**Django的ORM常用查询操作总结**

大于、大于等于:

1. \_\_gt 大于>
2. \_\_gte 大于等于>=
3. Student.objects.filter(age\_\_gt=10) // 查询年龄大于10岁的学生
4. Student.objects.filter(age\_\_gte=10) // 查询年龄大于等于10岁的学生
5. 特别注意：这里的下划线是双下划线，下面将介绍的也都是双下划线。

小于、小于等于：

1. \_\_lt 小于<
2. \_\_lte 小于等于<=
3. Student.objects.filter(age\_\_lt=10) // 查询年龄小于10岁的学生
4. Student.objects.filter(age\_\_lte=10) // 查询年龄小于等于10岁的学生

like:

1. \_\_exact 精确等于 like 'aaa'
2. \_\_iexact 精确等于 忽略大小写 ilike 'aaa'
3. \_\_contains 包含 like '%aaa%'
4. \_\_icontains 包含,忽略大小写 ilike '%aaa%'，但是对于sqlite来说，contains的作用效果等同于icontains。

in:

1. \_\_in
2. 查询年龄在某一范围的学生
3. Student.objects.filter(age\_\_in=[10, 20, 30])

is null / is not null：

1. \_\_isnull 判空
2. Student.objects.filter(name\_\_isnull=True) // 查询用户名为空的学生
3. Student.objects.filter(name\_\_isnull=False) // 查询用户名不为空的学生

不等于/不包含于：

1. Student.objects.filter().excute(age=10) // 查询年龄不为10的学生
2. Student.objects.filter().excute(age\_\_in=[10, 20]) // 查询年龄不在 [10, 20] 的学生

其他常用模糊查询：

1. \_\_startswith 以…开头
2. \_\_istartswith 以…开头 忽略大小写
3. \_\_endswith 以…结尾
4. \_\_iendswith 以…结尾，忽略大小写
5. \_\_range 在…范围内
6. \_\_year 日期字段的年份
7. \_\_month 日期字段的月份
8. \_\_day 日期字段的日

多表连接查询：

1. class A(models.Model):
2. name = models.CharField(u'名称')
3. class B(models.Model):
4. aa = models.ForeignKey(A)
6. B.objects.filter(aa\_\_name\_\_contains='searchtitle')#查询B表中外键aa所对应的表中字段name包含searchtitle的B表对象。

**返回新QuerySets的API**

1. 方法名 解释
2. filter() 过滤查询对象。
3. exclude() 排除满足条件的对象
4. annotate() 使用聚合函数
5. order\_by() 对查询集进行排序
6. reverse() 反向排序
7. distinct() 对查询集去重
8. values() 返回包含对象具体值的字典的QuerySet
9. values\_list() 与values()类似，只是返回的是元组而不是字典。
10. dates() 根据日期获取查询集
11. datetimes() 根据时间获取查询集
12. none() 创建空的查询集
13. all() 获取所有的对象
14. union() 并集
15. intersection() 交集
16. difference() 差集
17. select\_related() 附带查询关联对象 一对一，多对一 查询优化
18. prefetch\_related() 预先查询 一对多，多对多 查询优化
19. extra() 附加SQL查询
20. defer() 不加载指定字段
21. only() 只加载指定的字段
22. using() 选择数据库
23. select\_for\_update() 锁住选择的对象，直到事务结束。
24. raw() 接收一个原始的SQL查询

1.filter():

1. filter(\*\*kwargs)
2. 返回满足查询参数的对象集合。
3. 查找的参数（\*\*kwargs）应该满足下文字段查找中的格式。多个参数之间是和AND的关系。
4. Student.objects.filter(age\_\_lt=10)#查询满足年龄小于10岁的所有学生对象

2.exclude():

1. exclude(\*\*kwargs)
3. 返回一个新的QuerySet，它包含不满足给定的查找参数的对象
5. Student.objects.exclude(age\_\_gt=20, name='lin')#排除所有年龄大于20岁且名字为“lin”的学员集

