LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL XIII

LARAVEL II

Oleh:

(TSAQIF HISYAM SAPUTRA) (23111104024)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

1.1 Dasar Teori

Migration adalah fitur Laravel yang memungkinkan developer untuk mendefinisikan dan memodifikasi struktur database menggunakan kode PHP. Migration berfungsi seperti version control untuk database, memudahkan tim untuk berbagi perubahan skema database.

Database di Laravel dapat diakses menggunakan tiga metode utama:

1. Raw SQL Queries
   - Menggunakan method `DB::insert()`, `DB::select()`, `DB::update()`, `DB::delete()`
   - Query ditulis dalam bentuk SQL murni
   - Cocok untuk query kompleks yang sulit dibuat dengan Query Builder
2. Query Builder
   - Menggunakan `DB::table('nama_tabel')`
   - Menyediakan interface yang fluent untuk membuat query
   - Lebih aman dari SQL injection karena menggunakan parameter binding
3. Eloquent ORM
   - Object-Relational Mapping (ORM) bawaan Laravel
   - Setiap tabel database memiliki Model yang berkorespondensi
   - Menyediakan fitur seperti relationships, mutators, dan accessors

Schema Builder adalah class yang digunakan dalam migration untuk mendefinisikan struktur tabel dengan method seperti:

- `Schema::create()` - Membuat tabel baru
- `Schema::table()` - Memodifikasi tabel yang sudah ada
- `Schema::drop()` - Menghapus tabel

1.2 Tujuan

1. Memahami konsep migration dalam Laravel
2. Mampu membuat file migration untuk mendefinisikan struktur database
3. Memahami perbedaan antara Raw SQL, Query Builder, dan Eloquent ORM
4. Mampu mengimplementasikan operasi insert data menggunakan ketiga metode tersebut
5. Memahami kapan menggunakan masing-masing metode

1.3 Manfaat

1. Dapat mengelola struktur database dengan version control

2. Memudahkan kolaborasi tim dalam pengembangan database
3. Memahami best practice dalam mengakses database di Laravel
4. Dapat memilih metode yang tepat sesuai kebutuhan aplikasi
5. Meningkatkan keamanan aplikasi dari SQL injection

BAB II

HASIL PRAKTIKUM

2.1 TUGAS 12.1 – Migration Database

2.1.1 Membuat File Migration

Perintah untuk membuat migration:

```
bash
php artisan make:migration create_produks_table
```

