### **什么是ORM**

编程的时候一般采用面向对象的方式，而数据库就是数据库，它可不懂什么是面向对象。

比如我要查一个用户名下的所有文章；

1. $user->posts

这就是面向对象的思维，可数据库可不知道这是什么意思，它只知道我这里有n条数据，要拿哪些数据你自己看着办。

要拿哪些数据，我们自然是知道的，写几个sql查询就拿出来了，可是如果下次又有需求要查询 例如一个用户名下的所有的评论，又或是一个用户名下的所有关注，那我们又得把这个sql的查询逻辑写一遍，可不可以不写这个查询的过程，你直接按照我的方式给我数据，而不是按照你数据库的方式。

ORM就是这样一个东西，它代替你去写SQL，作为一个中介的角色，让面向对象和数据库能够和谐共处。就好比你到商店买东西，售货员会帮你到仓库提货，你只需要告诉她需要什么就可以了。

ORM 不仅可以帮你取数据，增删改查都可以。

实现ORM有两种流行的模式，一种是Active Record，另一种叫做Data Mapper。

Active Record和Data Mapper的区别很简单，看了下面的解释可以秒懂。

### **Active Record 模式**

Laravel和Ruby On Rails框架都是用Active Record 模式来实现ORM的。  
比如Laravel，它有一个Model对象，它继承了一个基类，这个基类就是Active Record的实现，它里面包含了完整的数据库操作方法。  
于是你可以这样写：

1. $user = new User;
2. $user->username = 'Woody';
3. $user->save();

你看，首先$user的属性和数据库的字段是一一对应的，save()这个方法就是模型User继承的基类里来的，它里面包含一整套数据库操作方法。

用过Laravel Eloquent的人都会理解，这种写法是非常简单易懂，也非常符合直觉的。

### **Data Mapper 模式**

大型框架Sympony 采用Doctrine2 ORM引擎，它是一种Data Mapper模式实现的ORM。  
Data Mapper和Active Record的区别其实非常简单，我们看下面的例子：

1. $user = new User;
2. $user->username = 'Woody';
3. EntityManager::persist($user);

我们看到它在存数据的时候，没有用$user->save()去存，而是用了另一个类EntityManager去存。也就是说，它的模型$user没有继承一个数据库操作类，$user和数据库完全是分离开的。

它这样做有什么好处呢？$user没有继承一个庞大的基类，变得轻便多了。就好像蜗牛甩掉了重重的壳，变成了蛞蝓。  
这是反Active Record人士最重要的攻击点，他们会嫌Active Record性能差。  
另外一个就是在Data Mapper模式下，模型，数据操作层，数据库这三者是完全分开的，所以不必一一对应，这样在某些情况下是很灵活的。

### **我该选谁**

显然，第一个发明ORM的人，他最先想到的应该是Data Mapper模式，他使用了一段时间后，发现其实大多数场景是Active Record模式的设定就够了，于是就发明了Active Record模式（这些都是我瞎掰的）。

也就是说，Active Record模式是一种适合大多数场景的简化配置。

但是在某些情况下，你想更充分，更灵活的使用ORM的全部威力，比如别人留给你的一个烂摊子，让你改，这个时候就数据库和模型对象层分开解耦岂不是更好？

Active Record模式简单易用，Data Mapper模式更强大，灵活，高性能，所以更专业，代价就是难用一点。

看个人喜好，哪个其实都可以。