**Тема занятия № 33: Модуль 18. Поддержка баз данных postgresql**

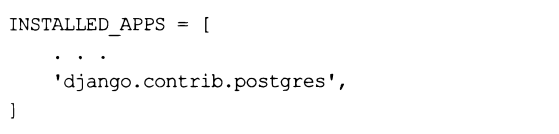
1. Дополнительные инструменты для поддержки PostgreSQL

Django предоставляет расширенные средства для работы с СУБД postgresql.

Помимо этого, существует библиотека django-localflavor, предоставляющая дополнительные поля моделей, форм и элементы управления.

**Дополнительные инструменты для поддержки postgresql**

Эти инструменты реализованы в приложении django. Cont rib. Postgres, поставляемом в составе фреймворка. Чтобы они успешно работали, приложение следует добавить в список зарегистрированных в проекте:

****

Внимание!

Полная русскоязычная документация по postgresql 12 находится по интернет-адресу [Https://postgrespro.ru/docs/postgresql/12/index](https://postgrespro.ru/docs/postgresql/12/index).

**Объявление моделей для работы с postgresql**

**Поля, специфические для postgresql**

Классы всех ЭТИХ полей объявлены В модуле django.contrib.postgres. Fields:

П integerrangefieid— поле диапазона, хранящее диапазон целочисленных значений обычной длины (32-разрядное) в виде его начального и конечного значений.

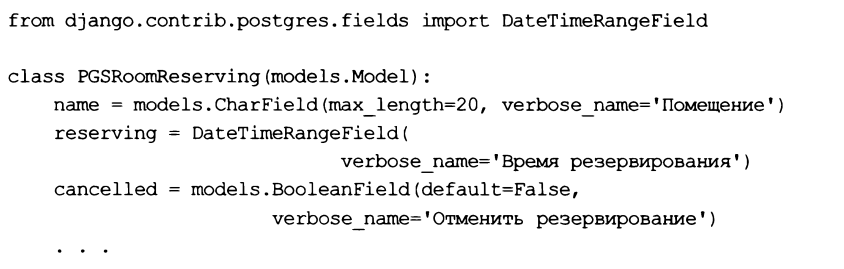
□ bigintegerrangefieid— поле диапазона, хранящее диапазон целочисленных значений двойной длины (64-разрядное).

□ decimalrangefield (начиная с Django 2.2) — поле диапазона, хранящее диапазон чисел фиксированной ТОЧНОСТИ В виде объектов типа Decimal ИЗ модуля decimal Python.

□ daterangefieid— поле диапазона, хранящее диапазон значений даты в виде объектов типа date ИЗ модуля datetime.

□ datetimerangefieid — поле, хранящее диапазон временных отметок в виде объектов типа datetime ИЗ модуля datetime.

Пример объявления модели pgsroomreserving С полвм типа datetimerangefieid:

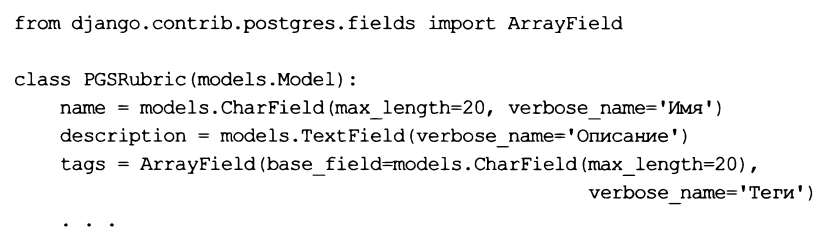
****

□ arrayfieid— поле списка, хранящее список Python, элементы которого обязательно должны принадлежать одному типу. Дополнительные параметры:

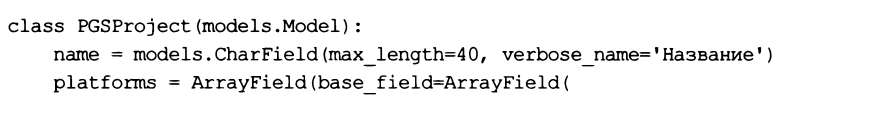
• base fieid —тип элементов, сохраняемых в поле списков. Указывается в виде объекта (не класса!) Соответствующего поля модели;

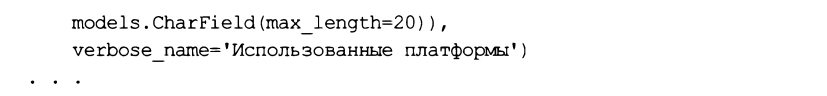
• size— максимальный размер сохраняемых в поле списков в виде целого числа. При попытке записать в поле список большего размера возникнет ошибка. По умолчанию— None (максимальный размер списков не ограничен).

Пример объявления в модели рубрик pgsrubric поля списка tags, хранящего строковые величины:

****

Пример объявления в модели pgsproject поля списка platforms, хранящего поля списка, которые, в свою очередь, хранят строковые значения (фактически создается поле, способное хранить двумерные списки):



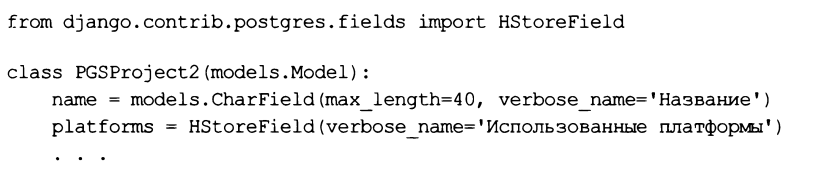


□ hstorefieid — поле словаря, хранящее словарь Python. Ключи элементов такого словаря должны быть строковыми, а значения могут быть строками или None.

Внимание!

Для успешной работы поля hstorefieid следует добавить в базу данных расширение Hstore. Как добавить в базу данных postgresql расширение, будет рассказано позже.

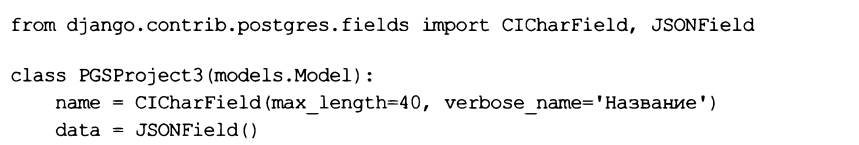
Пример объявления в модели pgsproject2 поля словаря platforms:

****

S

□ jsonfieid —поле, хранящее данные в формате JSON.

□ cicharfieid— то же самое, что и charfield, только при выполнении поиска по этому полю не учитывается регистр символов:

****

□ citextfieid — то же самое, что и textfieid, только при выполнении поиска по этому полю не учитывается регистр символов.

□ ciemailfieid —то же самое, что и emaiifieid, только при выполнении поиска по этому полю не учитывается регистр символов.

Внимание!

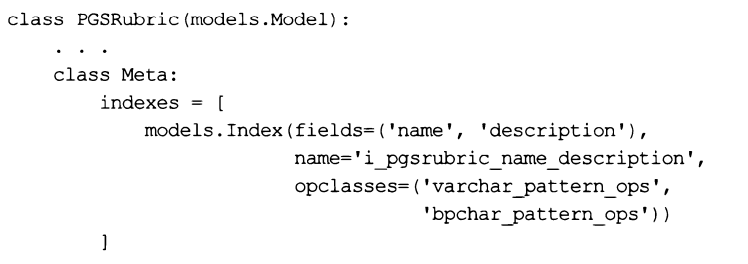
Чтобы ПОЛЯ cicharfield, citextfieid И ciemailfieid успешно работали, В базу данных следует добавить расширение citext.

**Индексы postgresql**

Индекс, создаваемый указанием в конструкторе поля параметров unique, db index, Primary key или посредством класса index, получит тип В-Тгее, подходящий для большинства случаев.

Начиная с Django 2.2, при создании индексов посредством класса index можно указать у индексируемых полей классы операторов postgresql. Для этого применяется необязательный параметр opciasses конструктора класса. Ему нужно присвоить последовательность имен классов операторов, представленных строками: первый класс оператора будет применен к первому индексированному полю, второй — ко второму полю и т. д.

Пример создания индекса по полям паше И description С указанием у ПОЛЯ паше класса оператора varchar pattern ops, а у поля description — класса оператора Bpchar\_pattеrn\_ops:

****

Для создания индексов других типов, поддерживаемых postgresql, следует применять следующие классы ИЗ модуля django. Contrib.postgres. Indexes:

□ btreeindex — индекс формата В-Tree.

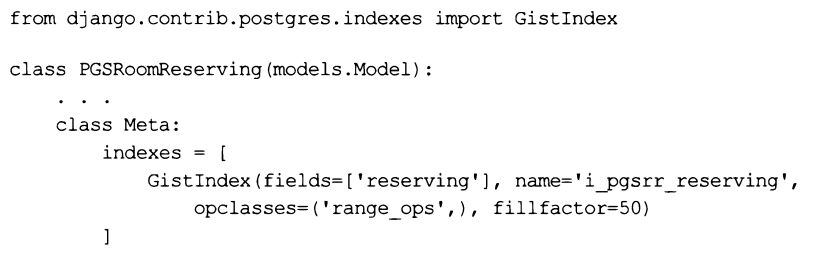
Дополнительный параметр fillfactor указывает степень заполнения индекса в процентах в виде целого числа от 10 до 100. По умолчанию — None (90%).

□ Gistindex — индекс формата gist. Дополнительные параметры:

• buffering— если True, то при построении индекса будет включен механизм буферизации, если False — не будет включен, если None — включать или не включать буферизацию, решает СУБД (по умолчанию — None);

• fillfactor — степень заполнения индекса в процентах в виде целого числа от 10 ДО 100. По умолчанию--- None (90%).

Пример создания такого индекса ПО ПОЛЮ reserving модели pgsroomreserving с применением класса оператора range ops:

****

Внимание!

