Laravel MongoDB Relation Dilema

Oleh : Hasnul Arief Fikri

25 Mei 2024

Sumber : https://fiona.usu.ac.id/laravel-mongodb-relation-dilema

MongoDB adalah jenis database NoSQL yang berarti setiap barisnya tidak memiliki struktur yang konsisten baik dari jumlah kolom ataupun jenis kolom. Pemanfaatan MongoDB banyak digunakan untuk penyimpanan data yang bersifat masif dan transaksional.

Bagi saya yang programmer laravel dan sudah terbiasa dengan SQL, bermigrasi ke NoSQL mempunyai tantangan tersendiri. Syukurnya saat ini MongoDB sudah memberi dukungan penuh ke laravel https://github.com/mongodb/laravel-mongodb yang dulunya disupport oleh komunitas.

Tantangan implementasi adalah apakah merubah behavior dan struktur pengembangan menjadi full NoSQL atau tetap menggunakan fitur-fitur unggul laravel seperti relation namun membuat MongoDB tetap seperti MySQL. Jujur secara hitung-hitungan saya tidak rela melepas fitur-fitur unggulan laravel. Karena selain pengerjaan cepat, code yang saya bangun juga lebih rapi dan readable.

Masalah terjadi saat kita mendeklarasikan relation dengan MongoDB.

```php
public function items(): HasMany
{
    return $this->hasMany(TagihanItem::class, 'tagihan_id');
}
```

Ini adalah contoh dekalrasi relation HasMany pada dua collection yaitu **Tagihan** dan **TagihanItem** dengan **foreign_key = “tagihan_id”**. Relation ini bekerja sempurna selama tagihan_id menggunakan tipe data **String**.

Di sisi lain MongoDB punya fitur aggregation yang sangat bagus. Dalam fitur aggregation kita dapat melakukan summary, count, average, dengan berbagai kondisi. Selain itu yang juga penting kita bisa melakukan **$lookup** ke collection lain saat aggregation (mirip seperti join di SQL). Namun yang menjadi masalah adalah **foreign_key** harus bertipe data **ObjectId**.

Yap ini dia dilemanya, saat kita performing select data dengan relation kita perlu foreign_key **String** namun saat aggregation kita perlu **ObjectId**. Masalah sudah dapat, lalu apa yang harus dilakukan.

1. mongodb bisa perform aggregation $lookup dengan tipe data **String** namun belum berhasil saya lakukan baik di Laravel ataupun raw query MongoDB
2. MongoDB menyarankan menyimpan foreign_key sebagai ObjectId ketimbang String untuk efisiensi storage
3. Akhirnya saya membuat kesimpulan untuk mengextend package Laravel-MongoDB dan membuat fungsi sendiri. Dengan harapan tidak mengganggu fungsi-fungsi lain yang

telah stable.

```php
public function items(): HasMany
{
    return $this->hasManyObjectId(TagihanItem::class, 'tagihan_id');
}
```

Kalian bisa buat langsung dalam modelnya, atau membuat class baru yang kemudian di-extend oleh model kalian, atau bisa juga seperti saya membuat Trait. Untuk fungsi **hasManyObjectId** saya copypaste sepenuhnya dari fungsi **hasMany.** Tidak ada perubahan yang saya buat karena yang terpenting return-nya yang saya arahkan ke custom class **HasMany**.

```php
<?php

namespace App\Traits;

use App\Helpers\CustomLaravelMongoDB\HasMany;
use MongoDB\Laravel\Eloquent\Model as MongoDBModel;

trait MongoRelationObjectId
{

    public function hasManyObjectId($related, $foreignKey = null, $localKey = null)
    {
        // Check if it is a relation with an original model.
        if (!is_subclass_of($related, MongoDBModel::class)) {
            return parent::hasMany($related, $foreignKey, $localKey);
        }

        $foreignKey = $foreignKey ?: $this->getForeignKey();

        $instance = new $related();

        $localKey = $localKey ?: $this->getKeyName();
        return new HasMany($instance->newQuery(), $this, $foreignKey, $localKey);
    }
}
```

Yang terakhir pada custom class **HasMany** saya mengoverride fungsi **addEagerConstraints** agar setiap keys yang didapat dari foreign dapat diiterasi dan diconvert menjadi **ObjectId** sebelum dieksekusi laravel menggunakan query whereIn

```php
<?php

declare(strict_types=1);

namespace App\Helpers\CustomLaravelMongoDB;

use MongoDB\BSON\ObjectId;
use MongoDB\Laravel\Relations\HasMany as MongoDBHasMany;

class HasMany extends MongoDBHasMany
{
    public function addEagerConstraints(array $models)
    {
        $whereIn = $this->whereInMethod($this->parent, $this->localKey);

        $keys = $this->getKeys($models, $this->localKey);
        $new_keys = [];
        foreach ($keys as $key) {
            $new_keys[] = new ObjectId($key);
        }

        $this->whereInEager(
            $whereIn,
            $this->foreignKey,
            $new_keys,
            $this->getRelationQuery()
        );
    }
}
```

Yak demikian dilema singkat saya dengan tambahan code ini (mudah-mudahan tidak ada major code breaks update) saya bisa membuat relasi kepada kolom string ataupun kolom ObjectId pada Laravel-MongoDB. Kedepannya akan saya tambahkan agar dapat support hasOne, belongTo, belongsToMany, dst. Kedepannya saya harap ada juga yang membuatkan pull request agar apapun tipe datanya tetap bisa kita eksekusi menggunakan satu fungsi.

Wassalam