# Meta类属性汇总：

class Meta:

abstract = … : True or False, 标识本类是否为抽象基类

app\_label = … : 定义本类所属的应用，比如 app\_label=’myapp’

db\_table = … : 映射的数据表名， 比如 db\_table = ‘comments’

默认为app\_comments

db\_tablespace= …:映射的表空间名称。

default\_related\_name = …: 定义本模型的反向关系引用名称，默认与模型名一致。

get\_latest\_by = … : 定义按哪个字段排列以获得模型的开始和结束的记录，本属性通常指向一个日期或整数的模型字段

managed = … : True or False, 定义为Django的manage.py 命令行工具是否管理本模型。默认为True；若设为False，则运行python manage.py migrate 时将不会在数据库中生成本模型的数据表，所以需要手工维护数据库的定义。

order\_with\_respect\_to=… : 定义本模型可以按照某外键引用的关系排序。

ordering=… :本模型记录的默认排序字段，可以设置多个字段，默认以升序排列，如果以降序排列则需要在字段名前加“负号”。

class Meta:

ordering = [‘user\_name’, ‘-pub\_date’]

按user\_name升序和pub\_date降序排列

default\_permissions=…:模型的操作权限

默认为default\_permissions=(‘add’, ‘change’, ‘delete’)

proxy: True or False, 本模型及继承自本模型的子模型是否为代理模型

required\_db\_features:

required\_db\_vendor:

unique\_together:用来设置不重复的字段组合，必须唯一（可以将多个字段做联合唯一）

class Meta:

unique\_together = ((“user\_name”, “pub\_date”),)

定义每个user\_name在同一个pub\_date中只能有一条数据表记录

可以设置多个这样的唯一约束

index\_together:定义联合索引的字段，可以设置多个。

class Meta:

index\_together = [[“pub\_date”, “deadline”],]

verbose\_name:

verbose\_name\_plural:

# 常用字段参数

null: 默认为False，定义是否允许相应的数据库字段为null  
blank: 定义字段是否为空。

null与blank的区别：

null是一个数据库中的非空约束；

blank用于字段的HTML表单验证，即判断用户是否可以不输入数据

choices：定义字段为可选值

LEVEL = (

(‘1’, ‘Very good’),

(‘2’, ‘Good’),

)

在数据库中实际存储的是第1个值，而在HTML页面中显示的是第2个值

default: 设定默认值

help\_text: HTML页面中输入控件的帮助字符串

primary\_key: 定义字段是否为主键， 为True或false

unique: 是否为字段定义数据库的唯一约束。

on\_delete=models.CASCADE 对 OneToOneField和ForeignKey两字段有效。当被关联模型的记录被删除时，本模型的记录也被删除。

class Contact:

account = OneToOneField(“Account”, on\_delete=models.CASCADE)

….

a1 = Account(…)

a1.save()