При индексировании таким индексом полей, отличных от integerrangefield, Biglntegerrangefield, decimalrangefield, datetimerangefield И daterangefield, следует установить в базе данных расширение btree gist.

□ spgistindex (начиная с Django 2.2) — индекс формата SP-gist.

Дополнительный параметр fillfactor указывает степень заполнения индекса в процентах в виде целого числа от 10 до 100. По умолчанию — None (90%).

□ Hashindex (начиная с Django 2.2) — индекс на основе хэшей.

Дополнительный параметр fillfactor указывает степень заполнения индекса в процентах в виде целого числа от 10 до 100. По умолчанию — None (90%).

□ Brinindex —индекс формата BRIN. Дополнительные параметры:

• autosummarize— если True, то СУБД будет подсчитывать и сохранять в индексе итоговую информацию по каждому блоку записей, если False или None — не будет делать этого (по умолчанию — None). Такая итоговая информация позволяет ускорить фильтрацию и сортировку, однако увеличивает объем индекса;

• pages per range — количество страниц в блоке записей, по которому будет подсчитываться итоговая информация, в виде целого числа. По умолчанию — None (128 страниц).

□ Ginindex —индекс формата GIN. Дополнительные параметры:

• fastupdate — если True или None, то будет задействован механизм быстрого обновления индекса, при котором обновляемые записи сначала добавляются во временный список, а уже потом, при выполнении обслуживания базы данных, переносятся непосредственно в индекс. Если False, то механизм быстрого обновления будет отключен. По умолчанию — None;

• gin pending iist iimit — объем временного списка обновляемых записей в виде целого числа в байтах. По умолчанию — None (4 Мбайт).

Внимание!

При индексировании таким индексом полей, отличных от arrayfieid и jsonfield, следует установить в базе данных расширение btree gin.

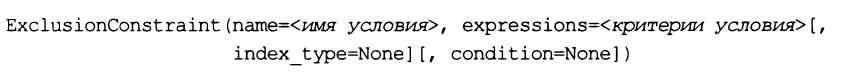
**Специфическое условие postgresql**

В параметре constraints модели можно указать условие Exclusionconstraint ИЗ модуля django.contrib.postgres.constraints, поддержка которого появилась в Django 3.0. Оно указывает критерий, которому НЕ должны удовлетворять записи, добавляемые в модель. Если запись удовлетворяет этим критериям, то она не будет добавлена в модель, и сгенерируется исключение Integrityerror ИЗ модуля django.db.

Внимание!

Для успешной работы этого условия следует установить в базе данных расширение Btree\_gist.

Формат конструктора класса Exclusionconstraint:



Имя условия задается в виде строки и должно быть уникальным в пределах проекта.

Критерии условия указываются в виде списка или кортежа, каждый элемент которого задает один критерий и должен представлять собой список или кортеж из двух элементов:

□ имени поля в виде строки или экземпляра класса f . Значение этого поля из добавляемой записи будет сравниваться со значениями полей из записей, уже имеющихся в базе данных;

□ оператора сравнения, поддерживаемого языком SQL, в виде строки. Посредством этого оператора будет выполняться сравнение значений поля, заданного в первом элементе.

Для задания операторов сравнения также можно применять следующие атрибуты класса Rangeoperators ИЗ модуля django.contrib.postgres. Fields:

• EQUAL —= (’’равно”);

• notequal — о (”не равно”);

• contains — @> (’’содержит” — диапазон из имеющейся записи содержит диапазон или значение из добавляемой записи);

• contained by — <@ (’’содержится” — диапазон или значение из имеющейся записи содержится в диапазоне из добавляемой записи);

• overlaps — @@ (’’пересекаются” — оба диапазона пересекаются);

• fully lt — «(’’строго слева” — диапазон из имеющейся записи меньше, или находится левее, диапазона из добавляемой записи);

• fully gt — » (’’строго справа” — диапазон из имеющейся записи больше, или находится правее, диапазона из добавляемой записи);

• not lt— &> (”не левее”— все значения диапазона из имеющейся записи меньше нижней границы диапазона из добавляемой записи);

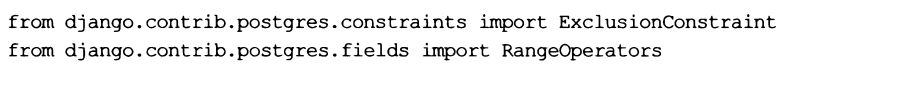
• not-gt— &< (”не правее”— все значения диапазона из имеющейся записи больше верхней границы диапазона из добавляемой записи);

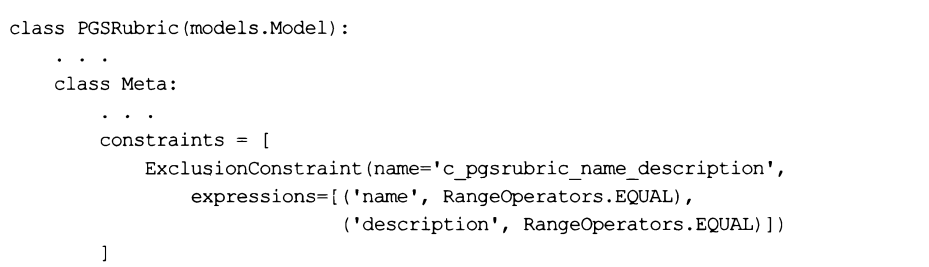
• adjacent to----- 1 - (’’примыкают” — диапазоны примыкают друг к другу, т. е. имеют общую границу).

Параметр index type указывает тип создаваемого индекса в виде строки ’gist\* (gist) или ' spgist ' (SP-gist). Если не указан, то будет создан индекс gist.

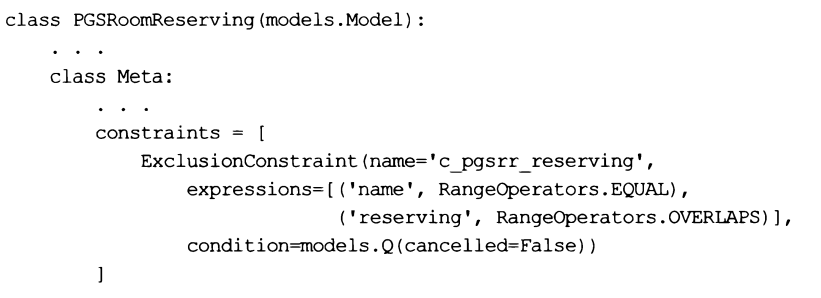
Параметр condition позволяет задать критерий фильтрации, ограничивающий набор записей, к которым будет применяться создаваемое условие. Критерий указывается в виде экземпляра класса q. Если не задан, то условие будет применяться ко всем записям в таблице.

Пример создания в модели pgsrubric условия, запрещающего добавлять в базу рубрики с совпадающими названиями и описаниями:



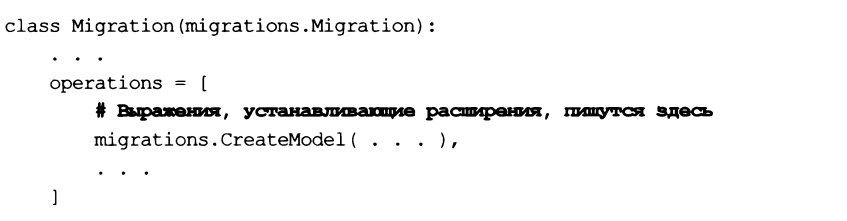


Пример создания в модели pgsroomreserving условия, запрещающего дважды резервировать одно и то же помещение на тот же или частично совпадающий промежуток времени, но не мешающее отменить резервирование помещения:

****

Расширение postgresql— это эквивалент библиотеки Python. Оно упаковано в компактный пакет, включает в себя объявление новых типов полей, индексов, операторов, функций и устанавливается в базе данных подачей специальной sqlкоманды.

Установка расширений postgresql выполняется в миграциях. Класс миграции Migration, объявленный В Модуле каждой миграции, содержит атрибут operations, хранящий последовательность операций, которые будут выполнены с базой данных при осуществлении миграции. В начале этой последовательности и записываются выражения, устанавливающие расширения:

****

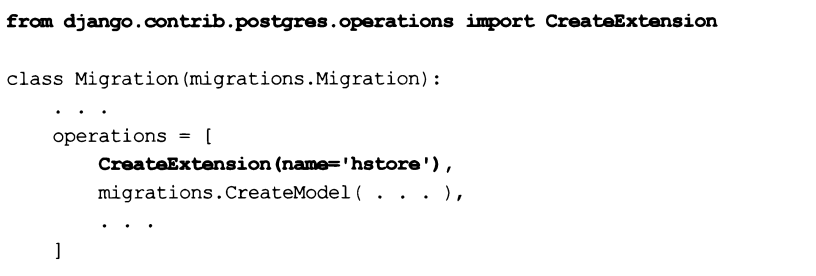
Эти выражения должны создавать экземпляры классов, представляющих миграции.

Все эти классы, объявленные в модуле django.contrib.postgres.operations, перечислены далее:

□ createextension — устанавливает в базе данных расширение с заданным в виде строки именем:

Createextension(пате=<имя расширения^

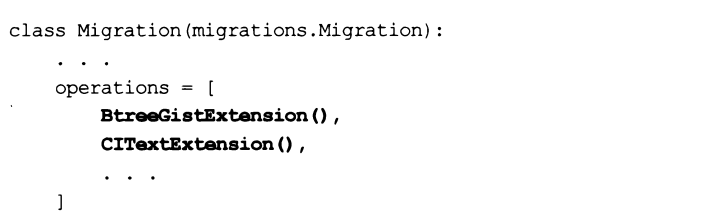
Пример установки расширения hstore:

****

□ btreeginextension —устанавливает расширение btree\_gin;

□ btreegistextension —устанавливает расширение btree\_gist;

□ citextextension —устанавливает расширение citext. Пример:

****

□ cryptoextension —устанавливает расширение pgcrypto;

□ hstoreextension —устанавливает расширение hstore;

□ trigramextension —устанавливает расширение pg trgm;

□ unaccentextension —устанавливает расширение unaccent.

