**Тема занятия № 29: Модуль 16. Модели: расширенные инструменты**

1. Управление выборкой полей

Модели Django предоставляют ряд расширенных инструментов: средства для управления выборкой полей, связи с дополнительными параметрами, полиморфные связи, наследование моделей, объявление своих диспетчеров записей и наборов записей и инструменты для управления транзакциями.

**Управление выборкой полей**

При выборке набора записей Django извлекает из таблицы значения полей только текущей модели. При обращении к полю связанной модели фреймворк выполняет дополнительный SQL-запрос для извлечения содержимого этого поля, что может снизить производительность.

Помимо этого, выполняется выборка значений из всех полей текущей модели. Если какие-то поля хранят данные большого объема (например, большой текст), их выборка займет много времени и отнимет существенный объем оперативной памяти.

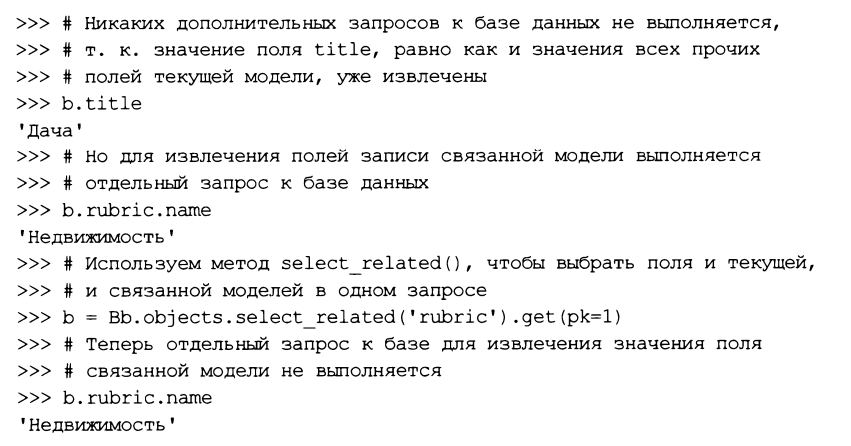
Далее приведены методы, поддерживаемые диспетчером записей (классом Manager) и набором записей (классом queryset), которые позволяют управлять выборкой значений полей:

О select\_related (<поле внешнего ключа 1>, <поле внешнего ключа 2> . . . <поле внешнего ключа п>) — будучи вызван у записи вторичной модели, указывает извлечь связанную запись первичной модели. В качестве параметров записываются имена полей внешних ключей, устанавливающих связь с нужными первичными моделями.

Метод извлекает единичную связанную запись. Его можно применять только в моделях, связанных связью "один-с-одним", и вторичных моделях в случае связи "один-со-многими".

Пример:



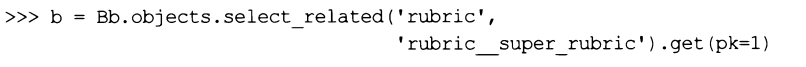


Применяя подобный синтаксис, можно выполнить выборку модели, связанной с первичной моделью. Предположим, что модель вь связана с моделью Rubric, которая, В СВОЮ очередь, связана с моделью superrubric через поле внешнего ключа super rubric и является вторичной для этой модели. Тогда МЫ можем ВЫПОЛНИТЬ выборку связанной записи модели superrubric, написав выражение вида:

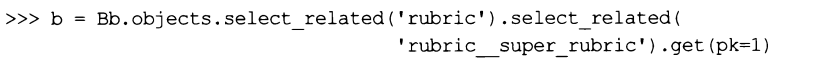


Можно выполнять выборку сразу нескольких связанных моделей, написав такой

Код:



Или такой:

