1. **表示外键关系的related\_name属性**。使用related\_name属性可以重命名外键关系。

例如数据库存在Blog表，以及Entry表，Entry表中的Blog 域是外键关系，参照Blog表。默认情况下下我们可以：

b = Blog.objects.get(id=1)

b.entry\_set.all() # 通过b.entry\_set来反向获取所以参照本对象的Entry对象们。

b.entry\_set.filter(head\_line\_\_contains=”Lennon”) # 通过b.entry\_set来反向获取所以参照本对象的Entry对象们。并且还可以进行筛选。

当定义Entry表的Blog外键域时，指定related\_name属性为entries的时候。我么们便会重写xx\_set反向获取的行为。

例如在Entry表定义中blog = ForeignKey(Blog, on\_delete=models.CASCADE, related\_name='entries')

我们便可以：

>>> b.entries.all() # 通过related\_name来反向获取。

>>> b.entries.filter(headline\_\_contains='Lennon') # 通过related\_name来筛选反向获取。

2.

定义表中Meta类中Unique\_together限制，表示所指定域的组合必须在该表中是唯一的。

例如：

class Vote(models.Model):  
 choice = models.ForeignKey(Choice, related\_name="votes", on\_delete=models.CASCADE)  
 poll = models.ForeignKey(Poll, on\_delete=models.CASCADE)  
 voted\_by = models.ForeignKey(User, on\_delete=models.CASCADE)  
  
 class Meta:  
 unique\_together = ("poll", "voted\_by") # 表示在表中, poll, voted\_by的组合在表中记录中必须唯一。实际代表逻辑为一个用户只能在一份调查中投一次票

3.