Расширения можно добавить как в обычной миграции, вносящей изменения в базу данных, так и в "пустой" миграции.

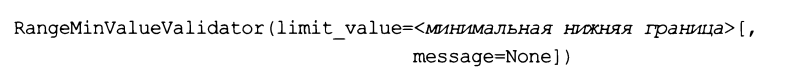
**Валидаторы postgresql**

Специфические для postgresql валидаторы, объявленные в модуле django.

Contrib.postgres.validators, перечислены далее:

□ rangeminvaiuevaiidator — проверяет, не меньше ли нижняя граница диапазона, сохраняемого в поле диапазона, заданной в параметре limit value величины.

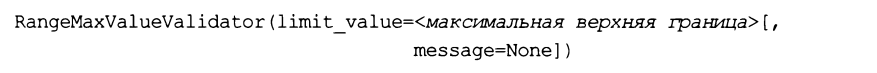
Формат конструктора:

****

Параметр message задает сообщение об ошибке; если он не указан, то используется стандартное. Код ошибки: ”min\_vaiue”.

Начиная с Django 2.2, в качестве значения параметра limit value можно указать функцию, не принимающую параметров и возвращающую минимальную нижнюю границу;

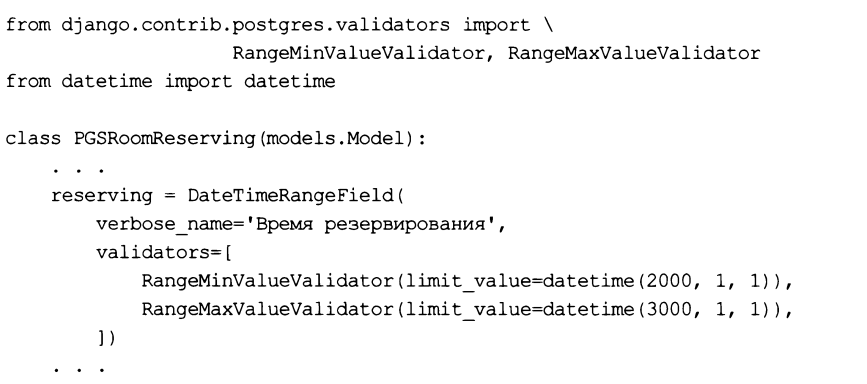
□ rangemaxvaiuevaiidator— проверяет, не превышает ли верхняя граница диапазона, сохраняемого в поле диапазона, заданную в параметре limit value величину. Формат конструктора:

****

Параметр message задает сообщение об ошибке; если он не указан, используется стандартное. Код ошибки: "max value".

Начиная с Django 2.2, в качестве значения параметра limit value можно указать функцию, не принимающую параметров и возвращающую максимальную верхнюю границу.

Пример использования валидаторов rangeminvaluevalidator И rangemaxvaiuevaiidator:

****

□ arrayminlengthvaiidator— проверяет, не меньше ли размер заносимого списка заданного в первом параметре минимума. Формат конструктора:

Arrayminlengthvaiidator (<минимальныи размер> [, message=None ] ) параметр message задает сообщение об ошибке — если он не указан, то выдается стандартное сообщение. Код ошибки: "min iength".

Начиная с Django 2.2, в качестве значения первого параметра можно указать функцию, не принимающую параметров и возвращающую минимальный размер списка в виде целого числа;

□ arraymaxlengthvaiidator — проверяет, не превышает ли размер заносимого списка заданный в первом параметре максимум. Используется полем типа Arrayfieid с указанным параметром size. Формат конструктора:

Arraymaxlengthvaiidator(<максимальный размер>[, message=None]) параметр message задает сообщение об ошибке — если он не указан, то используется стандартное. Код ошибки: "max length".

Начиная с Django 2.2, в качестве значения первого параметра можно указать функцию, не принимающую параметров и возвращающую максимальный размер списка в виде целого числа;

□ keysvaiidator— проверяет, содержит ли словарь, записанный в поле словаря, элементы с заданными ключами. Формат конструктора:

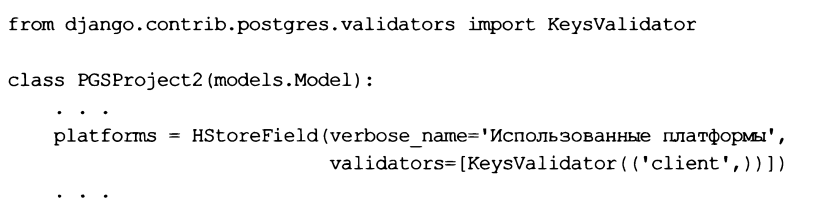
Keysvaiidator(<ключи>[, strict=False][, messages=None]) первым параметром указывается последовательность строк с ключами, наличие которых в словаре следует проверить.

Если параметру strict дать значение True, то будет выполняться проверка, не присутствуют ли в словаре какие-либо еще ключи, помимо перечисленных в первом параметре. Если значение параметра strict равно False или этот параметр вообще не указан, то такая проверка проводиться не будет.

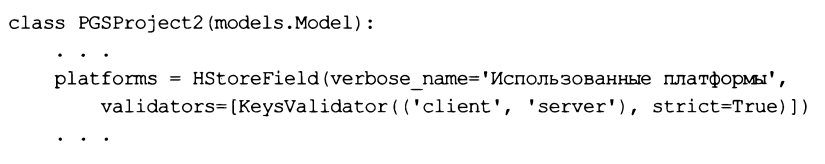
Параметру message, задающему сообщения об ошибках, следует присвоить словарь с элементами missing keys и extra keys; первый задаст сообщение об отсутствии в словаре ключей, перечисленных в первом параметре, а второй — сообщение о наличии в словаре "лишних" ключей. Если сообщения об ошибках не заданы, будут выводиться сообщения по умолчанию.

Коды ошибки: "missing keys" — при отсутствии в словаре требуемых элементов, "extra keys" —при наличии в словаре "лишних" элементов.

Пример проверки наличия в сохраняемом в поле platforms словаре элемента с ключом client:

****

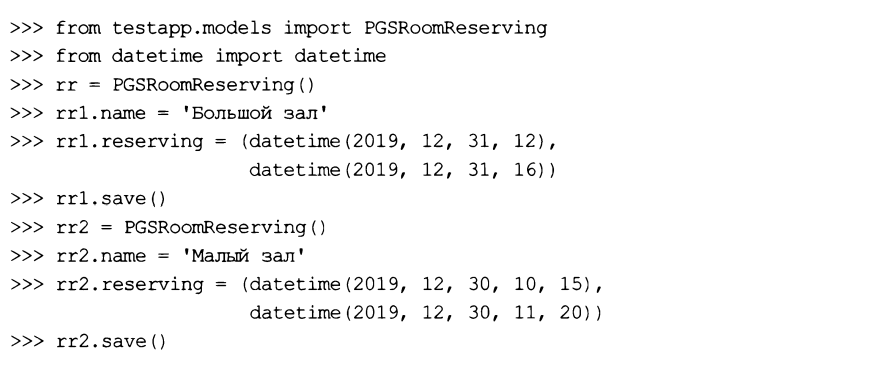
Пример проверки наличия в сохраняемом в поле platforms словаре элементов с ключами client, server и отсутствия там других элементов:

****

**Запись и выборка значений полей в postgresql**

Запись и выборка отдельных значений из полей типов, специфичных для Postgresql, имеет следующие особенности:

□ поля диапазона— сохраняемый диапазон должен представлять собой список или кортеж из двух элементов — нижней и верхней границы диапазона. Если в качестве одной из границ указать None, то диапазон не будет ограничен с соответствующей стороны. Примеры:



Заносимый в поле диапазон также может быть представлен экземпляром класса numericrange (ДЛЯ ПОЛЯ типа integerrangefield, biglntegerrangefield И Decimalrangefield), daterange (ДЛЯ ПОЛЯ daterangefield) ИЛИ datetimetzrange (для ПОЛЯ datetimerangefield). Все ЭТИ классы объявлены в модуле psycopg2.

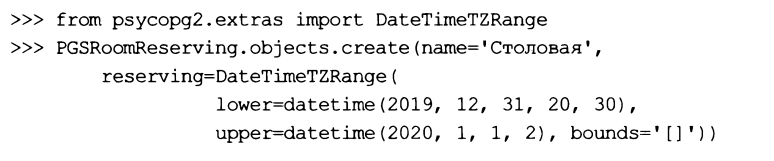
Extras. Формат их конструкторов:



Параметры lower и upper задают соответственно нижнюю и верхнюю границу диапазона. Если какой-либо из этих параметров не указан, то соответствующая граница получит значение None, и диапазон не будет ограничен с этой стороны.

Параметр bounds указывает, будут нижняя и верхняя границы входить в диапазон или нет. Его значением должна быть строка из двух символов, первый из которых задает вхождение или не вхождение в диапазон нижней границы, второй — верхней. Символ [ обозначает вхождение соответствующей границы в диапазон, символ ( — не вхождение.

