

Modul 8 : Transaksi Sederhana dengan metode CRUD

0.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Membuat Database dan table.
2. Mengetahui library yang di pakai dan setting database
3. Membuat Transaksi sederhana.

0.2 Alat & Bahan

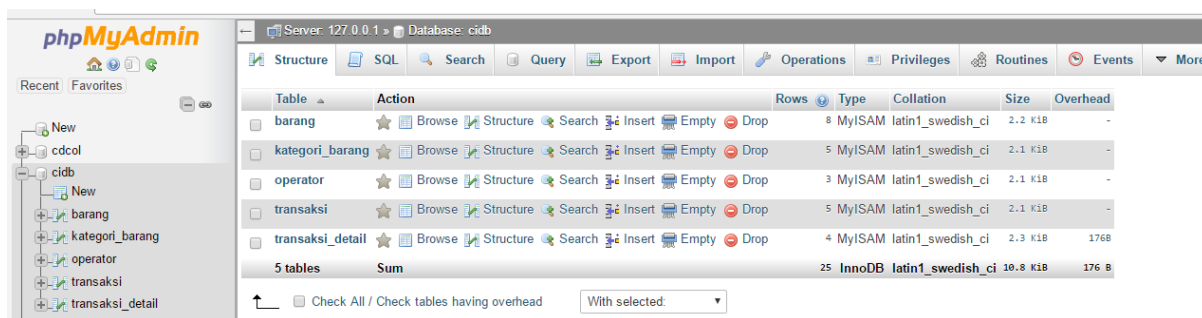
Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software xampp, framework CI, notepad++ yang telah terinstall pada masing-masing PC.

0.3 Skenario form transaksi

Dalam membuat halaman transaksi yang membedakan dengan master data adalah keterlibatan beberapa table dalam menampilkan pada saat form transaksi itu tampil dan saat penyimpanan pada saat transaksi itu selesai di lakukan. Pada contoh pembuatan transaski kali ini, pada saat form transaksi itu tampil data dari barang akan di tampilkan pada saat pilih barang, kemudian pada saat di klik simpan, maka barang akan tersimpan secara otomatis ke table detail_transaksi yang sudah di buat namun data ini bisa di hapus per item, nomor transaksi akan otomatis auto increment, jika di klik selesai maka tanggal dan id_user yang di set statis akan tersimpan pada table transaksi. Sebelum memulai membuat transaski alangkah lebih baiknya menyiapkan file CI dan membuat folder pada web server dengan nama POS_CI.

0.3.1 Membuat database dan import table

Untuk mencoba mendemonstrasikan Database MySQL dengan framework CodeIgniter terlebih dahulu kita harus membuat database dan tablenya terlebih dahulu. Langkah pertama membuat database MySQL dengan nama **cidb**, kemudian import table dengan nama file post.sql yang tersedia pada asisten praktikum masing-masing. Jika sudah benar maka table yang terbentuk akan seperti gambar berikut :



The screenshot shows the phpMyAdmin interface for a database named 'cidb'. The left sidebar shows the database structure with 'cidb' selected, containing tables: 'barang', 'kategori_barang', 'operator', 'transaksi', and 'transaksi_detail'. The main panel displays a table overview for 'cidb' with the following data:

Table	Action	Rows	Type	Collation	Size	Overhead
barang	[Browse] [Structure] [Search] [Insert] [Empty] [Drop]	8	MyISAM	latin1_swedish_ci	2.2 KiB	-
kategori_barang	[Browse] [Structure] [Search] [Insert] [Empty] [Drop]	5	MyISAM	latin1_swedish_ci	2.1 KiB	-
operator	[Browse] [Structure] [Search] [Insert] [Empty] [Drop]	3	MyISAM	latin1_swedish_ci	2.1 KiB	-
transaksi	[Browse] [Structure] [Search] [Insert] [Empty] [Drop]	5	MyISAM	latin1_swedish_ci	2.1 KiB	-
transaksi_detail	[Browse] [Structure] [Search] [Insert] [Empty] [Drop]	4	MyISAM	latin1_swedish_ci	2.3 KiB	176B
5 tables	Sum	25	InnoDB	latin1_swedish_ci	10.8 KiB	176 B

0.3.2 Mengetahui library yang dipakai.

Sebelum menulis kode program baik berupa model, controller atau vie alangkah lebih baiknya kita me-load library, helper yang diperlukan untuk pembuatan transaski ini. Yang pertama harus diperhatikan adalah onfigurasi database pada folder application\config\database.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
```