2.1.2 Struktur Migration

File: `database/migrations/2025_12_24_084621_create_produks_table.php`

```
php
<?php

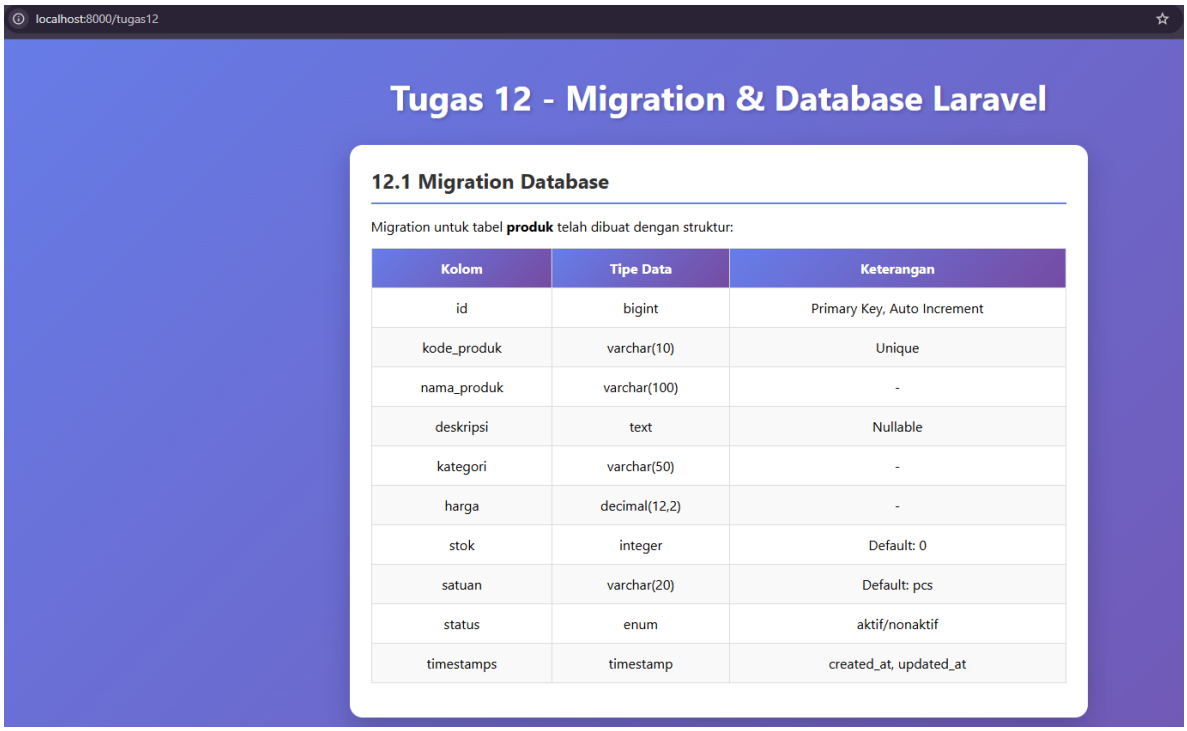
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('produk', function (Blueprint $table) {
            $table->id();
            $table->string('kode_produk', 10)->unique();
            $table->string('nama_produk', 100);
            $table->text('deskripsi')->nullable();
            $table->string('kategori', 50);
            $table->decimal('harga', 12, 2);
            $table->integer('stok')->default(0);
            $table->string('satuan', 20)->default('pcs');
            $table->enum('status', ['aktif', 'nonaktif'])-
>default('aktif');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('produk');
```

```
    }
};
```

2.1.3 Struktur Tabel Produk



2.1.4 Menjalankan Migration

```
bash
php artisan migrate
```

2.2 TUGAS 12.2 – Fungsi Insert Data

2.2.1 Model Eloquent

````
File: `app/Models/Produk.php`

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Produk extends Model
{
    protected $table = 'produk';
````
````

```
    protected $fillable = [
        'kode_produk',
        'nama_produk',
        'deskripsi',
        'kategori',
        'harga',
        'stok',
        'satuan',
        'status',
    ];
}
```

2.2.2 Insert dengan Raw SQL queries

```
php
public function insertRawSQL()
{
    $result = DB::insert(
        "INSERT INTO produk (kode_produk, nama_produk,
deskripsi, kategori, harga, stok, satuan, status, created_at,
updated_at)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW())",
        [
            'PRD001',
            'Laptop ASUS ROG',
            'Laptop gaming dengan spesifikasi tinggi',
            'Elektronik',
            15000000.00,
            10,
            'unit',
            'aktif'
        ]
    );

    return $result; // true jika berhasil
}
```

- **Menggunakan `DB::insert()` dengan query SQL murni**
- **Parameter menggunakan placeholder `?` untuk mencegah SQL injection**
- **Nilai parameter dikirim dalam array terpisah**
- **Return value berupa boolean (true/false)**

2.2.3 Insert dengan Query Builder