****

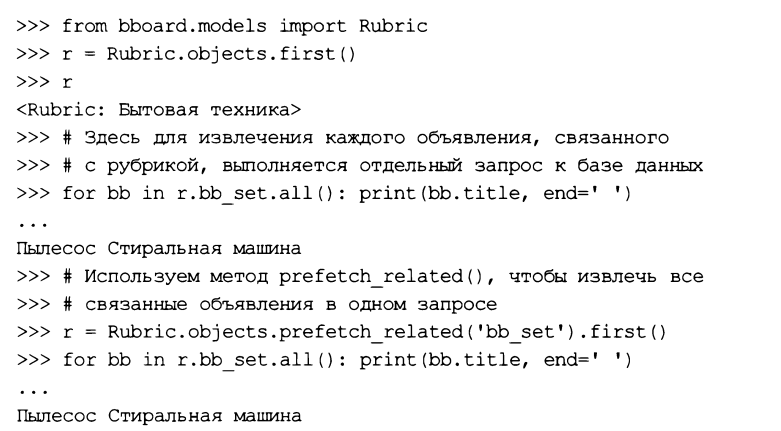
Чтобы отменить выборку связанных записей, заданную предыдущими вызовами метода seiect\_reiated(), достаточно вызвать этот метод, передав в качестве параметра значение None;

□ prefetch\_related (<связь 1>, <связь 2> . . . связь п>) —будучи вызван у записи первичной модели, указывает извлечь все связанные записи вторичной модели.

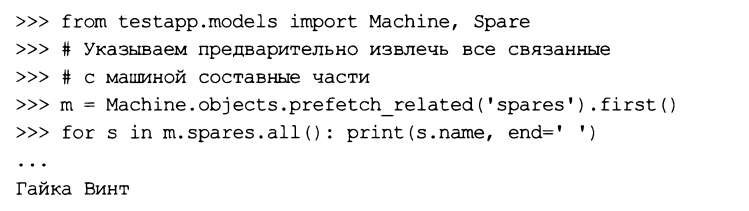
Метод извлекает набор связанных записей. Он применяется в моделях, связанных связью "многие-со-многими", и первичных моделях в случае связи "один со-многими”.

В качестве связи можно указать:

• строку с именем: атрибута, применяющегося для извлечения связанных записей — если метод вызывается у записи первичной модели, и установлена связь "один-со-многими":

****

D поля внешнего ключа — если установлена связь "многие-со-многими":

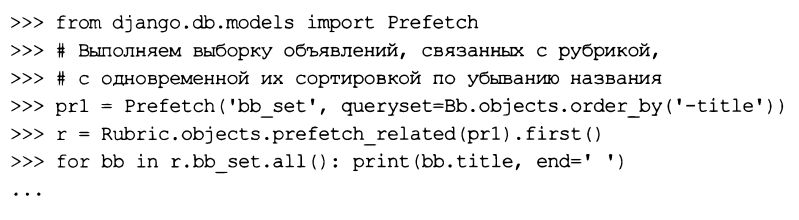
****

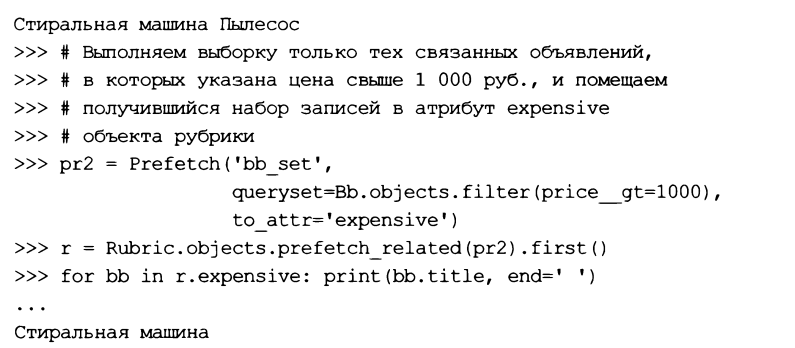
Экземпляр класса Prefetch из модуля django.db.models, хранящий все необходимые сведения для выборки записей. Конструктор этого класса вызывается в формате: Prefetch(<связь>[, queryset=None][, to\_attr=None]) связь указывается точно так же, как было описано ранее.

Необязательный параметр queryset задает набор записей для выборки связанных записей. В этом наборе записей можно указать какую-либо фильтрацию, сортировку или предварительную выборку полей связанных записей методом seiect reiatedo (см. Ранее).

Необязательный параметр to attr позволяет указать имя атрибута, который будет создан в объекте текущей записи и сохранит набор выбранных записей связанной модели.