```

/*
| -----
| DATABASE CONNECTIVITY SETTINGS
| -----
| This file will contain the settings needed to access your database.
|
| For complete instructions please consult the 'Database Connection'
| page of the User Guide.
|
| -----
| EXPLANATION OF VARIABLES
| -----
|
| ['dsn']          The full DSN string describe a connection to the database.
| ['hostname']     The hostname of your database server.
| ['username']     The username used to connect to the database
| ['password']     The password used to connect to the database
| ['database']     The name of the database you want to connect to
| ['dbdriver']     The database driver. e.g.: mysqli.
|
|         Currently supported:
|             cubrid, ibase, mssql, mysql, mysqli, oci8,
|             odbc, pdo, postgre, sqlite, sqlite3, sqlsrv
| ['dbprefix']     You can add an optional prefix, which will be added
|                 to the table name when using the Query Builder class
| ['pconnect']     TRUE/FALSE - Whether to use a persistent connection
| ['db_debug']     TRUE/FALSE - Whether database errors should be displayed.
| ['cache_on']     TRUE/FALSE - Enables/disables query caching
| ['cachedir']     The path to the folder where cache files should be stored
| ['char_set']     The character set used in communicating with the database
| ['dbcollat']     The character collation used in communicating with the database
|
|                 NOTE: For MySQL and MySQLi databases, this setting is only used
|                 as a backup if your server is running PHP < 5.2.3 or MySQL <
5.0.7
|
|                 (and in table creation queries made with DB Forge).
|                 There is an incompatibility in PHP with
mysql_real_escape_string() which
|
|                 can make your site vulnerable to SQL injection if you are using
a
|
|                 multi-byte character set and are running versions lower than
these.
|
|                 Sites using Latin-1 or UTF-8 database character set and
collation are unaffected.
| ['swap_pre']     A default table prefix that should be swapped with the dbprefix
| ['encrypt']      Whether or not to use an encrypted connection.
| ['compress']     Whether or not to use client compression (MySQL only)
| ['stricton']     TRUE/FALSE - forces 'Strict Mode' connections
|
|                 - good for ensuring strict SQL while developing
| ['failover']     array - A array with 0 or more data for connections if the main
should fail.
| ['save_queries'] TRUE/FALSE - Whether to "save" all executed queries.
|
|                 NOTE: Disabling this will also effectively disable both
|                 $this->db->last_query() and profiling of DB queries.
|                 When you run a query, with this setting set to TRUE (default),
|                 CodeIgniter will store the SQL statement for debugging purposes.
|                 However, this may cause high memory usage, especially if you run
|                 a lot of SQL queries ... disable this to avoid that problem.
|
| The $active_group variable lets you choose which connection group to
| make active. By default there is only one group (the 'default' group).
|
| The $query_builder variables lets you determine whether or not to load
| the query builder class.
*/

```

```

$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn' => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'cidb',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => TRUE,
    'db_debug' => TRUE,
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);

```

Sesuaikan autoload pada folder application\config\autoload.php.

```

<?php
defined('BASEPATH') OR exit('No direct script access allowed');

/*
| -----
| AUTO-LOADER
| -----
| This file specifies which systems should be loaded by default.
|
| In order to keep the framework as light-weight as possible only the
| absolute minimal resources are loaded by default. For example,
| the database is not connected to automatically since no assumption
| is made regarding whether you intend to use it. This file lets
| you globally define which systems you would like loaded with every
| request.
|
| -----
| Instructions
| -----
|
| These are the things you can load automatically:
|
| 1. Packages
| 2. Libraries
| 3. Drivers
| 4. Helper files
| 5. Custom config files
| 6. Language files
| 7. Models
|
*/

/*
| -----
| Auto-load Packages
| -----

```

```

| Prototype:
|
|   $autoload['packages'] = array(APPPATH.'third_party', '/usr/local/shared');
|
| */
$autoload['packages'] = array('database',);

/*
| -----
|   Auto-load Libraries
| -----
| These are the classes located in the system/libraries folder
| or in your application/libraries folder.
|
| Prototype:
|
|   $autoload['libraries'] = array('database', 'email', 'session');
|
| You can also supply an alternative library name to be assigned
| in the controller:
|
|   $autoload['libraries'] = array('user_agent' => 'ua');
| */
$autoload['libraries'] = array('database', 'table');

/*
| -----
|   Auto-load Drivers
| -----
| These classes are located in the system/libraries folder or in your
| application/libraries folder within their own subdirectory. They
| offer multiple interchangeable driver options.
|
| Prototype:
|
|   $autoload['drivers'] = array('cache');
| */
$autoload['drivers'] = array();

