute**.

Pada proses pertama: **prepared**, kita mempersiapkan query yang akan dijalankan, tetapi tanpa ‘data’. Bagian dimana ‘data’ berada digantikan dengan tanda tanya (?), seperti berikut ini:

"SELECT * FROM mahasiswa_ilkom WHERE nama=?"

atau

"INSERT INTO mahasiswa_ilkom VALUES (?, ?, ?, ?, ?)"

Secara teknis, query diatas akan langsung di kirim PHP ke MySQL Server. Di dalam MySQL, perintah tersebut disimpan untuk sementara menunggu proses berikutnya: **bind**.

Proses kedua adalah **bind**. Dalam tahap ini, kita akan mengirimkan data yang telah ditandai dalam proses *prepare*. Data disini adalah bagian yang diberi tanda “?”. Jika di dalam proses *prepare* hanya butuh 1 data, kita mengirimkan 1 data. Tetapi jika kita butuh 5 data, kita mengirimkan 5 data (sesuai dengan query yang ditulis dalam tahap *prepare*).

Setelah proses *prepare* dan *bind*, berikutnya adalah menjalankan prepared statement (**execute**).

Kenapa Harus Menggunakan Prepared Statements?

Keuntungan terbesar dari penggunaan *prepared statements* adalah dalam hal keamanan. Untuk aplikasi '*nyata'*, bagian "data" dari suatu query biasanya berasal dari user. Seorang user yang jahil bisa saja menambahkan perintah SQL pada kotak inputan *user name*. Metoda ini dikenal juga dengan **SQL Injection**.

Dengan memisahkan perintah query dengan datanya, kita bisa mencegah penyisipan query.

Cara Penggunaan Prepared Statements mysqli

Pembahasan mengenai *prepared statements* cukup panjang dan akan saya bahas secara bertahap dalam 2 Praktikum berikutnya. Sebagai gambaran, berikut adalah contoh kode program untuk menampilkan data mahasiswa dari tabel **mahasiswa_ilkom** menggunakan *prepared statement* mysqli (*procedural style*). Penjelasan mengenai kode program ini akan kita bahas lengkap pada Praktikum berikutnya:

```
<?php

// buat koneksi dengan MySQL, gunakan database: universitas

$link = mysqli_connect('localhost', 'root', '','universitas');



// cek koneksi

if (!$link) {

     die('Koneksi   Error   :   '.mysqli_connect_errno().'   -
'.mysqli_connect_error());

}



// buat prepared statements

$stmt  =  mysqli_prepare($link,  "SELECT  *  FROM  mahasiswa_ilkom
```

```
WHERE nama=?");



// siapkan "data" query

$nama_mhs="Neil Situmorang";



// hubungkan "data" dengan prepared statements: bind

mysqli_stmt_bind_param($stmt, "s", $nama_mhs);



// jalankan query: execute

mysqli_stmt_execute($stmt);



// cek hasil query

if (!$stmt) { die('Query Error : '.mysqli_errno($link).' -
'.mysqli_error($link));

}



// ambil hasil query

$result=mysqli_stmt_get_result($stmt);



// tampilkan hasil query

while ($row= mysqli_fetch_row($result)) {

     echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
```

```
     echo "<br />";

 }



// tutup statements

mysqli_stmt_close($stmt);



// tutup koneksi

mysqli_close($link);

?>
```

Jika kode program diatas dijalankan, hasilnya adalah sebagai berikut:

```
 1
     099145055 Neil Situmorang 22 Medan 1.90
```

Disini saya menggunakan database **universitas**, dan tabel **mahasiswa_ilkom**. Database dan tabel ini kita buat pada Praktikum PHP MySQL: Cara Menampilkan Tabel MySQL dari PHP (mysql_fetch_row).

Dalam program diatas, saya menggunakan **prepared statements** untuk menampilkan data **mahasiswa_ilkom** dengan *nama = "Neil Situmorang"*. Saya akan membahas kode program diatas secara detail dalam Praktikum berikutnya: Cara Menampilkan Data MySQL Menggunakan Prepared Statements mysqli.

Praktikum PHP MySQL Part 7: Cara Menampilkan Data dengan mysqli Prepared Statements

Dalam Praktikum belajar PHP MySQL kali ini kita akan membahas tentang penggunaan **prepared statements**, yakni cara menampilkan data MySQL menggunakan mysqli prepared statements.