3.annotate():

1. nnotate(args, \*kwargs)
2. 使用提供的聚合表达式查询对象。
3. 表达式可以是简单的值、对模型（或任何关联模型）上的字段的引用或者聚合表达式（平均值、总和等）。
4. annotate()的每个参数都是一个annotation，它将添加到返回的QuerySet每个对象中。
5. 关键字参数指定的Annotation将使用关键字作为Annotation 的别名。 匿名参数的别名将基于聚合函数的名称和模型的字段生成。 只有引用单个字段的聚合表达式才可以使用匿名参数。 其它所有形式都必须用关键字参数。
6. 例如，如果正在操作一个Blog列表，你可能想知道每个Blog有多少Entry：
7. >>> from django.db.models import Count
8. >>> q = Blog.objects.annotate(Count('entry'))
9. # The name of the first blog
10. >>> q[0].name
11. 'Blogasaurus'
12. # The number of entries on the first blog
13. >>> q[0].entry\_\_count
14. 42

4.order\_by():

1. order\_by(\*fields)
3. 默认情况下，根据模型的Meta类中的ordering属性对QuerySet中的对象进行排序
4. Student.objects.filter(school="阳关小学").order\_by('-age', 'name')
5. 上面的结果将按照age降序排序，然后再按照name升序排序。"-age"前面的负号表示降序顺序。 升序是默认的。 要随机排序，使用"?"，如下所示：
6. Student.objects.order\_by('?')

5. reverse():

1. reverse()
2. 反向排序QuerySet中返回的元素。 第二次调用reverse()将恢复到原有的排序。
3. 如要获取QuerySet中最后五个元素，可以这样做：
4. my\_queryset.reverse()[:5]
5. 这与Python直接使用负索引有点不一样。 Django不支持负索引。

6.distinct()：

1. distinct(\*fields)
2. 去除查询结果中重复的行。
3. 默认情况下，QuerySet不会去除重复的行。当查询跨越多张表的数据时，QuerySet可能得到重复的结果，这时候可以使用distinct()进行去重。

7. values()：

1. values(fields, \*expressions)
2. 返回一个包含数据的字典的queryset，而不是模型实例。
3. 每个字典表示一个对象，键对应于模型对象的属性名称。如：
4. # 列表中包含的是Student对象
5. >>> Student.objects.filter(name\_\_startswith='Lin')
6. <QuerySet [<Student: Lin Student>]>
8. # 列表中包含的是数据字典
9. >>> Student.objects.filter(name\_\_startswith='Lin').values()
10. <QuerySet [{'id': 1, 'name': 'Linxiao', 'age': 20}]>
11. 另外该方法接收可选的位置参数\*fields，它指定values()应该限制哪些字段。如果指定字段，每个字典将只包含指定的字段的键/值。如果没有指定字段，每个字典将包含数据库表中所有字段的键和值。如下：
12. >>> Student.objects.filter(name\_\_startswith='Lin').values()
13. <QuerySet [{'id': 1, 'name': 'Linxiao', 'age': 20}]>
15. >>> Blog.objects.values('id', 'name')
16. <QuerySet [{'id': 1, 'name': 'Linxiao'}]>

8.values\_list()：

1. values\_list(\*fields, flat=False)
2. 与values()类似，只是在迭代时返回的是元组而不是字典。每个元组包含传递给values\_list()调用的相应字段或表达式的值，因此第一个项目是第一个字段等。 像这样： >>> Student.objects.values\_list('id', 'name')
3. <QuerySet [(1, 'Linxiao'), ...]>

F对象 / Q 对象 的使用

例如：使所有商品价格减11 Course.objects.update(price=F(‘price’)-11

例如：输出所有py系列并且销量大于等于5000

Course.objects.filter(Q(title\_icontains=’py’)&(volume\_gte=5000))