Django聚合查询和分组查询

聚合查询是指对一个数据表（Model）中某个字段的数据进行部分或者全部统计查询的一种方式，比如所有全部书的平均价格或者是书籍的总数量等等，在这些时候就会使用到聚合查询这种方法。而分组查询同样也属于聚合查询中的一种，只是更加复杂一点而已，在学习本节的知识时候，如果你有较好的 MySQL 知识储备，那么学习本节知识将会变得再简单不过了。下面就让我们开始学习吧。

1. 聚合查询的应用

1) 不带分组的聚合查询

对数据表计算统计值，需要使用 aggregate 方法，提供的参数可以是一个或者多个聚合函数，aggregate 是 QuerySet 的一个子句，它的返回值是一个字典类型，键是聚合的关键字，值是聚合后的统计结果。

不带分组的聚合查询是指对将全部数据进行集中统计查询，Django 定义了一些常用的聚合函数，比如求平均值（Avg）、计数（Count）、求最值（Max和Min）以 Sum 求和。它们统一定义在 django.db.models 模块中，所以再使用聚合函数时，同样需要提前导入，为了方便使用，我们采用下面的方式引入：

from django.db.models import *

它的语法格式如下所示，它的返回值是一个字典，以统计结果变量名为 key，以统计值为 value：

MyModel.objects.aggregate(统计结果变量名=聚合函数('列名'))

我们通过求所有书籍的价格平均值与所书籍数量来进行实例演练：

```
#求所有书籍的平均价格
from index.models import Book
from django.db.models import *
result =Book.objects.aggregate(myAvg=Avg('price'))
print("平均价格是:", result['myAvg'])
print("result=", result)
#result= {'myAvg': Decimal('47.800000')}
#求一共有多少本书
result =Book.objects.aggregate(MyCulate=Count('title'))
print("数据记录总个数是:", result['MyCulate'])
print("result=", result)
#result= {'MyCulate': 5}
#传递多个聚合函数一起求值
result=Book.objects.aggregate(l=Min("price"),m=Max("price"),n=Avg("retail_price"))
print("result=",result)
#result= {'l': Decimal('25.00'), 'm': Decimal('65.00'), 'n': Decimal('127.800000')}
```

2. 聚合分组查询应用

分组聚合是指通过计算查询结果中每一个对象所关联的对象集合，从而得出总计值(也可以是平均值或总和)，即为查询集的每一项生成聚合。简单的理解就是对 QuerySet中的每一个 Model 对象都生成一个统计值。分组聚合使用 annotate 方法完成。它的语法格式和聚合查询一样如下所示：

QuerySet.annotate(结果变量名=聚合函数('列名'))

分组聚合的实现主要两个步骤：首先使用 MyModel.objects.values 获得要分组聚合的列，它的返回结果是一个 QuerySet 类型的字典，然后通过 QuerySet.annotate(变量名=聚合函数('列名')) 的方法分组聚合得到相应的结果。下面我们通过实例进行说明，通过分组聚合查询获取价格相同的书籍数量：

```
#在index/views.py 添加代码
from django.db.models import Count
from index.models import Book,PubName
def test_annotate(request):
    # 得到所有出版社的查询集合QuerySet
    bk_set = Book.objects.values('price')
    # bk=Book.objects.get(id=1)
    # print('书名:',bk.title,'出版社是:',bk.pub.pubname)
    # 根据出版社QuerySet查询分组, 出版社和Count的分组聚合查询集合
    bk_count_set = bk_set.annotate(myCount=Count('price'))  # 返回查询集合
```

```
    for item in bk_count_set: #通过外键关联进行查询bk_set.pub.pubname
        print("价格是:", item['price'], "同等价格书籍数量：", item['myCount'])
    return HttpResponse('请在CMD命令行控制台查看结果')
#路由配置为忘记:path('annotate/',views.test_annotate)
```

最终在CMD命令行会得到如下输出：

价格是: 59.00 同等价格书籍数量： 1
价格是: 25.00 同等价格书籍数量： 1
价格是: 45.00 同等价格书籍数量： 2
价格是: 65.00 同等价格书籍数量： 1

由于 annotate 的返回值是一个 QuerySet 对象，所以我可以通过 query 属性查看其执行的 SQL 语句，如下所示：

```
SELECT `index_book`.`price`, COUNT(`index_book`.`price`) AS `myCount` FROM `index_book` GROUP BY `index_book`.`price` ORDER BY NULL
```

从 SQL 语句可以得出，annotate 按照 price 分组，并且在 SELECT 中对 price 进行了计数。默认情况下，annotate 会对每一个 Model 对象计算统计值。但是，如果使用了 values 方法中指定的字段，Django 会先按照该字段对 Model 对象进行分组，再去对每个分组计算统计值。

3. 总结归纳

本节主要讲解了聚合查询以及分组聚合查询的使用方法，还给大家介绍了几个常用的聚合函数。聚合查询和分组查询分别调用不同的方法来实现，聚合查询是 aggregate，而分组聚合查询是 annotate。后者经常配合 values 方法来选取要分组的字段。

在这里大家还要注意，annotate 和 values 方法的顺序非常重要，会影响实际的查询效果。上面实例就是 values 在前的情况，如果 annotate 在前会产生不同的影响，如下所示：

```
In [41]: Book.objects.annotate(t=Max('price')).values('id','t')
Out[41]: <QuerySet [{'id': 1, 't': Decimal('59.00')}, {'id': 2, 't': Decimal('25.00')}, {'id': 3, 't': Decimal('45.00')}, {'id': 4, 't': Decimal('65.00')}, {'id': 5, 't': Decimal('45.00')}]>#按照values提供的参数分别作为
```

至此我们将 Django 提供的各种查询方式讲解完毕，我们用了 7 节的内容对此知识点做了详细介绍。熟练掌握这些章节的内容，对我们从数据库获取数据的操作会有很大帮助，也能够满足一些实际业务上需求。