Seperti biasa, karena mysqli memiliki 2 jenis gaya pemrograman (*procedural style* dan *object style*), saya akan membahas keduanya secara terpisah.

Cara Menampilkan Data MySQL Menggunakan Prepared Statements mysqli

Setelah membahas teori tentang pengertian prepared statements pada Praktikum sebelumnya, kita akan langsung praktek mengenai cara penggunaan *prepared statements* untuk menampilkan sebuah data dari database MySQL.

Kali ini saya akan mencoba menampilkan tabel **mahasiswa_ilkom** dengan *prepared statements*. Tabel ini kita buat pada Praktikum PHP MySQL: Cara Menampilkan Tabel MySQL dari PHP (mysql_fetch_row).

Seperti yang telah dibahas, untuk membuat prepared statements, kita membutuhkan 3 langkah: **prepared, bind,** dan **excecute**.

Langkah pertama: *prepared*

Sebagai contoh, saya ingin menampilkan seluruh kolom dari tabel **mahasiswa_ilkom** dimana nama mahasiswanya adalah "**Neil Situmorang**". Sesuai dengan fungsinya, di dalam proses **prepared** ini kita hanya butuh mempersiapkan query MySQL, tanpa ada data.

Untuk proses *prepared*, mysqli PHP menyediakan fungsi **mysqli_prepare()**. Fungsi ini membutuhkan 2 argumen, yakni variabel hasil pemanggilan fungsi **mysqli_connect()**, dan *prepared query* yang akan dijalankan. Berikut adalah contoh penulisannya:

```
<?php
     // buat koneksi dengan MySQL, gunakan database:
universitas
     $link = mysqli_connect('localhost', 'root',
'','universitas');

     // buat prepared statements
     $stmt = mysqli_prepare($link, "SELECT * FROM
mahasiswa_ilkom WHERE nama=?");
?>
```

Hasil pemanggilan fungsi **mysqli_prepare()** selanjutnya disimpan kedalam variabel **$stmt**. Variabel ini akan kita gunakan di dalam proses **bind**, **execute** dan dalam proses menampilkan data. Anda bebas jika ingin menukar variabel ini dengan nama lain.

Secara internal, ketika kita menggunakan fungsi **mysqli_prepare()**, query tersebut langsung di kirim ke MySQL Server. Selanjutnya di dalam MySQL, query akan disimpan sementara menunggu proses **bind**.

Langkah kedua: ***bind***

Pada proses **bind,** kita akan mengirimkan data kepada MySQL. Data yang akan dikirim adalah untuk menggantikan tanda "?" yang sebelumnya dibuat pada **proses prepared**. Di dalam **mysqli PHP**, proses *bind* dilakukan menggunakan fungsi

mysqli_stmt_bind_param(). Fungsi ini membutuhkan setidaknya 3 buah argumen, berikut contohnya:

```
<?php
     // siapkan "data" query
     $nama_mhs="Neil Situmorang";

     // hubungkan "data" dengan prepared statements
     mysqli_stmt_bind_param($stmt, "s", $nama_mhs);
?>
```

Argumen pertama dari fungsi **mysqli_stmt_bind_param()** adalah variabel hasil pemanggilan fungsi **mysqli_prepare()**, dalam contoh kita adalah varabel **$stmt**.

Argumen kedua adalah *string* yang menunjukkan jenis tipe data argumen ketiga, yakni data yang akan diinput kedalam query (kita akan membahas isi argumen kedua ini sesaat lagi).

Argumen ketiga adalah data yang akan menggantikan tanda "?" dari query, dalam contoh kita adalah "*Neil Situmorang*". Tetapi karena fungsi **mysqli_stmt_bind_param()** membutuhkan data dalam bentuk variabel, saya harus menyimpannya terlebih dahulu ke dalam variabel **$nama_mhs**.

Argumen kedua dari fungsi **mysqli_stmt_bind_param()** membutuhkan pembahasan tersendiri. Argumen ini berisi data string yang menunjukkan jenis tipe data argumen ketiga. PHP menyediakan 4 jenis tipe data:

- i = variabel bertipe integer
- d = variabel bertipe double
- s = variabel bertipe string
- b = variabel bertipe blob (binary)

Karena di dalam contoh saya mengunakan variabel **$nama_mhs** yang bertipe string, maka di dalam argumen kedua ditulis: "s". Tetapi apabila saya mengganti variabel ketiga menjadi **umur** yang bertipe integer, maka argumen kedua ini menjadi "i".

