Django中的reverse relationship是“反向关系”的意思。

有两个Model。

class Group(models.Model):

username = models.CharField(max\_length=200)

password = models.CharField(max\_length=200)

class Profile(models.Model):

group = models.ForeignKey(Group)

email = models.EmailField()

如果有一个Profile对象profile，可以通过profile.group来操作Group对象，对于一个Group对象group，如果你希望去访问和它相关联的Profile对象，django提供了group.profile\_set.all()来访问对应的Profile对象，但这种写法不直观，可以使用related\_name来自定义一个更好的名字，修改Profile的ForeignKey的定义：

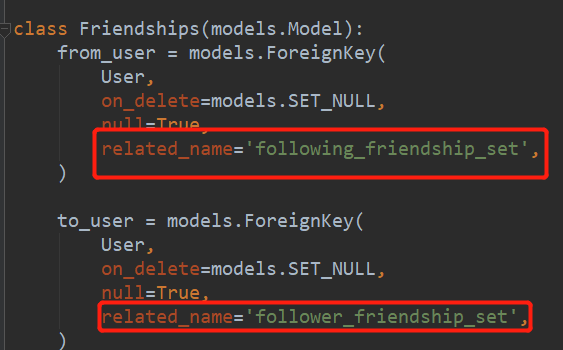
class Profile(models.Model):

group = models.ForeignKey(Group, related\_name="profiles")

email = models.EmailField()

group.profiles.all()

对于Model有多个外键的话，就需要定义related\_name，否则会报错，例如Friendship模型



如果不定义related\_name，则user.friendship\_set无法区分user是from\_user关联的friendship，还是作为to\_user关联的friend\_ship。

ForeignKey和OneToOneField都有这种反向查询的方式，不好的一点是反向查询的方式会访问数据库，所以UserProfile模型定义时，会自定义一个User.profile属性，利用缓存机制来快速访问。