/*
| -----
|   Auto-load Helper Files
| -----
| Prototype:
|
|   $autoload['helper'] = array('url', 'file');
| */
$autoload['helper'] = array('file','url','form','text','html');

/*
| -----
|   Auto-load Config files
| -----
| Prototype:
|

```

```

|   $autoload['config'] = array('config1', 'config2');
|
| NOTE: This item is intended for use ONLY if you have created custom
| config files. Otherwise, leave it blank.
|
| */
$autoload['config'] = array();

/*
| -----
| Auto-load Language files
| -----
| Prototype:
|
|   $autoload['language'] = array('lang1', 'lang2');
|
| NOTE: Do not include the "_lang" part of your file. For example
| "codeigniter_lang.php" would be referenced as array('codeigniter');
|
| */
$autoload['language'] = array();

/*
| -----
| Auto-load Models
| -----
| Prototype:
|
|   $autoload['model'] = array('first_model', 'second_model');
|
| You can also supply an alternative model name to be assigned
| in the controller:
|
|   $autoload['model'] = array('first_model' => 'first');
|
| */
$autoload['model'] = array('Persegi panjang model');

```

0.3.3 Membuat transaksi sederhana

Dalam membuat transaksi ini pertama kali kita akan membuat model untuk barang dan transaksi kemudian dilanjutkan dengan membuat conrollernya dan yang terakhir form transaksinya.

1. Membuat model

Untuk membuat model pada transaksi sederhana ini dibuat dua model untuk barang yang berfungsi untuk me-*loading* data agar dalam form transaksi bisa tampil daftar barang yang ada di table database dan ditampilkan pada form transaksinya dengan nama model_barang.php. adapun code program untuk modelnya adalah sebagai berikut :

```

<?php
class model_barang extends ci_model{

    function tampil_data()
    {
        $query= "SELECT b.barang_id,b.nama_barang,b.harga,kb.nama_kategori

```

```

        FROM barang as b,kategori_barang as kb
        WHERE b.kategori_id=kb.kategori_id";
    return $this->db->query($query);
}

function tampil_data_paging($halaman,$batas)
{
    $query= "SELECT b.barang_id,b.nama_barang,b.harga,kb.nama_kategori
    FROM barang as b,kategori_barang as kb
    WHERE b.kategori_id=kb.kategori_id limit $halaman,$batas";
    return $this->db->query($query);
}

function post($data)
{
    $this->db->insert('barang',$data);
}

function get_one($id)
{
    $param = array('barang_id'=>$id);
    return $this->db->get_where('barang',$param);
}

function edit($data,$id)
{
    $this->db->where('barang_id',$id);
    $this->db->update('barang',$data);
}

function delete($id)
{
    $this->db->where('barang_id',$id);
    $this->db->delete('barang');
}
}

```

Model selanjutnya adalah untuk transaksi yang berfungsi untuk menyimpan data transaksi dan detail transaksinya dengan nama model_transaksi.php. adapun code program untuk modelnya adalah sebagai berikut :

```

<?php
class model_transaksi extends ci_model
{

    function simpan_barang()
    {
        $nama_barang = $this->input->post('barang');
        $qty = $this->input->post('qty');
        $idbarang = $this->db->
>get_where('barang',array('nama_barang'=>$nama_barang))->row_array();
        $data = array('barang_id'=>$idbarang['barang_id'],
                    'qty'=>$qty,
                    'harga'=>$idbarang['harga'],
                    'status'=>'0');
        $this->db->insert('transaksi_detail',$data);
    }

    function tampilkan_detail_transaksi()
    {
        $query ="SELECT td.t detail id,td.qty,td.harga,b.nama_barang

```

```

        FROM transaksi_detail as td, barang as b
        WHERE b.barang_id=td.barang_id and td.status='0';
    return $this->db->query($query);
}

function hapusitem($id)
{
    $this->db->where('t_detail_id',$id);
    $this->db->delete('transaksi_detail');
}

function selesai_belanja($data)
{
    $this->db->insert('transaksi',$data);
    $last_id= $this->db->query("select transaksi_id from transaksi order by
transaksi_id desc")->row_array();
    $this->db->query("update transaksi_detail set
transaksi_id='".$last_id['transaksi_id']."' where status='0'");
    $this->db->query("update transaksi_detail set status='1' where
status='0'");
}
}

```

2. Membuat Controller

Untuk mengendalikan dan mengkomunikasikan antara model-model dan form yang ada maka perlu di buat controllernya dengan nama transaksi.php adapun kode programnya sebagai berikut :