Langkah ketiga: *execute*

Setelah proses **bind** selesai, langkah berikutnya adalah menjalankan query dengan menggunakan fungsi **mysqli_stmt_execute()**. Fungsi ini membutuhkan 1 buah argumen, yakni variabel hasil pemanggilan fungsi **mysqli_prepare():**

```
<?php
     mysqli_stmt_execute($stmt);
?>
```

Fungsi **mysqli_stmt_execute()** menginstruksikan kepada MySQL untuk segera menjalankan perintah *prepared statement* yang telah dibuat. Sampai disini proses menjalankan perintah query telah terkirim ke MySQL. Selanjutnya, kita akan menampilkan data hasil query.

Menampilkan data hasil query

Untuk menampilkan hasil query, kita mengambil data MySQL dengan fungsi **mysqli_stmt_get_result()**. Fungsi ini membutuhkan 1 argumen berupa variabel hasil fungsi **mysqli_prepare()**. Fungsi ini juga mengembalikan nilai bertipe *resources* yang selanjutnya bisa digunakan untuk menampilkan data.

```
<?php
     // ambil hasil query
     $result=mysqli_stmt_get_result($stmt);
?>
```

Selanjutnya, untuk menampilkan data kita bisa menggunakan cara biasa menggunakan **mysql_fetch_row()** atau **mysql_fetch_array()**:

```
<?php
     // tampilkan hasil query
     while ($row= mysqli_fetch_row($result)) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
          echo "<br />";
     }
?>
```

Langkah terakhir yang bersifat opsional (pilihan) adalah menutup proses *prepared statement* dengan fungsi **mysqli_stmt_close()**. Tetapi sama dengan fungsi **mysqli_close(),** jika kita tidak menulisnya, PHP secara otomatis akan menutup koneksi ke MySQL saat halaman selesai di proses.

Dengan menggabungkan seluruh fungsi-fungsi *prepared statements* yang telah kita bahas, berikut adalah contoh kode programnya secara lengkap:

```
<?php
     // buat koneksi dengan MySQL, gunakan database: universitas
     $link = mysqli_connect('localhost', 'root', '','universitas');

     // cek koneksi
     if (!$link) {
          die('Koneksi Error : '.mysqli_connect_errno().' - '.mysqli_connect_error());
     }
```

```
	// buat prepared statements
	$stmt = mysqli_prepare($link, "SELECT * FROM
mahasiswa_ilkom WHERE nama=?");

	// cek query
	if (!$stmt) {
		die('Query Error : '.mysqli_errno($link).' -
'.mysqli_error($link));
	}

	// siapkan "data" query
	$nama_mhs="Neil Situmorang";

	// hubungkan "data" dengan prepared statements
	mysqli_stmt_bind_param($stmt, "s", $nama_mhs);

	// jalankan query
	mysqli_stmt_execute($stmt);

	// ambil hasil query
	$result=mysqli_stmt_get_result($stmt);

	// tampilkan hasil query
	while ($row= mysqli_fetch_row($result)) {
		echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
		echo "<br />";
	}

	// tutup statements
	mysqli_stmt_close($stmt);

	// tutup koneksi
	mysqli_close($link);
?>
```

Jika kode program diatas dijalankan, hasilnya adalah sebagai berikut:

```
1
	099145055 Neil Situmorang 22 Medan 1.90
```

Selain fungsi-fungsi yang kita bahas pada Praktikum kali ini, saya juga menambahkan fungsi untuk memeriksa kesalahan seperti yang pernah kita bahas pada Praktikum cara menampilkan pesan kesalahan (error) mysqli, yakni fungsi **mysqli_errno** dan **mysqli_error()**.