Пример занесения в поле диапазона, у которого обе границы входят в диапазон:



Значение диапазона извлекается из поля также в виде экземпляра одного из перечисленных ранее классов диапазонов. Для работы с ним следует применять следующие атрибуты:

• lower— нижняя граница диапазона или None, если диапазон не ограничен снизу;

• upper— верхняя граница диапазона или None, если диапазон не ограничен сверху;

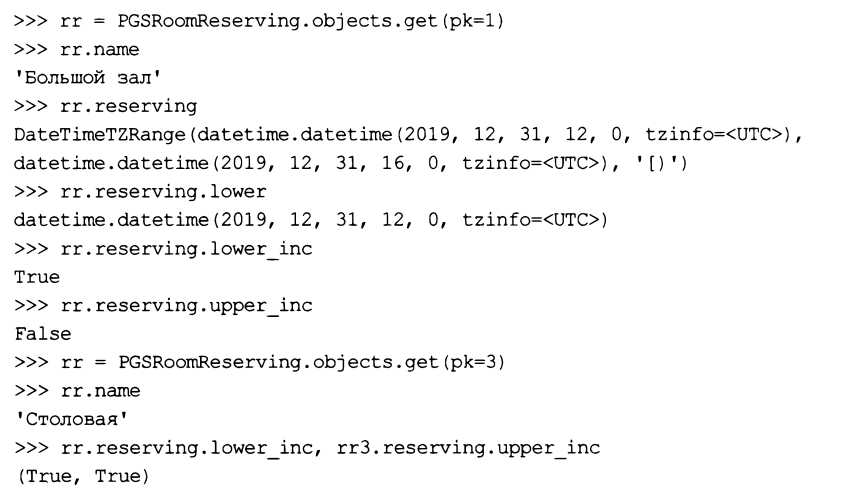
• lower inf — True, если диапазон не ограничен снизу, и False — в противном случае;

• upper inf —True, если диапазон не ограничен сверху, и False —в противном случае;

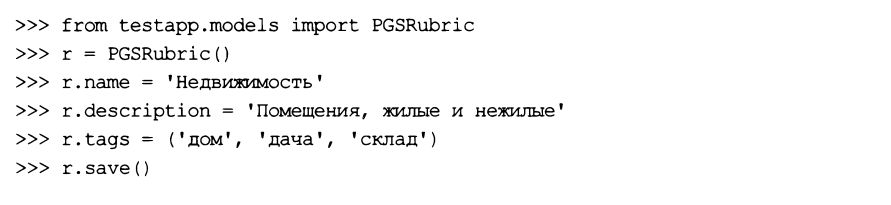
• lower inc —True, если нижняя граница не входит в диапазон, и False — если входит;

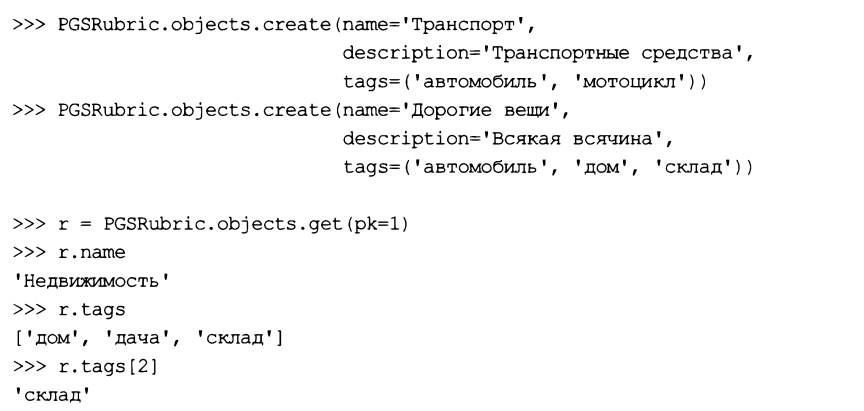
• upper inc —True, если верхняя граница не входит в диапазон, и False —если входит.

Примеры:

****

□ поле списка (arrayfieid) — сохраняемое значение можно указать в виде списка или кортежа Python:

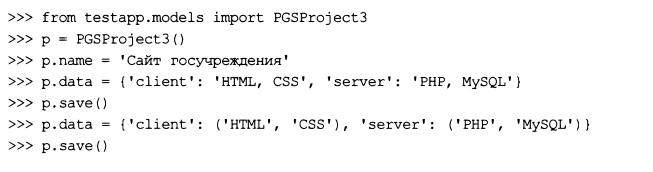


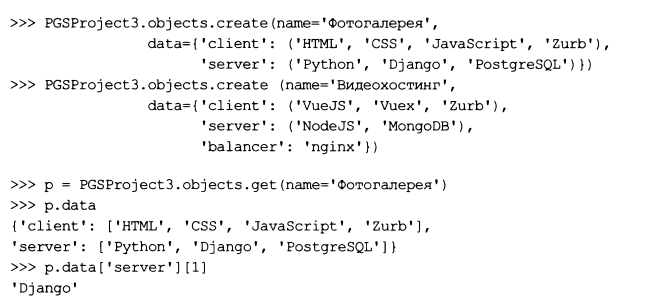


□ поле словаря (hstorefield):



□ jsonfieid — значение может быть числом, строкой, логической величиной, последовательностью, словарем Python или None:





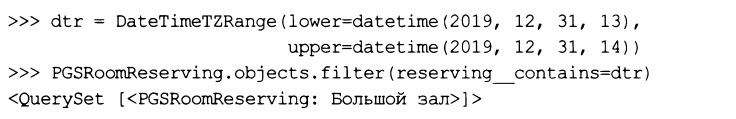
Значения В ПОЛЯ cicharfield, citextfieid И ciemailfieid ЗЯНОСЯТСЯ В ТОМ же виде, ЧТО И В аналогичные ПОЛЯ charfield, textfield И emailfield.

**Фильтрация записей в postgresql**

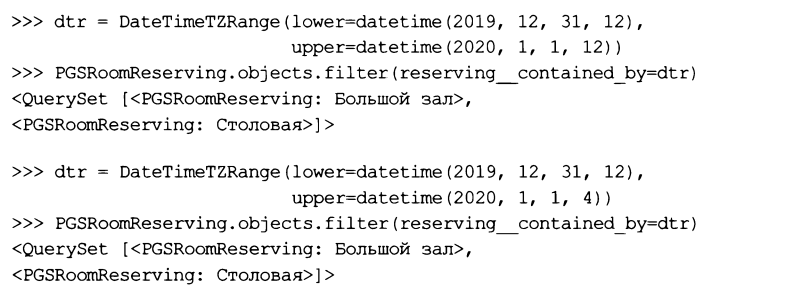
Для фильтрации записей по значениям полей специфических типов postgresql предоставляется ряд особых модификаторов:

□ поля диапазона:

• contains — хранящийся в поле диапазон должен содержать указанный диапазон:

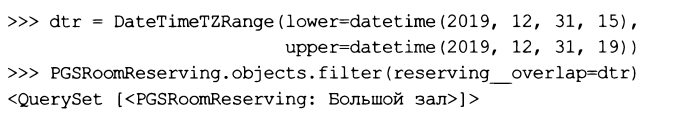
****

• contained by — хранящийся в поле диапазон должен содержаться в указанном диапазоне:

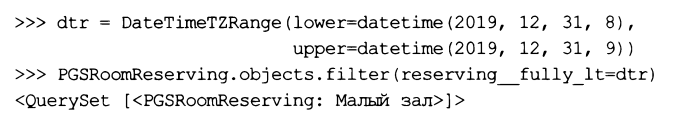
****

Модификатор contained by может применяться не только к полям диапазонов, НО И К ПОЛЯМ ТИПОВ integerfield, biglntegerfield, floatfield, datefieid И datetimefieid. В этом случае будет проверяться, входит ли хранящееся в поле значение в указанный диапазон;

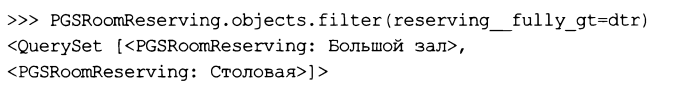
• overlaps —хранящийся в поле диапазон должен пересекаться с заданным:

****

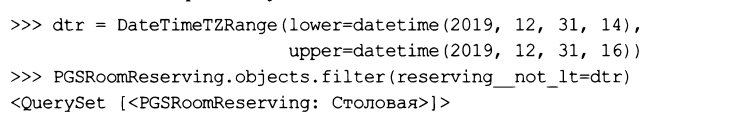
• fuiiy it — сохраненный в поле диапазон должен быть меньше (находиться левее) указанного диапазона:

****

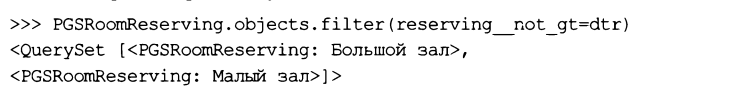
• fuily gt — сохраненный в поле диапазон должен быть больше (находиться правее) указанного диапазона:

****

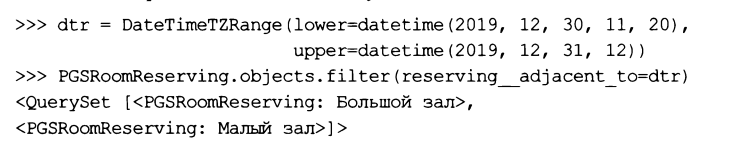
• not it — ни одна точка сохраненного в поле диапазона не должна быть меньше нижней границы указанного диапазона:

****

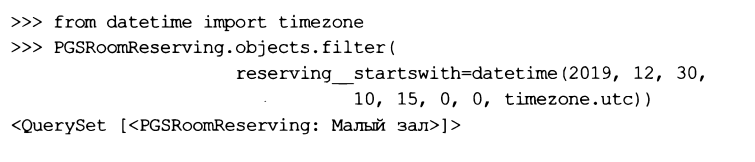
• not gt— ни одна точка сохраненного в поле диапазона не должна быть больше верхней границы указанного диапазона:

****

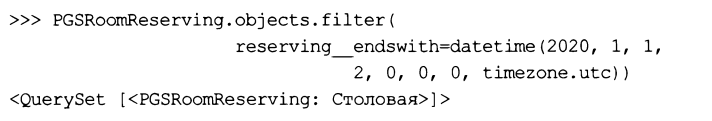
• adjacent to— граничное значение сохраненного в поле диапазона должно совпадать с граничным значением указанного диапазона:

****

• startswith— сохраненный в поле диапазон должен иметь указанную нижнюю границу:

