ORM

ORM，对象关系映射，`对象`和`关系`之间的映射。这样就可以使用面向对象的方式来操作数据库中的表。

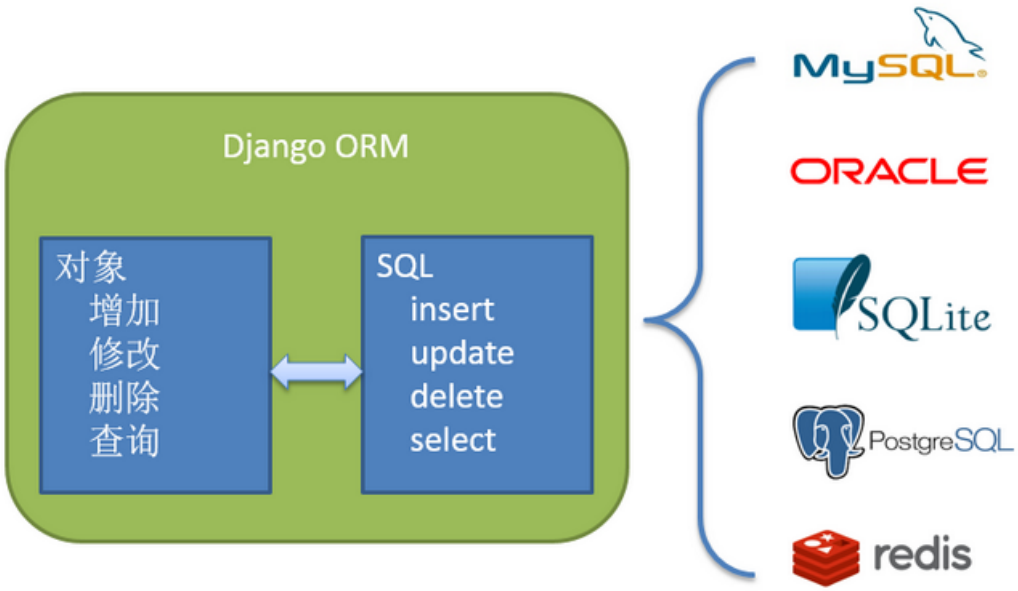
```
关系模型和Python对象之间的映射
table  => class     ，表映射为类
row    => object    ，行映射为实例
column => property  ，字段映射为属性
```

举例，有表student，字段为id int，name varchar，age int
映射到Python为

```
class Student:
    id = ?某类型字段
    name = ?某类型字段
    age = ?某类型字段

最终得到实例
class Student:
    def __init__(self):
        self.id = ?
        self.name = ?
        self.age = ?
```

Django ORM



对模型对象的CRUD，被Django ORM转换成相应的SQL语句以操作不同的数据源。

Model模型

字段类型

字段类	说明
AutoField	自增的整数字段。 如果不指定，django会为模型类自动增加主键字段
BooleanField	布尔值字段，True和False 对应表单控件CheckboxInput
NullBooleanField	比BooleanField多一个null值
CharField	字符串，max_length设定字符长度 对应表单控件TextInput
TextField	大文本字段，一般超过4000个字符使用 对应表单控件Textarea
IntegerField	整数字段
BigIntegerField	更大整数字段，8字节
DecimalField	使用Python的Decimal实例表示十进制浮点数。max_digits总位数，decimal_places小数点后的位数
FloatField	Python的Float实例表示的浮点数
DateField	使用Python的datetime.date实例表示的日期 auto_now=False每次修改对象自动设置为当前时间。 auto_now_add=False对象第一次创建时自动设置为当前时间。 auto_now_add、auto_now、default互斥 对应控件为TextInput，关联了一个js编写的日历控件
TimeField	使用Python的datetime.time实例表示的时间，参数同上
DateTimeField	使用Python的datetime.datetime实例表示的时间，参数同上
FileField	一个上传文件的字段
ImageField	继承了FileField的所有属性和方法，但是对上传的文件进行校验，确保是一个有效的图片
EmailField	能做Email检验，基于CharField，默认max_length=254
GenericIPAddressField	支持IPv4、IPv6检验，缺省对应文本框输入
URLField	能做URL检验，基于基于CharField，默认max_length=200