Prepared Statement mysqli Object Style

Sebagai alternatif, berikut adalah cara penulisan **prepared statement** menggunakan *object style mysqli*. Method yang digunakan relatif hampir sama dengan *procedural style* yang kita bahas diatas.

```
<?php
     // buat koneksi dengan MySQL, gunakan database:
universitas
     $mysqli = new mysqli("localhost",
"root","","universitas");

     // cek koneksi
     if ($mysqli->connect_errno) {
          die('Koneksi gagal: ' .$mysqli->connect_errno.' -
'.$mysqli->connect_error);
     }

     // buat prepared statements
     $stmt = $mysqli->prepare("SELECT * FROM mahasiswa_ilkom
WHERE nama=?");

     // cek query
     if (!$stmt) {
          die('Query Error : '.$mysqli->errno.' - '.$mysqli-
>error);
     }

     // siapkan "data" query
     $nama_mhs="Neil Situmorang";

     // hubungkan "data" dengan prepared statements
     $stmt->bind_param("s", $nama_mhs);

     // jalankan query
     $stmt->execute();

     // hubungkan hasil query
     $result = $stmt->get_result();

     // tampilkan query
     while ($row= $result->fetch_row()) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
          echo "<br />";
     }

     // tutup statements
     $stmt->close();

     // tutup koneksi
     $mysqli->close();
```

```
?>
```

Dalam Praktikum PHP MySQL kali ini kita telah membahas tentang cara penggunaan prepared statements MySQL untuk menampilkan data. Pada Praktikum berikutnya kita akan bahas cara menggunakan mysqli prepared statement untuk proses input data kedalam database MySQL.

Praktikum PHP MySQL Part 8: Cara Menginput Data dengan mysqli Prepared Statements

Cara Menginput Data dengan mysqli Prepared Statements

Perbedaan antara cara menginput data dan menampilkan data menggunakan **prepared statements** terletak pada cara penanganan query. Apabila dalam menampilkan data kita menggunakan perintah **SELECT**, maka untuk menginput data kita menggunakan query **INSERT**.

Fitur keamanan yang diberikan oleh *prepared statements* membuatnya cocok digunakan untuk perintah yang akan mengubah isi tabel seperti perintah **INSERT** atau **UPDATE**. Selain itu, penggunaan *prepared statement* untuk proses input data yang berulang juga akan mempercepat proses eksekusi.

Dalam contoh kasus kita kali ini, saya akan menambahkan 1 buah data tambahan kedalam tabel **mahasiswa_ilkom**. Saya juga tidak akan membahas secara mendalam fungsi-fungsi *prepared statement* yang digunakan, karena kita telah membahasnya pada Praktikum sebelumnya.

Proses pertama: prepared

Untuk menginput data kedalam tabel **mahasiswa_ilkom**, kita akan menggunakan query **INSERT…VALUES** (Jika anda ingin mempelajari cara-cara menginput data kedalam MySQL, silahkan kunjungi Praktikum MySQL: Cara menambahkan data kedalam tabel).

Berikut adalah fungsi **mysqli_prepare() untuk proses penambahan data:**

```
<?php
     // buat koneksi dengan MySQL, gunakan database:
universitas
     $link = mysqli_connect('localhost', 'root',
'','universitas');
```

```
    // buat prepared statements
    $stmt = mysqli_prepare($link, "INSERT INTO mahasiswa_ilkom
VALUES (?, ?, ?, ?, ?)");
?>
```

Perhatikan cara penulisan query INSERT diatas. Karena fungsi **prepared statements** yang memisahkan query dengan data, maka di kolom **VALUES**, kita tidak langsung menuliskan data yang akan ditambahkan, tetapi menggunakan tanda "?" sebanyak 5 kali untuk penanda data yang akan diinput. Kelima data ini selanjutnya akan ditambahkan pada saat proses **bind**.

Proses kedua: bind

Proses pengiriman data (**bind**) dilakukan dengan fungsi **mysqli_stmt_bind_param()**. Karena query INSERT kita membutuhkan 5 variabel, maka fungsi **mysqli_stmt_bind_param()** juga harus menyertakan kelima variabel ini (untuk mengganti karakter (?, ?, ?, ?, ?)).

