Jurus 4:
Kuasai Penyimpanan (storage) Database
Membuat Model Database

Model Database

Dalam sebuah model, kita harus mendefinisikan nama tabel, nama primary key, beserta dengan list kolom-kolomnya dengan tepat. Jika sudah, maka untuk memanggil data pada tabel yang bersangkutan, kita cukup memanggil melalui nama modelnya saja dengan method static ::get().

Untuk membuat database dan tabel ada 2 cara, yaitu:

1. Membuat model data table langsung didatabasenya dengan phpmyadmin
2. Membuat model data dengan migration

Model 2 lebih disarankan jika menggunakan laravel

Membuat Model

```
php artisan make:model berita –migration
```

Kemudian buka folder migrations yang ada di direktori proyekanda/database/migrations.

```php
<?php


use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('berita', function (Blueprint $table) {
            $table->id();
            $table->string('judul');
            $table->string('isi');
            $table->date('tanggal');
            $table->string('kdkategori',10);
            $table->integer('views')->default(0);
            $table->string('gambar');
            $table->string('idpengguna');
            $table->string('author');
            $table->integer('img_slider')->default(0);
            $table->timestamps();
        });
    }
```

```
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('berita');
    }
}
```

Perintah eksekusi migrasi

```
php artisan migrate
```

Hasil eksekusi perintah diatas akan membuat 1 buah file model berita.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class berita extends Model
{
    use HasFactory;
    protected $table = 'berita';

     #kalau kolom primary keynya bernama id, maka baris dibawah ini
     #boleh diisi, dan boleh juga tidak buat
    protected $primaryKey = 'id';

    protected $fillable = [
            'judul',
            'isi',
            'tanggal',
            'kdkategori',
            'views',
            'gambar',
            'idpengguna',
            'author',
            'img_slider'
    ];
}
```

Membuat model yang kedua

```
php artisan make:model kategori –migration
```

edit file migration

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateKategorisTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('kategori', function (Blueprint $table) {
            $table->string('kdkategori',10);
            $table->string('namakategori');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('kategori');
    }
}
```

Perintah eksekusi migrasi

```
php artisan migrate
```

Hasil eksekusi perintah diatas akan membuat 1 buah file model kategori.

Edit file Model Kategori

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class kategori extends Model
{
    use HasFactory;
    protected $table = 'kategori';

    protected $primaryKey = 'kdkategori';
    public $incrementing = false;

    protected $fillable = [
            'kdkategori',
            'namakategori'
    ];


}
```

Model Database Relasi

Bagaimana caranya menampilkan data pada 2 buah tabel yang memiliki relasi? Hal tersebut bisa diselesaikan dengan sangat mudah di Laravel. Sebelum lanjut ke penggunaannya, kita perlu mengenal model relasi database yang paling umum digunakan.

1. Relasi one to one. Relasi dimana 1 data pada sebuah tabel hanya memiliki relasi ke 1 data pada tabel yang lain. Misalnya, 1 data tabel berita memiliki relasi 1 nomor telepon di tabel kategori. Di laravel, kita menggunakan nama method hasOne(...) untuk mendefinisikan hal ini.
2. Relasi one to many. Relasi dimana 1 data pada sebuah tabel memiliki relasi ke beberapa data pada tabel yang lain. Misalnya, 1 data tabel kategori memiliki relasi banyak data barang di berita. Atau dengan kata lain, 1 kategori memiliki banyak data berita. Di laravel, kita menggunakan nama method hasMany(...) untuk mendefinisikan hal ini.
3. Relasi many to one (One to many Inverse). Relasi ini merupakan kebalikannya dari relasi one to many. Misalnya kita ingin mengetahui data berita memiliki kategori apa, maka relasi ini yang akan digunakan. Di laravel, kita akan sering menggunakannya dengan nama method belongsTo(...)
4. Relasi many to many. Relasi dimana banyak data pada sebuah tabel memiliki relasi ke banyak data juga pada tabel yang lainnya. Relasi tersebut terbentuk melalui sebuah tabel bantu. Misalnya, banyak data pada tabel tb_Siswa memiliki relasi peminjaman ke banyak data pada tabel tb_Buku. Relasi tersebut terbentuk dengan tabel bantu bernama tb_Transaksi. Relasi tersebut dapat dilakukan di Laravel dengan method belongsToMany(...).

Untuk mendefinisikan relasi pada sebuah model di Laravel, pertama-tama kita pilih dulu model relasi yang ingin kita buat. Misalnya pada contoh ada 2 buah tabel yaitu tabel berita dan kategori. Apa relasi antar kedua tabel tersebut? Relasinya adalah one to many pada kategori ke berita (karena 1 data kategori bisa memiliki banyak data berita), dan relasi inverse one to many pada berita ke kategori (karena banyak data kategori masing-masing hanya memiliki 1 data kategori). Jika relasi sudah diketahui, sekarang model sudah bisa dibuat. Kita buka kedua file model tersebut, dan kita buat definisinya seperti ini :