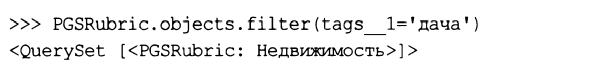
****

• startswith— сохраненный в поле диапазон должен иметь указанную верхнюю границу:

****

□ arrayfield:

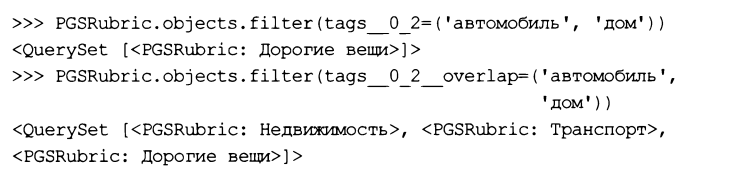
• <индекс элемента>— из сохраненного в поле списка извлекается элемент с заданным индексом, и дальнейшее сравнение выполняется с извлеченным элементом. В качестве индекса можно использовать лишь неотрицательные целые числа. Пример:

****

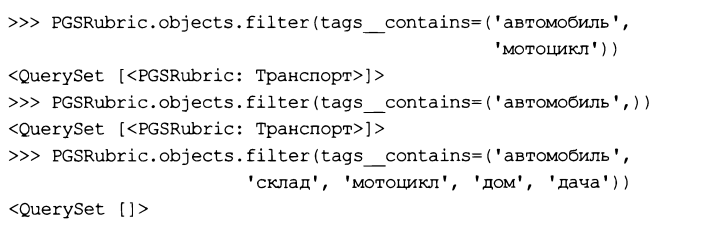
• <индекс первого элемента>\_<индекс последнего элемента> — ИЗ сохраненного

В поле списка берется срез, расположенный между элементами с указанными

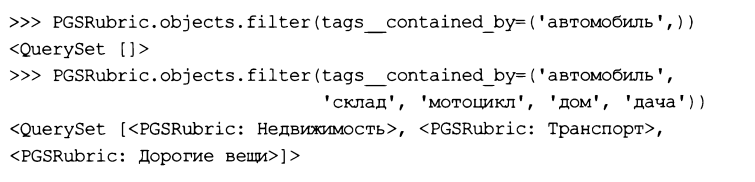
Индексами, и дальнейшее сравнение выполняется с полученным срезом. В качестве индексов допустимы лишь неотрицательные целые числа. Пример:

****

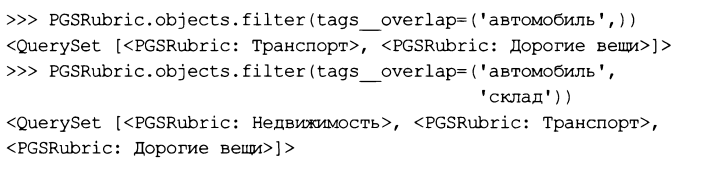
• contains — сохраненный в поле список должен содержать все элементы, перечисленные в заданной последовательности:

****

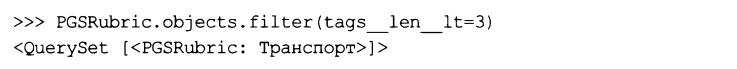
• contained by — все элементы сохраненного в поле списка должны присутствовать в заданной последовательности:

****

• overlap— сохраненный в поле список должен содержать хотя бы один из элементов, перечисленных в заданной последовательности:

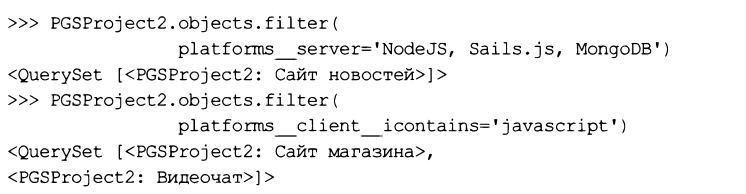
****

• len —определяется размер хранящегося в поле списка, и дальнейшее сравнение выполняется с ним:

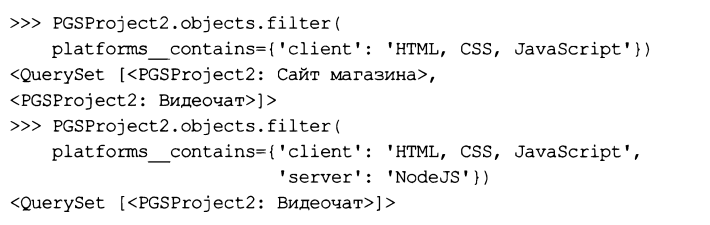
****

\* hstorefieid:

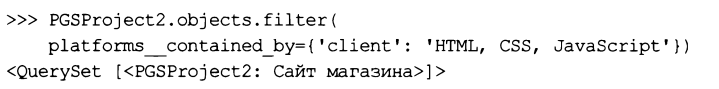
• <ключ>— из сохраненного в поле словаря извлекается элемент с заданным ключом, и дальнейшее сравнение выполняется с ним:

****

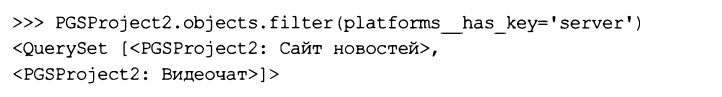
• contains — сохраненный в поле словарь должен содержать все элементы, перечисленные в заданном словаре:

****

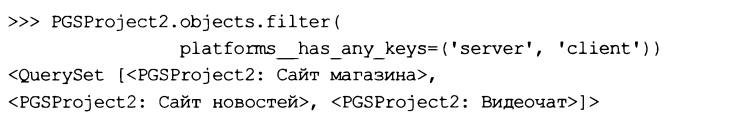
• contained by —все элементы сохраненного в поле словаря должны присутствовать в заданном словаре:

****

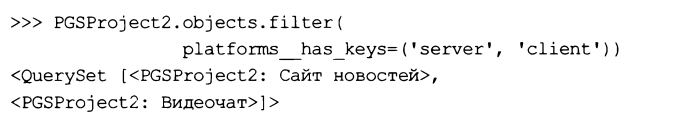
• has Rey — сохраненный в поле словарь должен содержать элемент с указанным ключом:

****

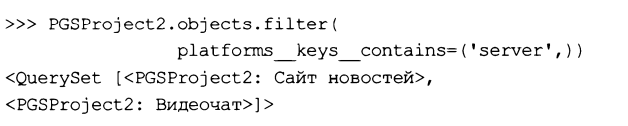
• has any keys —сохраненный в поле словарь должен содержать хотя бы один элемент с ключом, присутствующим в заданной последовательности:

****

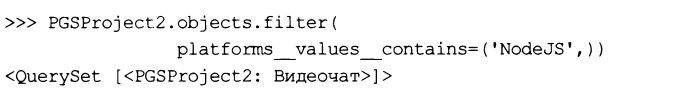
• has keys — сохраненный в поле словарь должен содержать все элементы с ключами, присутствующими в заданной последовательности:

****

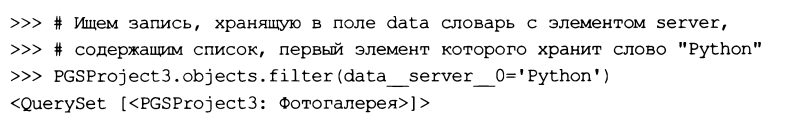
• keys — из сохраненного в поле словаря извлекаются ключи всех элементов, из ключей формируется список, и дальнейшее сравнение выполняется с ним:

****

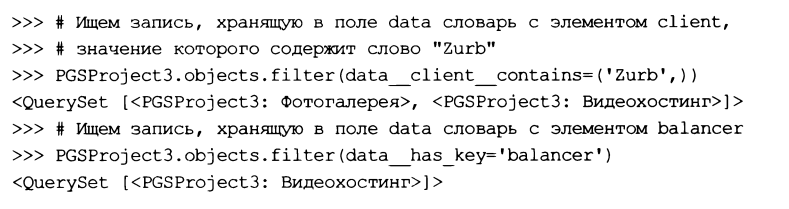
• values — из сохраненного в поле словаря извлекаются значения всех элементов, из значений формируется список, и дальнейшее сравнение выполняется с ним:

****

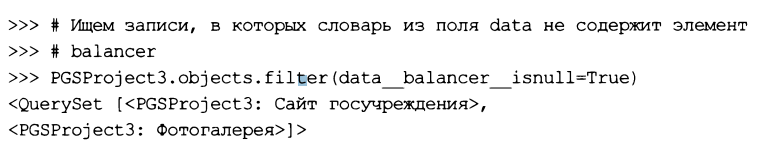
□ jsonfieid— можно обращаться к элементам и атрибутам хранящихся в поле объектов, перечисляя индексы, ключи, имена элементов или атрибутов через двойное подчеркивание (\_\_):

****

Также поддерживаются модификаторы contains, contained\_by, has\_key, has\_any\_Keys и has keys, описанные ранее:

****

Чтобы найти запись с JSON-данными, в которых отсутствует какой-либо элемент или атрибут, следует воспользоваться модификатором isnuli со значением True:

****

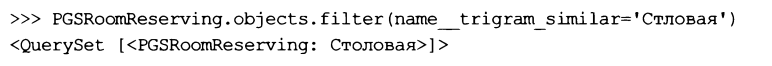
Два следующих специфических модификатора postgresql помогут при фильтрации записей по значениям строковых и текстовых полей:

□ trigram simiiar— хранящееся в поле значение должно быть похоже на заданное.

Внимание!

Для использования модификатора trigram simiiar следует установить расширение Pg\_trgm.