Sebelumnya, kita harus menuliskan masing-masing isian ke dalam variabel terpisah, lalu kemudian baru diinput kedalam fungsi **mysqli_stmt_bind_param()**. Berikut adalah contoh kode program dimana saya akan men-**bind** data ke dalam prepared statement:

```
<?php
    // hubungkan "data" dengan prepared statements
    mysqli_stmt_bind_param($stmt,      "ssisd",      $nim_mhs,
$nama_mhs, $umur_mhs , $tempat_lahir_mhs, $ipk_mhs);

    // siapkan "data" query
    $nim_mhs="089023020";
    $nama_mhs="Naira Alika";
    $umur_mhs=20;
    $tempat_lahir_mhs="Padang";
    $ipk_mhs=3.9;
?>
```

Dalam contoh diatas, fungsi **mysqli_stmt_bind_param()** memiliki 7 argumen. Argumen pertama adalah variabel **$stmt** hasil fungsi **mysqli_prepare()**, argumen kedua adalah string yang menunjukkan 5 tipe data. Argumen ke-3 sampai ke-7 adalah variabel yang akan berisi data yang ingin diinput.

Untuk argumen kedua saya menulis **"ssisd"**, yang merupakan singkatan dari "**s**tring **s**tring **s**tring **i**nteger **s**tring **d**ecimal". String ini akan berpasangan dengan variabel inputan ke-3 sampai ke-7:

- $nim_mhs = "089023020" (string).
- $nama_mhs = "Naira Alika" (string).

- $umur_mhs = 20 (integer).
- $tempat_lahir_mhs = Padang (string).
- $ipk_mhs = 3.9 (decimal / float).

Setelah penulisan fungsi **mysqli_stmt_bind_param()**, saya kemudian membuat data sample mahasiswa dengan nama "*Naira Alika*". Perhatikan bahwa isi data ini diinput ke dalam variabel yang sama dengan yang digunakan pada fungsi **mysqli_stmt_bind_param()** untuk argumen ke-3 s/d ke-7.

Data-data inilah yang nantinya di-*bind* dengan query asal, sehingga query kita akan menjadi:

```
1
     INSERT INTO mahasiswa_ilkom VALUES
2
     ("089023020", "Naira Alika", 20, "Padang", 3.9)
```

Proses ketiga: execute

Setelah query dan data selesai diinput, kita tinggal menjalankan query dengan fungsi **mysqli_stmt_execute()**:

```
<?php
     // jalankan query: execute
     mysqli_stmt_execute($stmt);
?>
```

Memeriksa Hasil Query

Untuk memeriksa apakah hasil query **INSERT** yang dijalankan berhasil atau tidak, bisa dilakukan dengan memeriksa variabel **$stmt** dan fungsi **mysqli_stmt_affected_rows()**, seperti contoh berikut:

```
<?php
     // cek hasil query
     if (!$stmt) {
          die('Query   Error   :   '.mysqli_errno($link).'   -
'.mysqli_error($link));
     }
     else {
          echo                                          "Penambahan
".mysqli_stmt_affected_rows($stmt)."data berhasil<br />";
          }
?>
```

Kondisi **if (!$stmt)** akan menjadi **TRUE** ketika query gagal, sehingga fungsi **die()** akan menghentikan proses dan menampilkan error yang terjadi. Tetapi jika query sukses, bagian else yang akan dijalankan.

Fungsi **mysqli_stmt_affected_rows()** mirip dengan fungsi **mysql_ affected_rows()** yang pernah kita bahas, dan akan menampilkan berapa jumlah baris yang diupdate. Karena pada contoh kali ini kita hanya menambahkan 1 buah data, maka hasilnya adalah: 1.

Agar lebih yakin, saya kemudian akan menampilkan seluruh tabel **mahasiswa_ilkom** dengan kode berikut:

```
<?php
     // jalankan query untuk memeriksa hasil inputan
     $result = mysqli_query($link, "SELECT * FROM
mahasiswa_ilkom");
     // tampilkan query
     while ($row=mysqli_fetch_row($result)) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]"
          echo "<br />";
     }
?>
```

Kode diatas sudah sering kita gunakan, sehingga saya tidak akan membahasnya lagi.

