# 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().

## 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 tb\_User memiliki relasi 1 nomor telepon di tabel tb\_Contact. 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 tb\_Category memiliki relasi banyak data barang di tb\_Inventory. Atau dengan kata lain, 1 kategori memiliki banyak data inventory. 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 barang di tb\_Inventory 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 tabel di tutorial sebelumnya](https://tianrosandhy.com/blog/tutorial-dasar-penggunaan-model-view-dan-controller-di-laravel-part-2) ada 2 buah tabel yaitu tabel tb\_Inventory dan tb\_Inventory\_Category. Apa relasi antar kedua tabel tersebut? Relasinya adalah one to many pada tb\_Inventory\_Category ke tb\_Inventory (karena 1 data kategori bisa memiliki banyak data barang), dan relasi inverse one to many pada tb\_Inventory ke tb\_Inventory\_Category (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

#InventoryModel.php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class InventoryModel extends Model

{

protected $table = 'tb\_inventory';

protected $primaryKey = 'id';

protected $fillable = [

'nama\_barang',

'jumlah',

'kategori',

'deskripsi',

'created\_at',

'updated\_at'

];

//relasi many to one (Saya adalah anggota dari model ......)

public function get\_kategori(){

return $this->belongsTo('App\\Model\\InventoryCategoryModel', 'kategori', 'id');

}

}

<?php

#InventoryCategoryModel.php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class InventoryCategoryModel extends Model

{

protected $table = 'tb\_inventory\_category';

protected $fillable = [

'nama\_kategori',

'deskripsi',

'created\_at',

'updated\_at'

];

//relasi one to many (Saya memiliki banyak anggota di model .....)

public function get\_inventory(){

return $this->hasMany('App\\Model\\InventoryModel', '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 InventoryModel akan memanggil data kategori yang sesuai dengan kolom kategori di model InventoryCategoryModel, dan relasi **get\_inventory** akan memanggil data inventory 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 :

$inventory = InventoryModel::get();

//diganti menjadi :

$inventory = InventoryModel::with('get\_kategori')->get();

//nama get\_kategori diambil dari nama relasi yang dibuat pada model InventoryModel

Apabila cara pemanggilannya sudah benar, maka variabel $inventory sudah berisi data inventory 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 tb\_Inventory\_Category. Karena relasi sudah terbentuk dengan nama "get\_kategori", maka pemanggilan kategori di view home.blade.php berubah menjadi seperti ini :

<!--sebelumnya-->

@foreach ($data as $row)

<tr>

<td>{{ $row->nama\_barang }}</td>

<td>{{ $row->jumlah }}</td>

<td>{{ $row-kategori }}</td>

<td>{{ $row->deskripsi }}</td>

<td></td>

</tr>

@endforeach

<!--sekarang-->

@foreach ($data as $row)

<tr>

<td>{{ $row->nama\_barang }}</td>

<td>{{ $row->jumlah }}</td>

<td>{{ $row->get\_kategori->nama\_kategori }}</td>

<td>{{ $row->deskripsi }}</td>

<td></td>

</tr>

@endforeach