Пример:

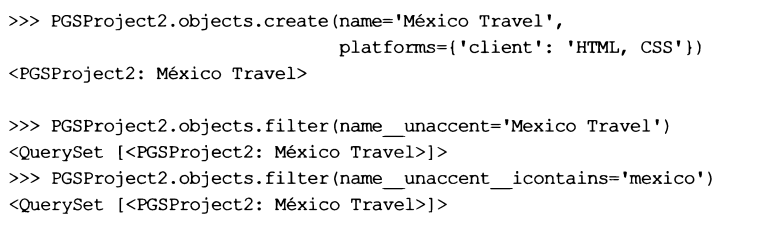
****

□ unaccent — выполняет сравнение хранящегося в поле и заданного значений без учета диакритических символов.

Внимание!

Для использования модификатора unaccent следует установить расширение unaccent.

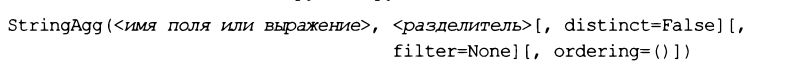
Примеры:

****

**Агрегатные функции postgresql**

Все классы специфических агрегатных функций postgresql объявлены в модуле Django.contrib.postgres.aggregates:

□ stringagg —возвращает строку, составленную из значений, извлеченных из указанного поля и отделенных друг от друга заданным разделителем:

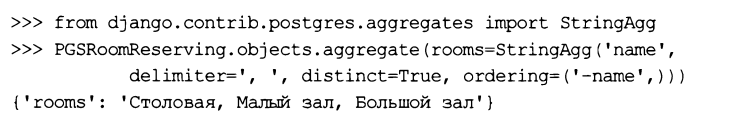
****

Первым параметром указывается имя поля, значения которого составят результирующую строку, или выражение, представленное экземпляром класса f. Вторым параметром задается строка с разделителем.

Если параметру distinct задано значение False (по умолчанию), то в результирующую строку войдут все значения заданного поля или выражения, если False— только уникальные значения. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q; если он не задан, фильтрация не выполняется.

Параметр ordering поддерживается, начиная с Django 2.2. Он указывает последовательность полей, по которым будет выполняться сортировка элементов в результирующей строке. Каждое поле задается в виде строки с именем (по умолчанию сортировка выполняется по возрастанию; чтобы задать сортировку по убыванию, следует предварить имя поля дефисом) или экземпляра класса F.

Пример выборки всех названий комнат ИЗ модели pgsroomreserving С сортировкой в обратном алфавитном порядке:

****

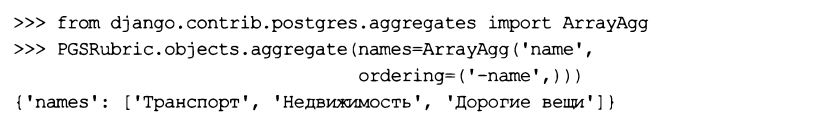
□ arrayagg —возвращает список из значений, извлеченных из указанного поля:

Arrayagg(<имя поля или выражение>[, distinct=False][, filter=None][, Ordering=()])

Первым параметром указывается имя поля, значения которого войдут в список, Или выражение, представленное экземпляром класса F. Если параметру distinct задано значение False (по умолчанию), то в результирующий СПИСОК ВОЙДУТ все Значения Заданного поля или выражения, если False — только уникальные значения. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q; если он не задан, фильтрация не выполняется.

Параметр ordering поддерживается, начиная с Django 2.2. Он указывает последовательность полей, по которым будет выполняться сортировка элементов в результирующей последовательности. Каждое поле может быть указано в виде строки с именем (по умолчанию сортировка выполняется по возрастанию; чтобы задать сортировку по убыванию, следует предварить имя поля дефисом) или экземпляра класса F.

Пример создания списка с именами рубрик из модели pgsrubric, отсортированных в обратном алфавитном порядке:

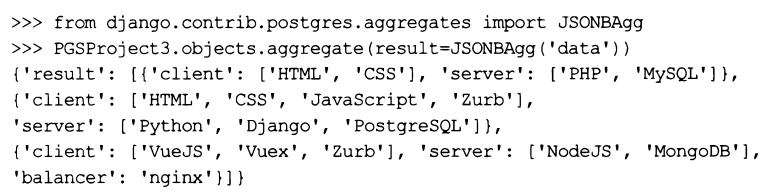
****

□ jsonbagg — возвращает значения указанного поля в виде JSON-объекта:



Первым параметром указывается имя поля, значения которого войдут в объект, или выражение, представленное экземпляром класса f. Параметр Filter указывает условие фильтрации записей в виде экземпляра класса q; если он не задан, фильтрация не выполняется.

Пример:

****

□ bitand — возвращает результат побитового умножения (И) всех значений заданного поля, отличных от None, или None, если все значения поля равны None:

Bitand(<имя поля или выражение>[, filter=None])

Первым параметром указывается имя поля, значения которого будут перемножаться, или выражение, представленное экземпляром класса f.

Параметр filter указывает условие фильтрации записей в виде экземпляра класса Q; если он не задан, фильтрация не выполняется.

□ Bitor — возвращает результат побитового сложения (ИЛИ) всех значений заданного поля, отличных от None, или None, если все значения поля равны None:



Параметры указываются в том же формате, что и у функции bitand (см. ранее).

□ booiand— возвращает True, если все значения заданного поля равны True, None— если все значения равны None или в таблице нет записей, и False — в противном случае:



Параметры указываются в том же формате, что и у функции bitand (см. Ранее).

□ вооюг— возвращает True, если хотя бы одно значение заданного поля равно True, None — если все значения равны None или в таблице нет записей, и False — в противном случае:



Параметры указываются в том же формате, что и у функции bitand (см. Ранее).

□ согг — возвращает коэффициент корреляции, вычисленный на основе значений из поля Y и поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



Первыми двумя параметрами указываются имена числовых полей ИЛИ выражения, представленные экземплярами класса f. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q;

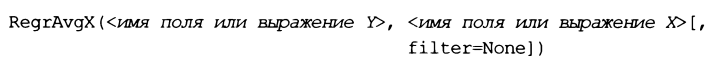
Если он не задан, фильтрация не выполняется.

□ covarpop— возвращает ковариацию населения, вычисленную на основе значений из поля y и поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



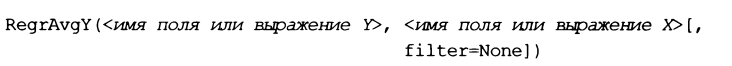
Первыми двумя параметрами указываются имена числовых полей ИЛИ выражения, представленные экземплярами класса f. Если параметру sample дать значение False, то будет возвращена выборочная ковариация населения, если дать значение True — ковариация населения в целом. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q; если он не задан, фильтрация не выполняется.

□ regravgx — возвращает среднее арифметическое, вычисленное на основе значений из поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



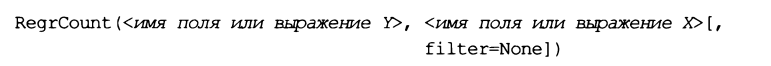
Параметры указываются в том же формате, что и у функции Corr (см. Ранее).

□ regravgy — возвращает среднее арифметическое, вычисленное на основе значений из поля Y, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



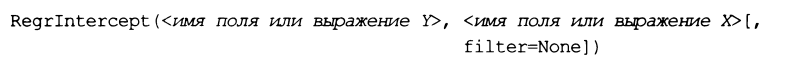
Параметры указываются в том же формате, что и у функции Corr (см. ранее).

□ regrcount — возвращает количество записей, в которых поле y и поле х хранят величины, отличные ОТ None, В виде целого числа (тип int) ИЛИ None, если в наборе нет ни одной записи:



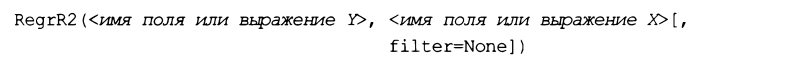
Параметры указываются в том же формате, что и у функции Corr (см. Ранее).

□ Regrintercept — возвращает величину точки пересечения оси Y и линии регрессии, рассчитанную методом наименьших квадратов на основе значений поля y и поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



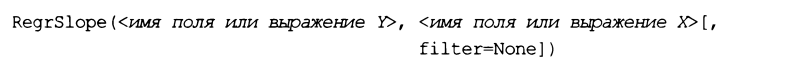
Параметры указываются в том же формате, что и у функции Corr (см. ранее).

□ regrr2 — возвращает квадрат коэффициента корреляции, вычисленный на основе Значений ИЗ поля Y и поля X, в виде вещественного числа (ТИП float) ИЛИ None, если в наборе нет ни одной записи:



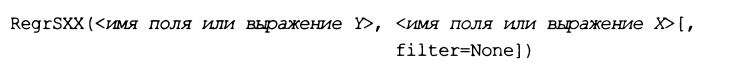
Параметры указываются в том же формате, что и у функции Corr (см. Ранее).

□ Regrsiope— возвращает величину наклона линии регрессии, рассчитанную методом наименьших квадратов на основе значений поля у и поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



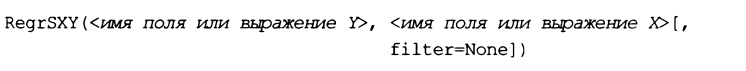
Параметры указываются в том же формате, что и у функции Corr (см. ранее).

□ Regrsxx — возвращает сумму площадей, рассчитанную на основе значений поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



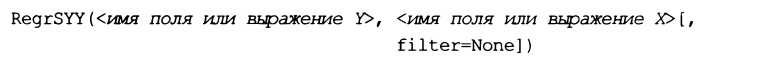
Параметры указываются в том же формате, что и у функции Corr (см. ранее).

□ regrsxy— возвращает сумму производных, рассчитанную на основе значений поля у и поля х, в виде вещественного числа (тип float) или None, если в наборе нет ни одной записи:



Параметры указываются в том же формате, что и у функции Corr (см. ранее).

□ regrsyy — возвращает сумму площадей, рассчитанную на основе значений поля Y, в виде вещественного числа (тип float) или None, если в наборе нет ни одно записи:



Параметры указываются в том же формате, что и у функции Corr (см. ранее).