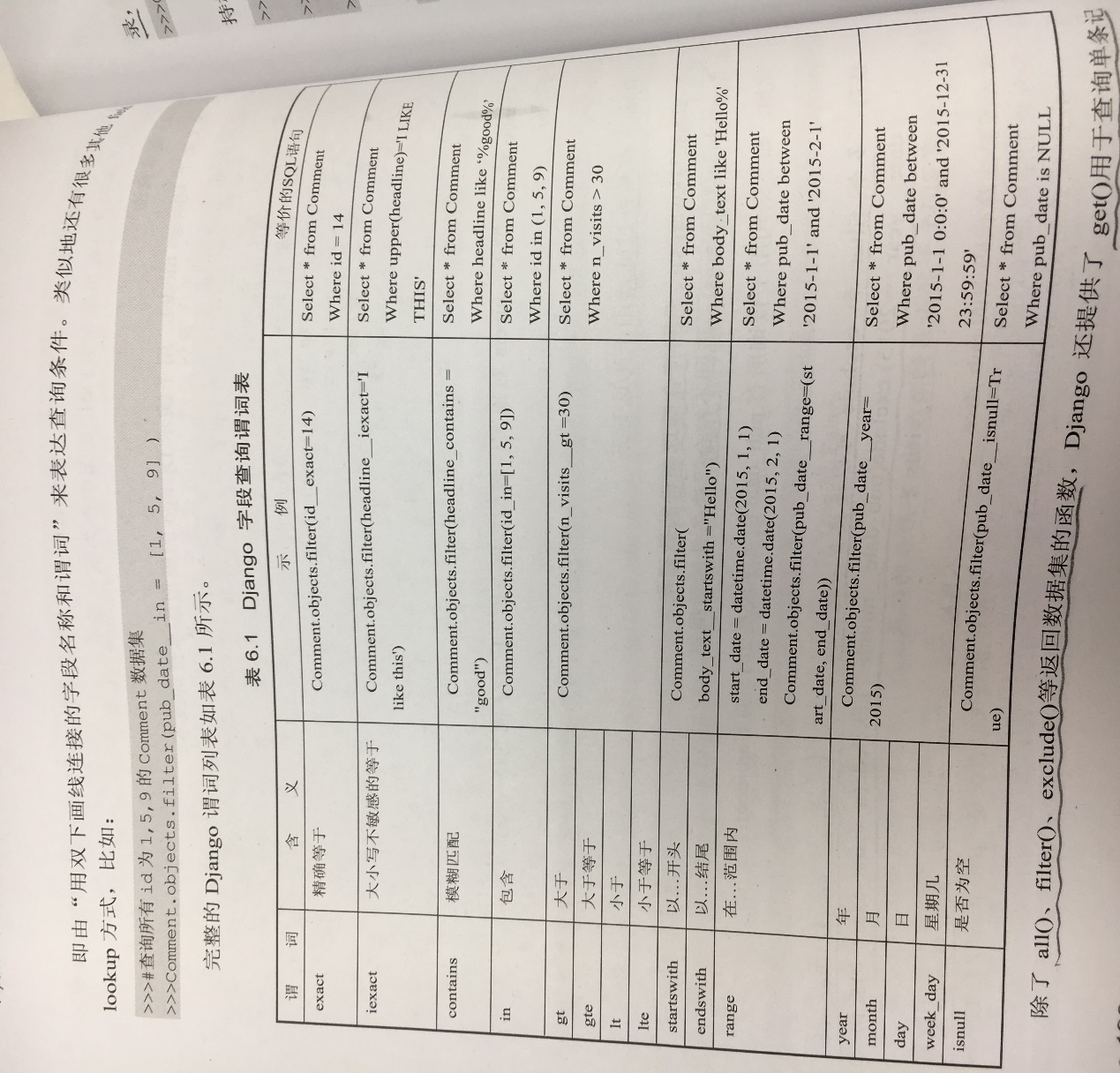
c1 = Contact(account=a1)

c1.save()

a1.delete()#此时a1与c1同时被删除

ForeignKey同理

# Django字段查询谓词表



Model.objects.get(…)

Model.objects.filter(…)

Model.objects.all(…)

Model.objects.order\_by(…)

Model.objects.exclude(…)

关系操作

1. 一对一关系：OneToOneField

account = models.OneToOneField(“Account”, on\_delete=models.CASCADE,primary\_key=True)

on\_delete:参数定义，当被关联模型Account的记录被删除时，本模型的记录如何处理

models.CASCADE:用于定义本记录也被删除

a1=Account(..)

a1.save()

c1=Contatct(account=a1,…)

c1.save()

a1.contact 等价于 c1

c1.account 等价于 a1

1. 一对多关系：ForeignKey

class Contact：

account = ForeignKey(‘Account’, on\_delete=models.CASCADE)

a1=Account(..)

a1.save()

c1=Contatct(account=a1,…)

c1.save()

c2=Contatct(account=a1,…)

c2.save()

c1.account 与 c2.account 等价于 a1

a1.contact\_set.all() 返回c1与c2

a1.contact\_set.count() 返回2

a1.delete() c1与c2同时删除

1. 多对多关系：ManyToManyField

class Contact：

accounts = ManyToManyField(‘Account’)

a1 = Account()

a1.save()

c1 = Contact(….)

c1.save()

c1.accounts.add(a1)

a2 = Account()

a2.save()

a2.contact\_set.add(c1)

# 面向对象ORM

1. 抽象类继承

在多个表有若干相同的字段时，可以使开发者将这些字段统一定义在抽象基类中。数据库中不会有MessageBase表

class MessageBase(models.Model):

….

class Meta:

abstract = True

class Moments(MessageBase):

….

1. 多表继承

class MessageBase(models.Model):

….

class Moments(MessageBase):

….