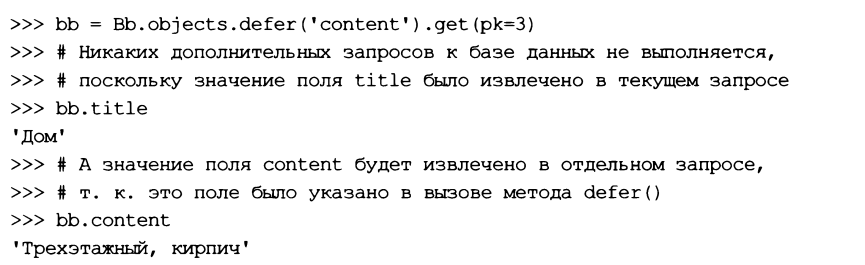
Примеры:





Можно выполнить выборку наборов записей по нескольким связям, записав их либо в одном, либо в нескольких вызовах метода prefetch\_related(). Чтобы отменить выборку наборов записей, заданную предыдущими вызовами метода Prefetch reiated (), следует вызвать этот метод, передав ему в качестве параметра значение None;

О defer(<имя поля 1>, <имя поля 2> . . . <имя поля п>) —указывает не извлекать значения полей с заданными именами в текущем запросе. Для последующего извлечения значений этих полей будет выполнен отдельный запрос к базе данных. Пример:

****

Можно указать не выполнять выборку значений сразу у нескольких полей, записав их либо в одном, либо в нескольких вызовах метода defer о. Чтобы отменить запрет выборки, заданный предыдущими вызовами метода defer (), Следует вызвать этот метод, передав ему в качестве параметра значение None;

□ only (<имя поля 1>, <имя поля 2> . . . <имя поля п>) —указывает не извлекать значения всех полей, кроме полей с заданными именами, в текущем запросе. Для последующего извлечения значений полей, не указанных в вызове метода, будет выполнен отдельный запрос к базе данных. Пример:



Вызов метода only о отменяет параметры выборки, заданные предыдущими вызовами методов only о и defer о. Однако после его вызова можно поставить вызов метода defer о — он укажет поля, которые не должны выбираться в текущем запросе.

1. Связи "многие-со-многими" с дополнительными данными

Ранее мы написали модели Machine (машина) и spare (отдельная деталь), связанные связью ”многие-со-многими”. Однако совершенно забыли о том, что машина может включать в себя более одной детали каждого наименования, и нам нужно где-то хранить количество деталей, входящих в состав каждой машины.

Реализовать это на практике можно, создав связь с дополнительными данными.

Делается это в два шага:

1. Объявить связующую модель, которая, во-первых, создаст связь ’’многие-со-многими” между ведущей и ведомой моделями, а во-вторых, сохранит дополнительные данные этой связи (в нашем случае — количество деталей, входящих в состав машины).

В связующей модели должны присутствовать:

• поле типа foreignkey для связи с ведущей моделью;

• поле типа foreignkey для связи с ведомой моделью;

• поля нужных типов для хранения дополнительных данных.

2. Объявить в ведущей модели поле типа manytomanyfieid для связи с ведомой моделью. В параметре through этого поля следует указать имя связующей модели, представленное в виде строки, а в параметре through\_fieids — кортеж из двух элементов:

• имени поля связующей модели, по которому устанавливается связь с ведущей моделью;

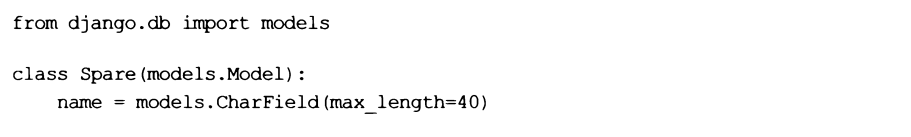
• имени поля связующей модели, по которому устанавливается связь с ведомой моделью.

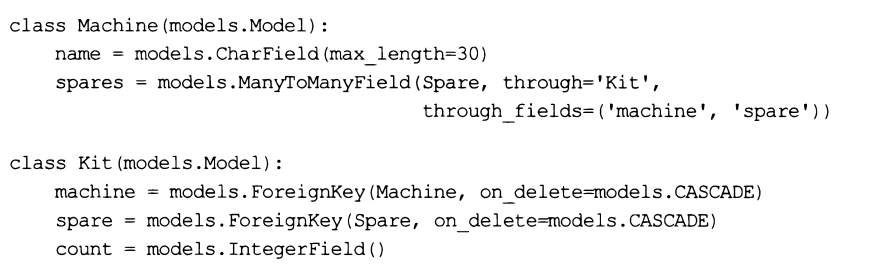
На заметку!