**Функции СУБД, специфичные для postgresql**

Классы ЭТИХ функций объявлены в модуле django. Contrib.postgres. Functions:

□ randomuuid — возвращает сгенерированный случайным образом уникальный универсальный идентификатор.

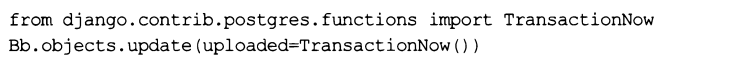
Внимание!

Для использования функции randomuuid следует установить расширение pgcrypto.

□ transactionnow— возвращает временную отметку (дату и время) запуска текущей транзакции или, если транзакция не была запущена, текущую временную отметку.

Если существует несколько вложенных друг в друга транзакций, будет возвращена временная отметка запуска наиболее ’’внешней” транзакции.

Пример:

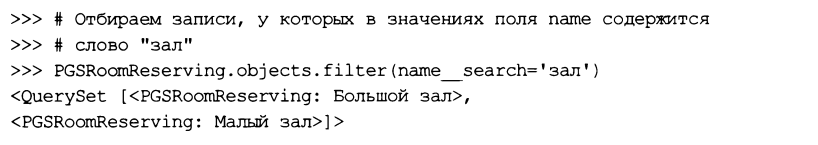
****

**Полнотекстовая фильтрация postgresql**

Полнотекстовая фильтрация — это фильтрация записей по наличию в строковых или текстовых значениях, сохраненных в их полях, заданных слов. Подобного рода фильтрация используется в поисковых интернет-службах.

**Модификатор search**

Модификатор search выполняет полнотекстовую фильтрацию по одному полю:

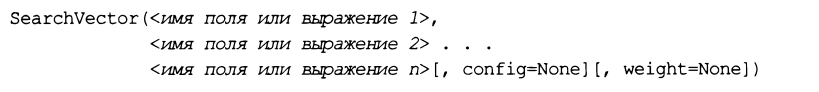
****

**Функции СУБД для полнотекстовой фильтрации**

Классы ЭТИХ функций объявлены В модуле django.contrib.postgres.search:

□ Searchvector— задает поля, по которым будет осуществляться фильтрация.

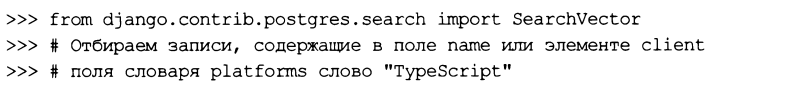
Искомое значение должно содержаться, по крайней мере, в одном из заданных полей:

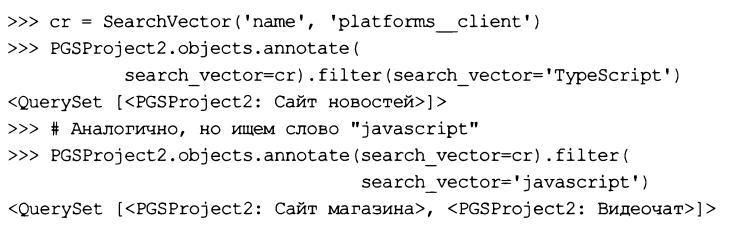
****

Можно указать либо имена полей в виде строк, либо выражения в виде экземпляров класса f.

Созданный экземпляр класса searchvector следует указать в вызове метода Annotate о в именованном параметре, создав тем самым вычисляемое поле. После этого можно выполнить фильтрацию по значению этого поля обычным способом — с помощью метода filter о.

Пример:

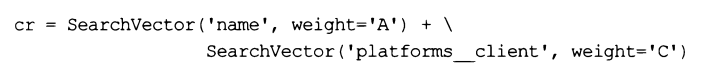


****

Экземпляры класса searchvector также можно объединять оператором +: Cr = Searchvector('name') + Searchvector('platforms\_\_client') параметр config позволяет указать специфическую конфигурацию поиска в формате postgresql в виде либо строки, либо экземпляра класса f с именем поля, в котором хранится эта конфигурация:



Параметр weight задает уровень значимости для поля (полей). Запись, в которой искомое значение присутствует в поле с большей значимостью, станет более релевантной. Уровень значимости указывается в виде предопределенных строковых значений 'А' (максимальный уровень релевантности), 'В', 'С или 'D' (минимальный уровень). Пример:

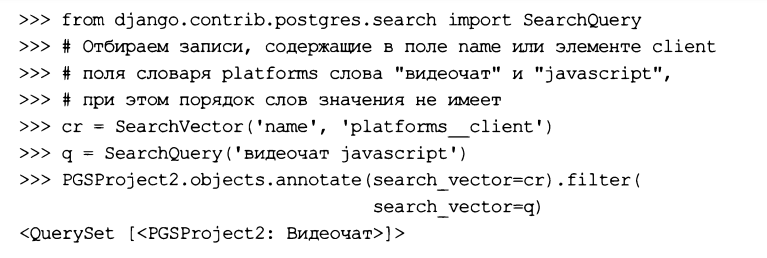


□ searchquery —задает искомое значение:

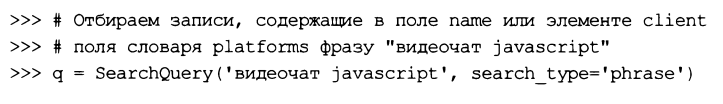


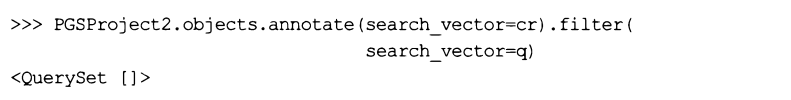
Параметр search type указывает режим поиска в виде одной из строк:

• 'plain' — будут отбираться записи, содержащие все слова, из которых состоит искомое значение, в произвольном порядке (поведение по умолчанию):

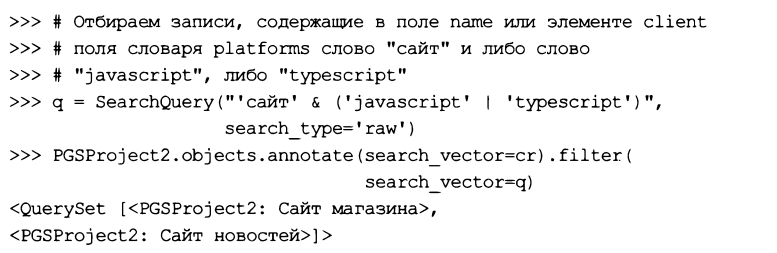
****

• 'phrase' — будут отбираться записи, содержащие заданное искомое значение:

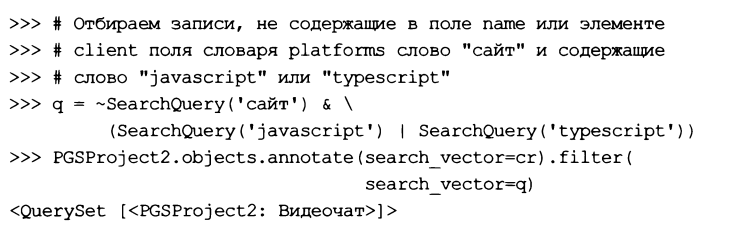




• ’raw’ — искомое значение обрабатывается согласно правилам записи логических выражений postgresql. В таком значении строковые величины берутся в одинарные кавычки, используются логические операторы & (И), | (ИЛИ) и круглые скобки:

****

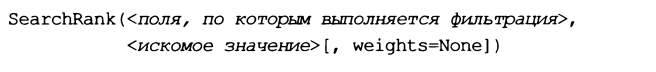
Искомое значение может быть указано в виде комбинации экземпляров класса Searchquery, содержащих одно слово и объединенных логическими операторами & (И), | (ИЛИ) и - (НЕ). Также можно применять круглые скобки. Пример:

****

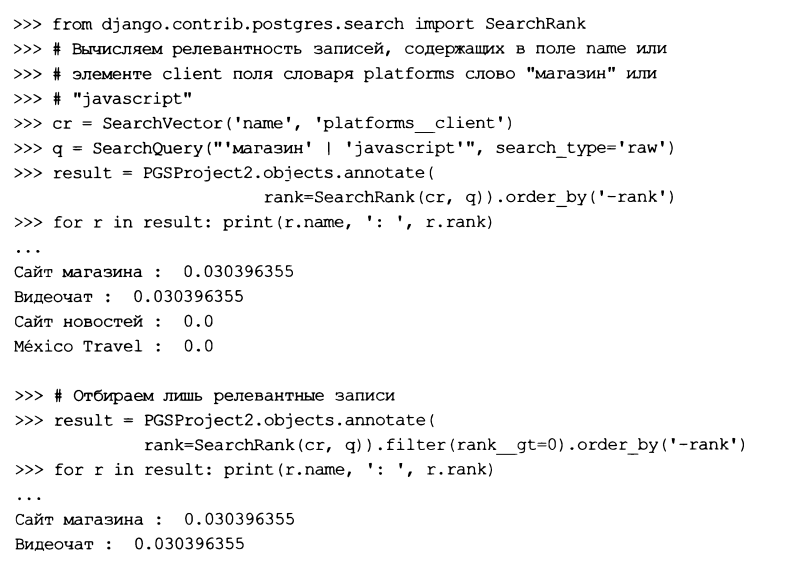
Параметр config указывает специфическую конфигурацию поиска в формате Postgresql.

□ searchrank —создает вычисляемое поле релевантности. Релевантность вычисляется на основе того, сколько раз искомое значение присутствует в содержимом записи, насколько близко отдельные слова искомого значения находятся друг к другу и т. П., и представляется в виде вещественного числа от 1.0 (наиболее подходящие записи) до 0.0 (записи, вообще не удовлетворяющие заданным критериям фильтрации).