1. 代理模型继承

继承子类只用于管理父类的数据，而不实际存储数据。

通过在子类的Meta中定义proxy=True属性来实现。

# URL映射

URL路径用正则表达式定义

‘^’: 以。。。为开始

‘$’: 以。。。为结束

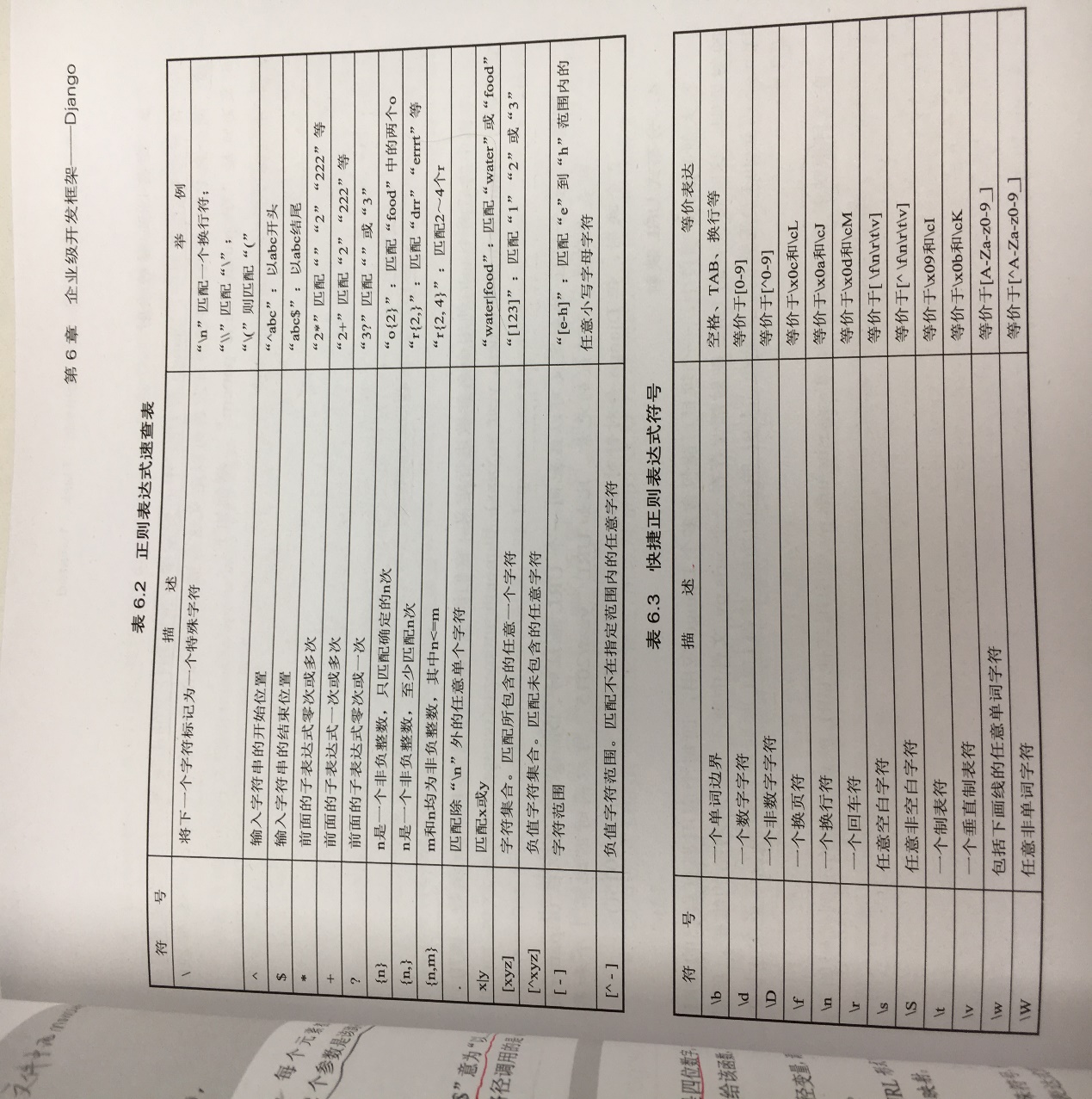
Eg: url(r’^/year/2015/$’, views.moments\_2015)

url(r’^/year/[0-9]{4}/$’, views.year\_moments) 匹配路径是“year/xxxx/”，其中要求xxxx是4位数字，并且把四位数字作为变量传递给该函数调用参数形式是year\_moments(request,xxxx)

url(r’^/year/[0-9]{4}/[0-9]{2}/$’, views.month\_moments) : month\_moments(request, xxxx, yy)

url(r’^/year/[0-9]{4}/[0-9]{2}/([0-9]+)$’, views.single) : single(request, xxxx, yy, zz)

正则表达式



命名URL参数映射

“?P<param\_name>pattern”

Eg: url(r‘^year/?P<year>([0-9]{4})/$’, views.year\_moments) : year\_moments(request, year=xxxx)

分布式URL映射

include 的使用

eg: url(r’^moments/’, include(‘djangosite.app.urls’))

以moments开头的URL被转接到djangosite.app.urls包中