Django在进行ORM查询获取queryset对象时，并不是运行代码后就去查询数据库，如：



只有执行了最后一句print时，才会真正去数据库中查找并加载数据。

每个Queryset都带有缓存集\_result\_cache，刚开始都是None。



直接打印缓存\_result\_cache，发现其为None.只有以下7种情况下，才会正式去数据库查询并加载：

1. 迭代，对queryset进行for循环迭代时，会访问数据库，如：

friendships = Friendship.objects.filter(to\_user=user)

return [friendship.from\_user for friendship in friendships]

但切记，不要像上述代码这样写，会产生N次Queries操作。如果有类似的需求，可以采用下面的写法：采用预加载的方式，有两次Queries

friendships = Friendship.objects.filter(to\_user=user,).prefetch\_related('from\_user')

# friendship.from\_user就不会再进行Query查询

return [friendship.from\_user for friendship in friendships]

1. 切片时加入step参数，单纯的切片不会访问数据库，如果User.objects.all()[:6]，但加入step如User.objects.all()[:6:2]就会访问数据库
2. Pickling序列化，使用Python的Pickle模块queryset对象时，会访问数据库。
3. repr()，将对象转化为供解释权读取的形式，方便Python交互式解释器，可以立即查看结果，因此用它操作queryset时，会立刻访问数据库。
4. len()计算queryset的长度时，会访问数据库。
5. list()将queryset转换为列表时，会访问数据库
6. bool判别，如：判断id为2的用户是否存在，会访问数据库

if User.objects.filter(id=2):

pass

如果只是判断是否存在，建议使用exists()访问，性能更快：

if User.objects.filter(id=2).exists():

pass