HINTS

EXCLUSIÓN DE CAMPOS DEL MODELO

Búsquedas en el índice: `raw()` admite indexación, por lo que si solo necesita el primer resultado, puede escribir:

```
first_person = Person.objects.raw('SELECT * FROM myapp_person')[0]
```

Sin embargo, la indexación y el corte no se realizan a nivel de base de datos. Si tiene una gran cantidad de objetos Person en su base de datos, es más eficiente limitar la consulta en el nivel SQL:

```
first_person = Person.objects.raw('SELECT * FROM myapp_person LIMIT
1')[0]
```

Aplazamiento de los campos modelo: éstos también pueden ser dejados fuera:

```
people = Person.objects.raw('SELECT id, first_name FROM myapp_person')
```

Los objetos `Person` devueltos por esta consulta serán instancias de modelo diferidas. Esto significa que los campos que se omiten de la consulta se cargarán a pedido. Por ejemplo:

```
>>> for p in Person.objects.raw('SELECT id, first_name FROM
myapp_person'):
...     print(p.first_name, # Esto será recuperado por la consulta
original
...           p.last_name) # Esto se recuperará a pedido
...
John Smith
Jane Jones
```

Desde el exterior, parece que la consulta ha recuperado tanto el nombre como el apellido. Sin embargo, este ejemplo en realidad emitió 3 consultas. Solo los nombres se recuperaron mediante la consulta `raw()`; mientras que los apellidos se recuperaron a pedido cuando se imprimieron.

Solo hay un campo que no puede omitir: el de clave principal. Django usa la clave principal para identificar instancias del modelo, por lo que siempre debe incluirse en una consulta sin formato. Se mostrará la excepción `FieldDoesNotExist` si olvida incluir la clave principal.

Si se desea realizar una consulta in situ en Django, puede hacerlo fácilmente usando `raw()`:

```
import django.db from models
class Users(models.Model):
    name = models.CharField()
    age = models.IntegerField()
    sex = models.CharField()
```

Especificar SQL para obtener datos.

Cuando la edad de la tabla de usuarios es superior a 40 usuarios:

```
sql = "SELECT * FROM users WHERE age >= 40"
users = Users.objects.raw(sql)
```

Los resultados son los mismos que los siguientes:

```
users = Users.objects.filter(age__gte=40)
```

ESPECIFICAR PARÁMETROS

También puede especificar parámetros para ejecutar:

```
sql = "SELECT * FROM users WHERE age >= %s"
users = Users.objects.raw(sql, [40])
```

Cuando se especifica con el tipo de diccionario, ¿cuál es el primer valor cuando hay más de un parámetro? Esto es difícil de entender, por lo que se recomienda lo siguiente:

```
sql = "SELECT * FROM users WHERE age >= %(age)s AND sex = %(sex)s"
params = {"age": 40, "sex": "male"}
users = Users.objects.raw(sql, params)
```