```

<?php
class transaksi extends CI_controller{
    function __construct() {
        parent::__construct();
        $this->load->model(array('model_barang','model_transaksi'));
    }

    function index()
    {
        if(isset($_POST['submit']))
        {
            $this->model_transaksi->simpan_barang();
            redirect('transaksi');
        }
        else
        {
            $data['barang']= $this->model_barang->tampil_data();
            $data['detail']= $this->model_transaksi-
>tampilkan_detail_transaksi()->result();
            $this->load->view('form_transaksi',$data);
        }
    }

    function hapusitem()
    {
        $id= $this->uri->segment(3);
        $this->model_transaksi->hapusitem($id);
        redirect('transaksi');
    }

    function selesai_belanja()
    {

```

```

        $tanggal=date('Y-m-d');
        //$user= $this->session->userdata('username');
        //$id_op= $this->db->get_where('operator',array('username'=>$user))-
>row_array();
        $data=array('operator_id'=>'3','tanggal_transaksi'=>$tanggal);
        $this->model_transaksi->selesai_belanja($data);
        redirect('transaksi');
    }

}

```

3. Membuat view

Untuk menyajikan data sebagai front end, perlu dibuat form view untuk input transaksi dan sebagai sarana komunikasi antara aplikasi dengan user pengguna. dengan nama form_transaksi.php adalah sebagai berikut :

```

<h3>Form Transaksi</h3>
<?php
echo form_open('transaksi');
?>
<table class="table table-bordered">
    <tr class="success"><th>Form</th></tr>
    <tr><td>
        <div class="col-sm-6">
            <input list="barang" name="barang" placeholder="masukan nama
barang" class="form-control">
        </div>
        <div class="col-sm-1">
            <input type="text" name="qty" placeholder="QTY" class="form-
control">
        </div>
    </td></tr>
    <tr><td>
        <button type="submit" name="submit" class="btn btn-
default">Simpan</button>
        <?php echo
anchor('transaksi/selesai_belanja','Selesai',array('class'=>'btn btn-default'))?>
    </td></tr>
</table>
</form>
<table class="table table-bordered" border = 2 >
    <tr class="success"><th colspan="6">Detail Transaksi</th></tr>
    <tr><th>No</th><th>Nama
Barang</th><th>Qty</th><th>Harga</th><th>Subtotal</th><th>Cancel</th></tr>
    <?php
    $no=1;
    $total=0;
    foreach ($detail as $r)
    {
        echo "<tr>
            <td>$no</td>
            <td>$r->nama_barang</td>
            <td>$r->qty</td>
            <td>$r->harga</td>
            <td>". $r->qty*$r->harga."</td>
            <td>".anchor('transaksi/hapusitem/'.$r-
>t_detail_id,'Hapus')."</td></tr>";
        $total=$total+ ($r->qty*$r->harga);
    }
}

```



```

        $no++;
    }
    ?>
    <tr><td colspan="4"><p align="right">Total</p></td><td><?php echo
$total;?></td></tr>
</table>

<datalist id="barang">

    <?php
    foreach ($barang->result() as $b)
    {
        echo "<option value='$b->nama_barang'>";
    }
    ?>

</datalist>

```

Setelah di tulis dan di cek kode program, pastikan tidak ada kesalahan dalam penulisan kode program php. Setelah di cek semua, silahkan panggil controllernya. Jika tidak ada yang salah dalam penulisan serta konfigurasi pada Ci-nya maka akan tampil seperti berikut ini :

Form Transaksi

Form

masukan nama barang

QTY

Simpan [Selesai](#)

Detail Transaksi					
No	Nama Barang	Qty	Harga	Subtotal	Cancel
Total				0	

Jika nama barang di klik maka akan muncul data sebagai berikut.

Form Transaksi

Form

masukan nama barang ▼

- mie sedap kari ayam
- mie sedap goreng
- mie soto ayam
- mie g enak
- minuman ringan
- nokia x400
- tas kulit
- tas kertas
- indomie

Transaksi		
harga	Subtotal	Cancel
Total	0	

0.4 Latihan

1. Modifikasi kode program diatas, sehingga pada saat pertama kali muncul maka tanggal pada form transaksi akan muncul secara otomatis sesuai dengan tanggal pada sistem komputernya.
2. Simpan data transaski dan detail transaksinya.

DAFTAR PUSTAKA

- Programming PHP, Kevin Tatroe, Peter MacIntyre & Rasmus Lerdorf, 2013; Oreilly
- Raharjo Budi 2015. Belajar otodidak framework codeigniter. Informatika.Bandung