```
<?php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class berita extends Model
{
    use HasFactory;
    protected $table = 'berita';

      #kalau kolom primary keynya bernama id, maka baris dibawah ini
      #boleh diisi, dan boleh juga tidak buat
    protected $primaryKey = 'id';

    protected $fillable = [
            'judul',
            'isi',
            'tanggal',
            'kdkategori',
            'views',
            'gambar',
```

```
            'idpengguna',
            'author',
            'img slider'
    ];

    //relasi many to one (Saya adalah anggota dari model ......)
    public function get_kategori(){
        return $this->belongsTo('App\\Model\\kategori', 'kategori', 'id');
    }

}
```

Dalam Model kategori:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class kategori extends Model
{
    use HasFactory;
    protected $table = 'kategori';

    #kalau kolom primary keynya bernama id, maka baris dibawah ini boleh
diisi, dan boleh juga tidak buat
    protected $primaryKey = 'kdkategori';
    public $incrementing = false;

    protected $fillable = [
            'kdkategori',
            'namakategori'
    ];

    public function get_berita(){
        return $this->hasMany('App\\Model\\berita, 'kategori', 'id');
    }

}
```

Pendefinisian relasi di model dibuat dengan membuat sebuah method dengan nama bebas. Nama tersebut nantinya yang akan dipanggil ketika relasi ingin dijalankan. Isinya method tersebut hanya return $this-> dengan nama method relasi yang sesuai dengan kebutuhan. **hasOne()** untuk relasi one to one, **hasMany()** untuk relasi one to many, atau **belongsTo()** untuk relasi inverse one to many. Parameter pertama adalah alamat lengkap class model dimana relasi akan diberikan, parameter kedua adalah nama kolom relasi trigger, dan parameter terakhir (opsional) adalah parameter primary key target relasi dijalankan. Kalau bingung dengan parameternya, pastikan saja dulu alamat model relasinya sudah benar, sisanya kalau error tinggal dituker-tuker aja (pengalaman banget sering ketuker-tuker).

Dari contoh script diatas, Relasi **get_kategori** di model berita akan memanggil data kategori yang sesuai dengan kolom kategori di model kategori, dan relasi **get_berita** akan memanggil data berita apa saja yang ada di kategori bersangkutan. Cara memanggilnya adalah dengan menggunakan method **with('nama_relasi')** saat pemanggilan model. Contoh pemanggilannya dapat kita buat di file HomeController.

```
//sebelumnya :
$beritas = berita::get();


//diganti menjadi :
$ beritas = berita::with('get_kategori')->get();

//nama get_kategori diambil dari nama relasi yang dibuat pada model berita
```

Apabila cara pemanggilannya sudah benar, maka variabel $beritas sudah berisi data berita beserta dengan relasi yang bernama 'get_kategori'. Untuk menampilkannya, kita harus menyebutkan nama relasinya dulu, kemudian diikuti dengan nama kolom relasi yang ingin ditampilkan. Sebagai contoh, saya ingin menampilkan data "nama_kategori" di tabel kategori. Karena relasi sudah terbentuk dengan nama "get_kategori", maka pemanggilan kategori di view berubah menjadi seperti ini :

```
<!--sebelumnya-->
@foreach ($beritas as $row)
     <tr>
          <td>{{ $row->judul }}</td>
          <td>{{ $row->isi }}</td>
          <td>{{ $row->kdkategori }}</td>
     </tr>
@endforeach

<!--sekarang-->
@foreach ($beritas as $row)
     <tr>
          <td>{{ $row->judul }}</td>
          <td>{{ $row->isi }}</td>
          <td>{{ $row->get_kategori->namakategori }}</td>
     </tr>
@endforeach
```