**ORM介绍**

**1.** ORM：Object Relationship Mapping

**2.** 大白话：对象模型与数据库表的映射

**python代码 和SQL代码角度理解：**

**class** Person(object):

name = **'xx'**

age = 18

country =**'xx'**

# Person类 -> 数据库中的一张表

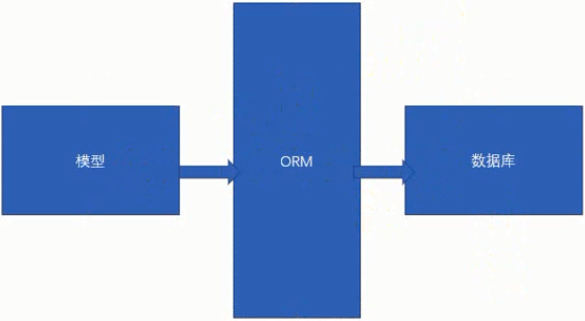
# Person类中的属性 -> 数据库中一张表字段

# Person类的一个对象 -> 数据库中表的一条数据

# p = Person('xx',xx)

# p.save()

#create table person(name varchar(200),age int,country varchar(100))



**SQLAlchemy**

随着项目的越来越大，采用写原生SQL的方式在代码中会出现大量重复的SQL语句，那么，问题，就出现了：

**1.**SQL语句重复利用率不高，越复杂的SQL语句条件越多，代码越长，会出现很多相近的SQL语句。

**2.**很多SQL语句 是在业务逻辑中拼接出来的，如果数据库需要更改，就要去修改这些逻辑，这会很容易漏掉对某些SQL语句的修改。

**3.**写SQL时容易忽略web安全问题，造成隐患。

**ORM**，全称**Object Relational Mapping**,中文名叫做**对象关系映射**，通过ORM我们可以通过类的方式去操作数据库而不用再写原生的SQL语句，通过把表映射成类，把行作为实例，把字段作为属性，ORM在执行对象操作的时候最终还是会把对象的操作转换为数据库的原生语句，但使用ORM有许多优点：

**1.**易用性：使用ORM做数据库开发可以有效减少重复SQL语句的概率，写出来的模型也更加直观、清晰

**2.**性能损耗小：ORM转换成底层数据库操作指令确实会有一些开销。但是从实际情况来看，这种性能损耗很少（不足5%），只要不是针对性能有严苛的要求，综合考虑开发效率、代码阅读性，带来的好处远大于性能损耗，而且项目越大作用越明显。

**3.**设计灵活：可以轻松的写出复杂的查询。

**4.**可移植性：SQLAlchemy封装了底层的数据库实现，支持多个关系数据库引擎，包括流行的Mysql、PostgreSQL和SQLite，可以非常轻松的切换数据库。