```
php
public function insertQueryBuilder()
{
    $result = DB::table('produk')->insert([
        'kode_produk' => 'PRD002',
        'nama_produk' => 'Samsung Galaxy S24',
        'deskripsi' => 'Smartphone flagship Samsung terbaru',
        'kategori' => 'Elektronik',
        'harga' => 12500000.00,
        'stok' => 25,
        'satuan' => 'unit',
        'status' => 'aktif',
        'created_at' => now(),
        'updated_at' => now(),
    ]);

    return $result; // true jika berhasil
}
```

- **Menggunakan `DB::table('nama_tabel')->insert()`**
- **Data dikirim dalam bentuk associative array**
- **Lebih readable dibanding Raw SQL**
- **Otomatis melakukan parameter binding**

2.2.4 Insert dengan Eloquent ORM

```
php
public function insertEloquent()
{
    $produk = Produk::create([
        'kode_produk' => 'PRD003',
        'nama_produk' => 'Nike Air Jordan',
        'deskripsi' => 'Sepatu basket original Nike',
        'kategori' => 'Fashion',
        'harga' => 3500000.00,
        'stok' => 50,
        'satuan' => 'pasang',
        'status' => 'aktif',
    ]);

    return $produk; // Object Model dengan ID
}
```

- **Menggunakan method `Model::create()`**

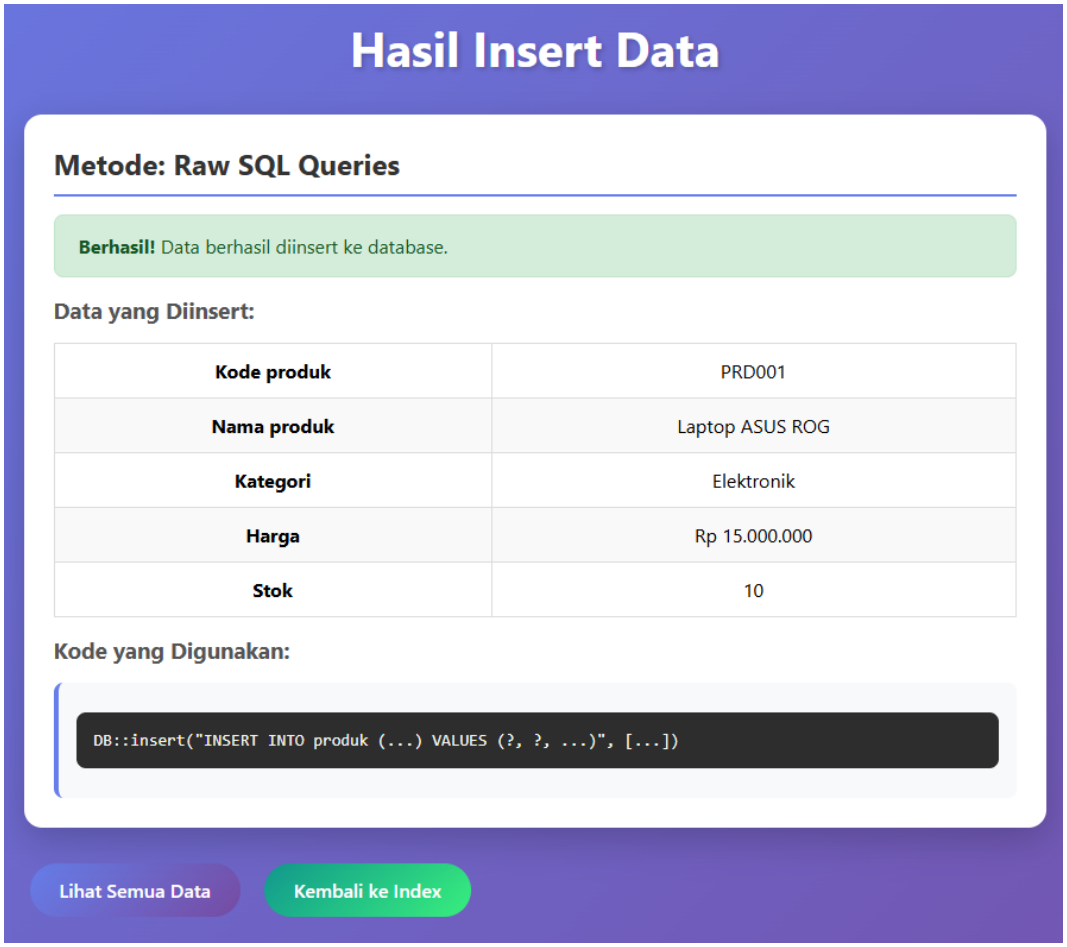
- **Kolom yang boleh diisi harus didefinisikan di `$fillable`**
- **Timestamps (created_at, updated_at) otomatis diisi**
- **Return value berupa object Model dengan semua atribut**

2.3 Perbandingan Metode Insert

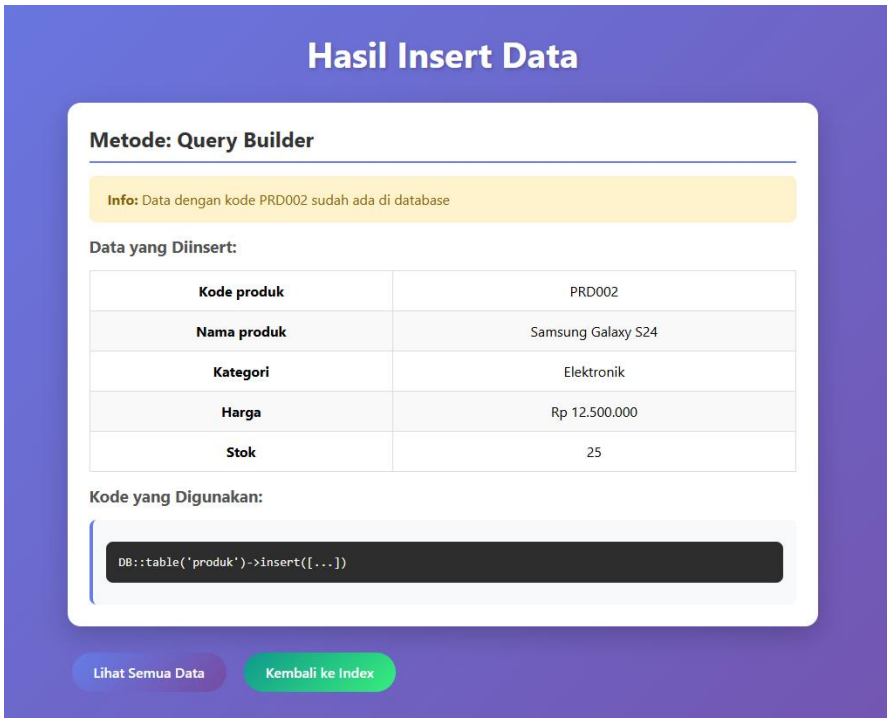
Aspek	Raw SQL	Query Builder	Eloquent ORM
Syntax	SQL murni	Fluen Interface	Object-oriented
Readability	Rendah	Sedang	Tinggi
Keamanan	Manual binding	Auto binding	Auto binding
Timestamps	Manual	Manual	Otomatis
Return Value	Boolean	Boolean	Model Object
Performance	Tercepat	Cepat	Sedikit lebih lambat
Use Case	Query kompleks	Query sederhana-menengah	CRUD standar

2.4 Routes yang dibuat

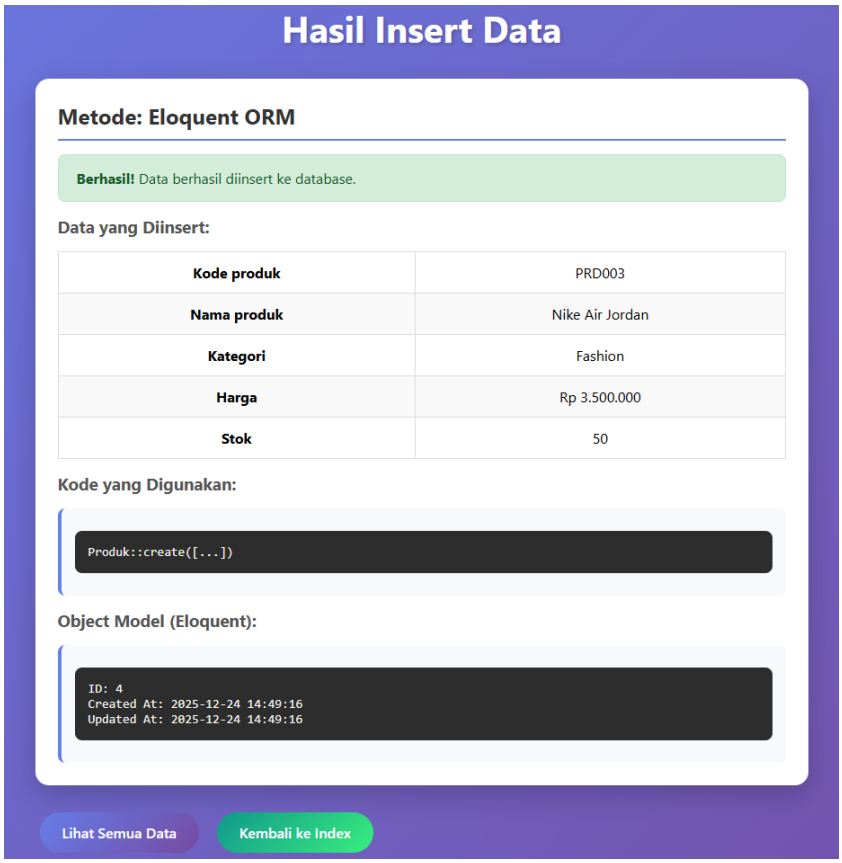
Insert dengan Raw SQL Queries



Insert dengan query builder



Insert Dengan Eloquent ORM



