🡪

values () 返回list of dictionary的结构。列表中每个字典包含key/value对为符合条件的条目的域名和域值。

values\_list() 返回list of tuples形式。每个tuple中包含符合条件的条目的域值。特别的flat=True，只需要某一个域值的时候，可以变为简单的列表形式。

🡪

select\_related()

select\_related()方法是在查询数据的同时将外键域（ForeignKey或者OneToOneField）中的数据预先加载到查询结果当中，当以后使用的该外键域种的数据时，不需要重新查询数据库。加快性能。

例子1：

数据库存在Entry表以及Blog表。

Entry表中的域Blog是外键域，指向Blog表中某一记录。

# 查询数据库

e = Entry.objects.get(id=5)

# 再次查询数据库获取外键域blog

b = e.blog

如果使用select\_related方法：

# 查询数据库

e = Entry.objects.select\_related('blog').get(id=5)

# 无需查询数据库，blog外键已经预先加载到结果集中

b = e.blog

例子2：

# 定义blogs为一个set集合

blogs = set ()

for e in Entry.objects. filter(pub\_date\_\_gt=timezone.now()).select\_related('blog'):

# 如果不使用select\_related则每次循环都需要再次询问数据库

blogs.add(e.blog)

例子3：

假定有数据库定义：

from django.db import models

class City(models.Model):

# ...

pass

class Person(models.Model):

# ...

hometown = models.ForeignKey(

City,

on\_delete=models.SET\_NULL,

blank=True,

null=True,

)

class Book(models.Model):

# ...

author = models.ForeignKey(Person, on\_delete=models.CASCADE)

查询语句b = Book.objects.select\_related('author\_\_hometown').get(id=4)会与预先加载author以及hometown,所以p = b.author和p.hometown都不需要再次访问数据库。

P. s如果filter()和select\_related()方法联用，那么filter()和select\_related()的调用顺序并没有关系。

P.s select\_related方法与get(), filter()返回的对象结果一样，都是指定查询表中对象。只是外键关系预先加载。