Terakhir adalah fungsi opsional untuk memutus koneksi dengan MySQL Server:

```
<?php
     // tutup statements
     mysqli_stmt_close($stmt);
     // tutup koneksi
     mysqli_close($link);
?>
```

Dengan menggabungkan seluruh pembahasan kita diatas, berikut adalah contoh kode program PHP lengkap mengenai cara menginput data MySQL menggunakan ***mysqli prepared statement***:

```
<?php
// buat koneksi dengan MySQL, gunakan database: universitas
$link = mysqli_connect('localhost', 'root', '','universitas');

// buat prepared statements
$stmt = mysqli_prepare($link, "INSERT INTO mahasiswa_ilkom
VALUES (?, ?, ?, ?, ?)");

// hubungkan "data" dengan prepared statements
mysqli_stmt_bind_param($stmt, "ssisd",$nim_mhs, $nama_mhs,
$umur_mhs , $tempat_lahir_mhs, $ipk_mhs);
```

```
// siapkan "data" query
$nim_mhs="089023020";
$nama_mhs="Naira Alika";
$umur_mhs=20;
$tempat_lahir_mhs="Padang";
$ipk_mhs=3.9;

// jalankan query
mysqli_stmt_execute($stmt);

// cek hasil query
if (!$stmt) { die('Query Error : '.mysqli_errno($link).' -
'.mysqli_error($link));
}else {
     echo "Penambahan ".mysqli_stmt_affected_rows($stmt)." data
berhasil<br />";
}

// jalankan query untuk memeriksa hasil inputan
$result = mysqli_query($link, "SELECT * FROM mahasiswa_ilkom");

// tampilkan query
while ($row=mysqli_fetch_row($result)) {
     echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
     echo "<br />";
}

// tutup statements
mysqli_stmt_close($stmt);

// tutup koneksi
mysqli_close($link);
?>
```

Apabila dijalankan, maka hasil yang didapat adalah sebagai berikut:

```
1
     Penambahan 1 data berhasil
2
     089023013 Alex Supriyanto 23 Surabaya 2.90
3    089023020 Naira Alika 20 Padang 3.90
4    089045001 Andi Suryo 23 Jakarta 2.70
5    099145055 Neil Situmorang 22 Medan 1.90
     109223041 Rani Sabrina 21 Padang 3.70
```

```
6

7
```

Bisa terlihat bahwa data mahasiswa baru: “**Naira Alika**” telah masuk kedalam tabel **mahasiswa_ilkom**.

Cara Menginput Multiple Data dengan mysqli Prepared Statements

Masih berkaitan dengan pembahasan kita, saya akan memodifikasi sedikit kode program untuk menampilkan cara menginput banyak data (*multiple data*) menggunakan **mysqli prepared statement**:

```
<?php
     // buat koneksi dengan MySQL, gunakan database:
universitas
     $link = mysqli_connect('localhost', 'root',
'','universitas');

     // buat prepared statements
     $stmt = mysqli_prepare($link, "INSERT INTO mahasiswa_ilkom
VALUES (?, ?, ?, ?, ?)");

     // hubungkan "data" dengan prepared statements
     mysqli_stmt_bind_param($stmt, "ssisd",$nim_mhs, $nama_mhs,
$umur_mhs , $tempat_lahir_mhs,$ipk_mhs);

     // siapkan "data" query 1
     $nim_mhs="089023023";
     $nama_mhs="Alika Shanum";
     $umur_mhs=21;
     $tempat_lahir_mhs="Medan";
     $ipk_mhs=3.8;

     // jalankan query 1
     mysqli_stmt_execute($stmt);

     // cek hasil query 1
     if (!$stmt) {
          die('Query Error : '.mysqli_errno($link).' -
'.mysqli_error($link));
     }else {
          echo "Penambahan
```

```
".mysqli_stmt_affected_rows($stmt)."data berhasil<br />";
     }

     // siapkan "data" query 2
     $nim_mhs="089023026";
     $nama_mhs="Rina Melita";
     $umur_mhs=22;
     $tempat_lahir_mhs="Lampung";
     $ipk_mhs=3.5;

     // jalankan query 2
     mysqli_stmt_execute($stmt);

     // cek hasil query 2
     if (!$stmt) {
          die('Query Error : ' .mysqli_errno($link).' -
'.mysqli_error($link));
     }else {
          echo "Penambahan
".mysqli_stmt_affected_rows($stmt)."data berhasil<br />";
     }

     // siapkan "data" query 3
     $nim_mhs="089023031";
     $nama_mhs="Joni Halim";
     $umur_mhs=21;
     $tempat_lahir_mhs="Palembang";
     $ipk_mhs=3.6;

     // jalankan query 3
     mysqli_stmt_execute($stmt);

     // cek hasil query 3
     if (!$stmt) {
          die('Query Error : '.mysqli_errno($link).' -
'.mysqli_error($link));
     }else {
          echo "Penambahan
".mysqli_stmt_affected_rows($stmt)."data berhasil<br />";
     }

     // jalankan query untuk memeriksa hasil inputan
     $result = mysqli_query($link, "SELECT * FROM
mahasiswa_ilkom");

     // tampilkan query
     while ($row=mysqli_fetch_row($result)) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
          echo "<br />";
```

```
    }

    // tutup statements
    mysqli_stmt_close($stmt);

    // tutup koneksi
    mysqli_close($link);
    ?>
```