缺省主键

缺省情况下，Django的每一个Model都有一个名为id的AutoField字段，如下

```
id = models.AutoField(primary_key=True)
```

如果显式定义了主键，这种缺省主键就不会被创建了。

Python之禅中说“显式优于隐式”，所以，如果有必要，还是尽量使用自己定义的主键，哪怕该字段名就是id，也是一种不错的选择。

Django 3.2中增加了`DEFAULT_AUTO_FIELD`对缺省主键字段类型进行设置。

字段选项

参考 https://docs.djangoproject.com/en/3.2/ref/models/fields/#field-options

值	说明
db_column	表中字段的名称。如果未指定，则使用属性名
primary_key	是否主键
unique	是否是唯一键
default	缺省值。这个缺省值**不是**数据库字段的缺省值，而是新对象产生的时候被填入的缺省值
null	表的字段是否可为null，默认为False
blank	Django表单验证中，是否可以不填写，默认为False
db_index	字段是否有索引

关系类型字段类

类	说明
ForeignKey	外键，表示一对多 ForeignKey('production.Manufacturer') 自关联ForeignKey('self')
ManyToManyField	表示多对多
OneToOneField	表示一对一

Model类

- 从基类 django.db.models.Model 派生出与表对应的Model类
- 内部定义`class Meta`，在其类属性上定义
  - db_table = 'employee'，如果不指定，表名为`<appname>_<model_name>`
  - unique_together定义联合唯一键，未来可能过期。建议使用UniqueConstraint
  - ordering=['-pk']定义默认排序规则
- 字段
  - 使用Model类的类属性定义来对应字段
  - 如果不迁移，可以和数据库字段定义的不一致
  - 字段属性
    - primary_key 是否主键，默认False
    - unique 是否唯一键，默认False
    - null 是否可以为null，默认False即必填
    - verbose_name 可视化的名字

- choices 提供枚举值，每一项枚举值是二元组(value, label)，值是存储用，label用来展示。`p.gender`获取字段值value，`p.get_gender_display()`获取label

项目目录DjangoTest下employee应用目录下的models.py，这里就是定义Model类的地方

```
from django.db import models

"""
CREATE TABLE `employees` (
  `emp_no` int(11) NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` smallint(6) NOT NULL DEFAULT '1' COMMENT 'M=1, F=2',
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
"""

class Employee(models.Model):
    class Gender(models.IntegerChoices): # 枚举类型，限定取值范围
        MAN = 1, '男'
        FEMALE = 2, '女'
    class Meta:
        db_table = 'employees'
    # 由于不是自增id主键字段，所以要定义主键
    emp_no = models.IntegerField(primary_key=True, verbose_name='工号') # 主键
    birth_date = models.DateField() # 默认null为False即必填
    first_name = models.CharField(max_length=14, verbose_name='名')
    last_name = models.CharField(max_length=16, verbose_name='姓')
    gender = models.SmallIntegerField(choices=Gender.choices, verbose_name='性别')
    hire_date = models.DateField()

    @property
    def name(self):
        return "{} {}".format(self.last_name, self.first_name)

    def __repr__(self):
        return "<E {}, {}>".format(self.emp_no, self.name)

    __str__ = __repr__
```

在项目根目录编写一个test.py，内容如下

```python
import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'salary.settings')
django.setup(set_prefix=False)

# 所有测试代码，都要在上面4行之下
from employee.models import Employee

emps = Employee.objects.all() # 结果集，本句不发起查询
print(type(emps)) # QuerySet查询集
print(*emps, sep='\n') # __repr__ 所有员工
print(emps[0].gender, emps[0].get_gender_display()) # 枚举类型
```

https://docs.djangoproject.com/en/3.2/ref/models/fields/#django.db.models.Field.choices

管理器

管理器非常重要，有了它才能操作数据库。

每一个非抽象的Model类必须有一个Manager实例。如果不指定，Django会默认指定一个Manager，就是属性objects。

参考 https://docs.djangoproject.com/en/3.2/topics/db/managers/