yii2 Querying Data

Created by 杨超, last modified on 2018 Jan 15

抽时间整理和复习一下yii2的文档读书笔记，基本用法就不赘述了，主要讲一些进阶用法，以前有些模糊的或者没有穿在一起思考的用法

所有内容绝大部分来自：http://www.yiiframework.com/doc-2.0/guide-db-active-record.html#querying-data

概念KeyLists

关键词/概念	说明	好处	坏处
relation		避免重复查询，写法优雅	默认一次request里只查询一次（有缓存），所以如果实时性要求较高或者长事务脚本要注意 如果有需要可以强制刷新
with	1 与joinWith相对 2 eagar load vs [lazy load]	避免多条查询, 提高效率	没有直接的坏处 查询更加隐式，外部参数的传递更加复杂，对使用者有较高要求 比如一个relation需要额外的外部参数传递但是漏传, 会造成语法ok但是逻辑错误，排查的时候比较难发现问题
inverse	1：N hasMany Relation时: 用来使得 N对应的1 relation 和 1 等价【有点绕，就是避免N Model里的 1 relation都指向同一个model，避免重复查询】	避免重复查询	比上面两个更难注意和理解，就是写的时候往往会忽略用inverseOf，但是如果你有一个好习惯 反查audit或者log里的request里的sql查询记录发现重复查询的时候就会想起来

Demos

1 relation 多级relation

order -> orderItems -> purchasedItems 理论上可以无限连下去，不过实际中很少有深度超过3的，因为N^3 你懂的

```
class Customer extends ActiveRecord
{
    public function getPurchasedItems()
    {
        // customer's items, matching 'id' column of `Item` to 'item_id' in OrderItem
        return $this->hasMany(Item::className(), ['id' => 'item_id'])
                    ->via('orderItems');
    }

    public function getOrderItems()
    {
        // customer's order items, matching 'id' column of `Order` to 'order_id' in OrderItem
        return $this->hasMany(OrderItem::className(), ['order_id' => 'id'])
                    ->via('orders');
    }

    public function getOrders()
    {
        // same as above
        return $this->hasMany(Order::className(), ['customer_id' => 'id']);
    }
}
```

2 eagar load vs lazy load

这个太简单了，列表式的查询 里面 含relation的 基本上都是 lazy load, with是eagar load

所以要 逐步给这种查询前面加上with，避免这种大量的sql查询降低接口速度

3 with 的原理

```php
public function findWith($with, &$models)
{
    $primaryModel = reset($models);
    if (!$primaryModel instanceof ActiveRecordInterface) {
        $primaryModel = new $this->modelClass();
    }
    $relations = $this->normalizeRelations($primaryModel, $with);
    /* @var $relation ActiveQuery */
    foreach ($relations as $name => $relation) {
        if ($relation->asArray === null) {
            // inherit asArray from primary query
            $relation->asArray($this->asArray);
        }
        $relation->populateRelation($name, $models);
    }
}
```

简单来说一句话: 就是把参数的里面的name 当做 relation， 然后执行对应的relation生成器将对应的models 查询完放到 findWith的实参 $models里，并且name做为key

这样外部使用这个name的时候就可以直接找到刚刚执行的结果了

有兴趣的可以自己看看源码 大致路径 ActiveQueryTraits::with --> findWith 这里面循环执行每个relation里的 relation对应的生成器 --> ActiveRelationTraits::populateRelation

4 Best partcise

1 多用with来预加载下面用到的relation

2 join的时候尽可能用relation而不是裸写表名

3 方法中尽可能写成relation 而不是 ActiveModels

4 用via来实现多级关联 代码尽可能集中

5 实时性要求高 或者 长脚本 中一定小心用relation

Last!! 测试一定要看log 或者 audit 看看查询语句是否有慢查询/重复查询 以及 是否满足期望