Kode diatas sedikit panjang, tetapi jika anda sudah memahami konsep **prepared statement**, fungsi-fungsi yang ada bisa dipahami dengan mudah.

Perhatikan bahwa ketika kita menginput banyak data, kita tidak perlu lagi mengirimkan query ke MySQL Server, tetapi cukup dengan mengubah variabel data dan menjalankan fungsi **mysqli_stmt_execute()**. Untuk situasi inilah prepared statement menjadi lebih efisien dari pada metoda **mysqli** 'biasa'.

Jika anda menjalankan kode program diatas, berikut adalah hasil yang didapat:

```
1
    Penambahan 1 data berhasil
2
    Penambahan 1 data berhasil
3
    Penambahan 1 data berhasil
4
    089023013 Alex Supriyanto 23 Surabaya 2.90
5
    089023020 Naira Alika 20 Padang 3.90
6
    089023023 Alika Shanum 21 Medan 3.80
7
    089023026 Rina Melita 22 Lampung 3.50
8
    089023031 Joni Halim 21 Palembang 3.60 089045001 Andi
    Suryo 23 Jakarta 2.70
9
    099145055 Neil Situmorang 22 Medan 1.90
10
    109223041 Rani Sabrina 21 Padang 3.70
11
    109245021 Santi Syanum 21 Malang 3.20
12
```

Cara Menginput Data dengan mysqli Prepared

Statements (object style)

Dalam pembahasan diatas, saya menggunakan *procedural style mysqli*, kali ini kita akan melihat modifikasi kode programnya jika menggunakan *object style mysqli*:

```
<?php
     // buat koneksi dengan MySQL, gunakan database:
universitas
     $mysqli = new mysqli("localhost", "root",
"","universitas");

     // cek koneksi
     if ($mysqli->connect_errno) {
          die('Koneksi gagal: ' .$mysqli->connect_errno.' -
'.$mysqli->connect_error);
     }

     // buat prepared statements
     $stmt = $mysqli->prepare("INSERT INTO mahasiswa_ilkom
VALUES (?, ?, ?, ?, ?)");

     // hubungkan "data" dengan prepared statements
     $stmt->bind_param("ssisd",$nim_mhs, $nama_mhs, $umur_mhs ,
$tempat_lahir_mhs,$ipk_mhs);

     // siapkan "data" query
     $nim_mhs="089023020";
     $nama_mhs="Naira Alika";
     $umur_mhs=20;
     $tempat_lahir_mhs="Padang";
     $ipk_mhs=3.9;

     // jalankan query
     $stmt->execute();

     // cek query
     if (!$stmt) {
          die('Query Error : '.$mysqli->errno.' - '.$mysqli-
>error);
     }else {
          echo "Penambahan ".$stmt->affected_rows. " data
berhasil<br />";
     }

     // jalankan query untuk memeriksa hasil inputan
     $result = $mysqli->query("SELECT * FROM mahasiswa_ilkom");

     // tampilkan query
```

```
     while ($row= $result->fetch_row()) {
          echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
          echo "<br />";
     }

     // tutup statements
     $stmt->close();

     // tutup koneksi
     $mysqli->close();
?>
```

Dalam Praktikum belajar PHP MySQL kali ini kita telah mempelajari cara menggunakan mysqli prepared statement untuk menginput data kedalam database. Praktikum kali ini menutup sesi pembahasan tentang **mysqli**. Dalam Praktikum berikutnya kita akan berkenalan dengan extension ketiga PHP-MySQL, yakni PDO: PHP Data Object.