Вообще-то параметр through\_fieids обязательно указывается только в том случае, если связующая модель связана с ведущей или ведомой несколькими связями, и, соответственно, в ней присутствуют несколько полей внешнего ключа, устанавливающих эти связи. Но знать о нем все равно полезно.

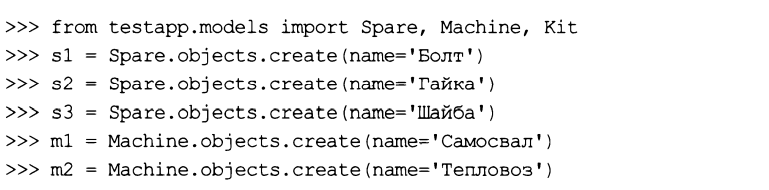
Ведомая модель объявляется так же, как и в случае обычной связи "многие-со-многими”.

Код трех моделей: ведомой spare, ведущей Machine и связующей Kit.





Написав классы моделей, сформировав и выполнив миграции, мы можем работать с данными с применением способов, хорошо знакомых нам по занятиям ранее. Сначала мы создадим записи в моделях Spare И Machine:



Связи мы можем создавать следующими способами:

□ напрямую— создавая записи непосредственно в связующей модели. Добавим в состав самосвала 10 болтов (сообщения, выводимые консолью, пропущены ради краткости):



□ методом add о, (начиная с Django 2.2),— добавив в него параметр through\_defaults и присвоив ему значения полей записи, создаваемой в связующей модели. Значением этого параметра должен быть словарь, элементы которого соответствуют полям связующей модели, а значения зададут значения для этих полей.

Для примера добавим в состав самосвала 100 гаек:



□ методом create о (начиная с Django 2.2),— добавив

В него аналогичный параметр through\_defauits.

В качестве примера создадим новую деталь — шпильку — и добавим две таковых в самосвал:



□ методом set о (начиная с Django 2.2), — вставив в его вызов аналогичный параметр through defauits. добавим в состав тепловоза 49 шайб и столько же болтов:

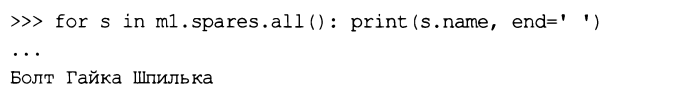


Внимание!

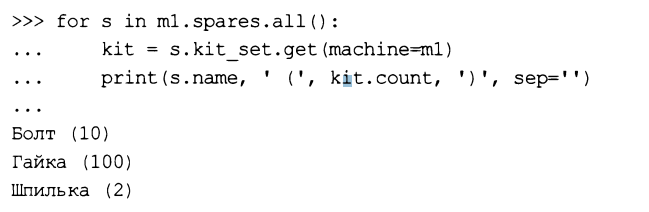
Значения, заданные в параметре through defaults метода set о, будут сохранены во всех записях, что создаются в связующей модели. Задать отдельные значения для отдельных записей связующей модели, к сожалению, нельзя.

При создании связей "многие-со-многими" вызовом метода set() настоятельно рекомендуется указывать в нем параметр clear со значением True. Это необходимо для гарантированного обновления значений полей в записях связующей модели.

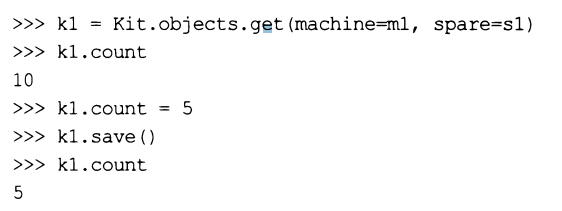
Проверим, какие детали содержит тепловоз:



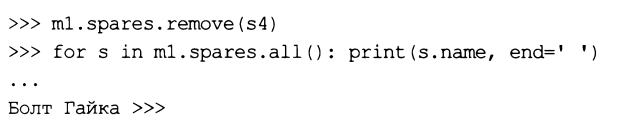
Выведем список деталей, которые содержит самосвал, с указанием их количества:

****

Уменьшаем количество входящих в состав самосвала болтов до пяти:

****

Для удаления связей можно пользоваться методами remove!) И clear о, для примера удалим из самосвала все шпильки:

****