```
php
// routes/web.php
Route::get('/tugas12', [ProdukController::class, 'index'])-
>name('tugas12.index');
Route::get('/tugas12/raw-sql', [ProdukController::class, 'insertRawSQL'])-
>name('tugas12.raw-sql');
Route::get('/tugas12/query-builder', [ProdukController::class,
'insertQueryBuilder'])->name('tugas12.query-builder');
Route::get('/tugas12/eloquent', [ProdukController::class,
'insertEloquent'])->name('tugas12.eloquent');
```

BAB III

KESIMPULAN & SARAN

3.1 Kesimpulan

Pengelolaan database di Laravel difasilitasi oleh fitur Migration yang menjamin struktur skema terorganisir layaknya *version control* dengan kemampuan *rollback*, sementara interaksi data dapat dilakukan melalui tiga metode utama sesuai kebutuhan spesifik. Raw SQL memberikan kontrol penuh untuk performa kritis dan *query* kompleks meski memerlukan penanganan keamanan manual, Query Builder menawarkan keseimbangan antara fleksibilitas dan *readability* yang aman, sedangkan Eloquent ORM menjadi pilihan terbaik untuk operasi CRUD standar karena fitur lengkapnya seperti *timestamp* otomatis, dukungan relasi, dan *return value* berupa objek yang memudahkan pengembangan.

3.2 Saran

Harusnya gunakan Eloquent ORM sebagai pilihan utama untuk operasi database karena lebih aman dan mudah dipelihara, serta optimalkan penggunaannya dengan mempelajari fitur Relationships dan mengimplementasikan Soft Deletes untuk data penting. Selain itu, sangat disarankan untuk menjaga konsistensi *environment* dengan selalu menggunakan Migration untuk setiap perubahan struktur dan Seeder untuk data awal, serta membatasi penggunaan Query Builder atau Raw SQL hanya pada situasi di mana kompleksitas atau kebutuhan performa tidak dapat ditangani oleh Eloquent.