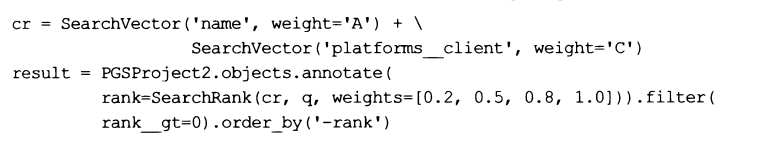
Формат конструктора:



Поле, по которому выполняется фильтрация, задается экземпляром класса Searchvector, а искомое значение—экземпляром класса searchquery. Примеры:

****

Параметр weights позволит задать другие величины уровней значимости полей для предопределенных значений 'А', 'в', 'С' и 'd' (см. описание класса Searchvector). Параметру присваивается список из четырех величин уровней значимости: первая задаст уровень для предопределенного значения 'D\*, вторая— для значения 'С', третья и четвертая— для 'В' и 'А'. Каждая величина уровня значимости указывается в виде вещественного числа от 0.0 (наименее значимое поле) до 1.0 (максимальная значимость). Пример:

****

**Функции СУБД для фильтрации по похожим словам**

Следующие две функции СУБД, классы которых объявлены в модуле Django. Contrib. Postgres. Search, выполняют фильтрацию записей, которые содержат слова, похожие на заданное.

Внимание!

Для использования этих функций следует установить расширение pg trgm.

**Функции СУБД для фильтрации по похожим словам**

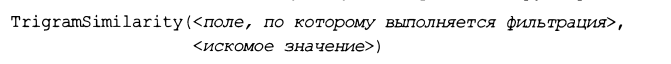
Следующие две функции СУБД, классы которых объявлены в модуле Django. Contrib. Postgres. Search, выполняют фильтрацию записей, которые содержат слова, похожие на заданное.

Внимание!

Для использования этих функций следует установить расширение pg trgm.

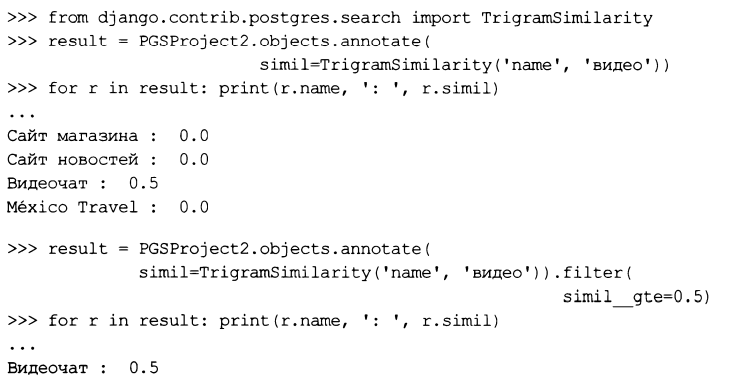
□ Trigramsimilarity— создает вычисляемое поле, хранящее степень похожести СЛОВ, присутствующих в содержимом записи, на заданное искомое значение.

Степень похожести представляется вещественным числом от 1.0 (точное совпадение) до 0.0 (совпадение отсутствует). Формат конструктора:

****

Поле, по которому выполняется фильтрация, МОЖНО указать В виде Либо строки с его именем, Либо экземпляра класса f.

Примеры:

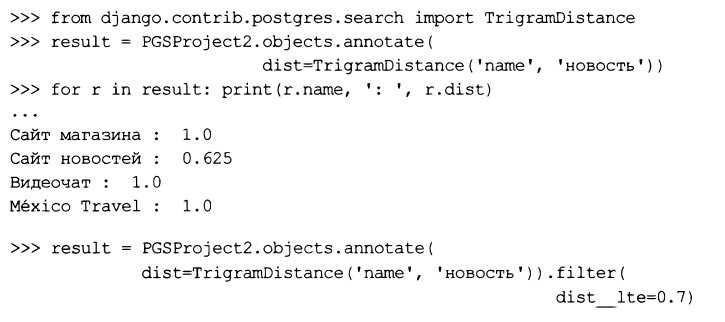
****

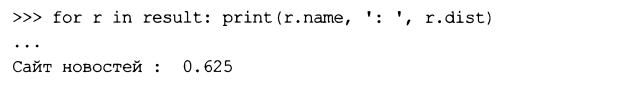
□ trigramdistance— создает вычисляемое поле, хранящее степень расхождения слов, присутствующих в содержимом записи, с заданным искомым значением.

Степень расхождения представляется вещественным числом от 1.0 (полное расхождение) до 0.0 (расхождение отсутствует). Формат конструктора:

Trigramdistance(<поле, по которому выполняется фильтрация>, <искомое значение>) поле, по которому выполняется фильтрация, МОЖНО указать В виде Либо строки с его именем, Либо экземпляра класса f.

Примеры:



****

**Поля форм, специфические для postgresql**

Все классы перечисленных далее полей объявлены в модуле django.contrib.

Postgres.forms!

□ integerrangefieid— поле диапазона, служащее для указания диапазона целочисленных значений.

□ decimairangefieid (начиная с Django 2.2) — поле диапазона, служащее для указания диапазона чисел фиксированной точности в виде объектов типа Decimal из модуля decimal Python.

□ daterangefieid— поле диапазона, служащее для указания диапазона значений даты В виде объектов типа date ИЗ модуля datetime.

□ datetimerangefieid— поле диапазона, служащее для ввода диапазона временных отметок В виде объектов типа datetime ИЗ модуля datetime.

□ simpiearrayfieid— поле списка. Выводит элементы списка в одном поле ввода, отделяя их друг от друга заданным разделителем. Дополнительные параметры:

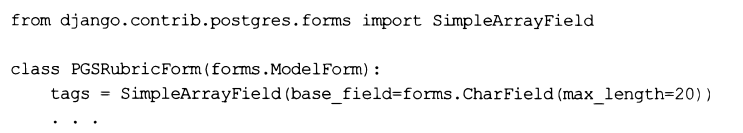
• base fieid —тип элементов списков, сохраняемых в поле. Указывается в виде объекта (не класса!) Соответствующего поля формы;

• delimiter— строка с разделителем, которым отделяются друг от друга отдельные элементы списка (по умолчанию — запятая);

• min iength — минимальный размер списка в виде целого числа (по умолчанию — не ограничен);

• max length — максимальный размер списка в виде целого числа (по умолчанию — не ограничен).

Пример:

****

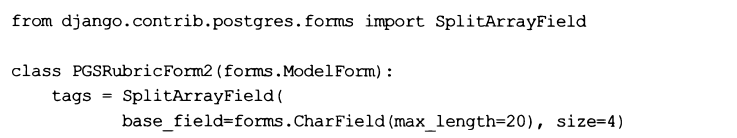
□ spiitarrayfieid —разделенное поле списка. Каждый элемент списка выводится в отдельном поле ввода. Дополнительные параметры:

• base fieid —тип элементов списков, сохраняемых в поле. Указывается в виде объекта (не класса!) Соответствующего поля формы;

• size— количество выводимых на экране элементов списка в виде целого числа. Если превышает количество элементов, имеющихся в списке, то на экран будут выведены пустые поля ввода. Если меньше количества элементов в списке, то на экран будут выведены все элементы;

• remove trailing nuiis — если False, в связанном поле модели будут сохранены все элементы занесенного пользователем списка, даже пустые; если True— пустые элементы в конце списка будут удалены (по умолчанию — False).

Пример:

****

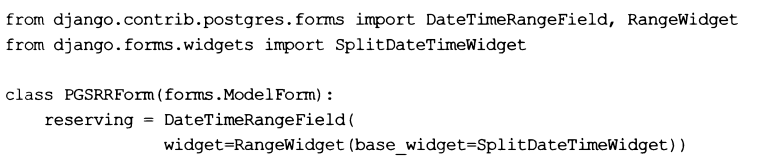
□ hstorefieid — поле словаря. Выводит словарь в виде исходного кода на языке Python в области редактирования.

□ jsonfieid— поле JSON. Выводит содержимое поля в виде его JSON-нотации в области редактирования.

**Элементы управления, специфические для postgresql**

Класс rangewidget ИЗ модуля django.contrib.postgres. Forms представляет элемент управления для ввода диапазона значений какого-либо типа. Параметр base widget задает элемент управления, используемый для ввода каждого из граничных значений диапазона, в виде экземпляра класса нужного элемента управления или ссылки на сам этот класс.

Пример указания для ввода граничных значений диапазона, хранящегося в поле Reserving модели pgsroomreserving, элемента управления splitdatetimewidget:

****

1. Библиотека django-localflavor: дополнительные поля для моделей и форм

**Библиотека django-localflavor: Дополнительные поля для моделей и форм**

Библиотека django-localflavor предоставляет два поля модели, предназначенные для хранения банковских сведений, и несколько классов полей формы, в которые заносятся сведения, специфические для разных стран.

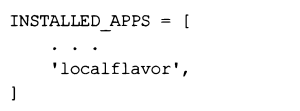
**Установка django-localflavor**

Установка библиотеки выполняется подачей команды:



Помимо django-localflavor, будет установлена библиотека python-stdnum, необходимая для работы.

Далее следует включить присутствующее в составе библиотеки приложение Local flavor в список зарегистрированных в проекте:

****

**Поля модели, предоставляемые django-localflavor**

Оба класса полей объявлены в модуле localflavor.generic.models:

О ibanfieid —хранит номер международного банковского счета (IBAN) в строковом виде. Дополнительные параметры:

• inciude countries — указывает перечень стран, международные банковские номера которых допускается заносить в поле. Значением параметра должен быть кортеж из кодов стран в формате ISO 3166-1 alpha 2, записанных в виде строк (пример: (’ gb •, ’ us', • er ', ’ de ’)). Коды стран в формате ISO 3166-1 Alpha 2 можно найти по интернет-адресу https://ru.wikipedia.org/wiki/ISO-3166-1.

Чтобы разрешить заносить в поле коды всех стран, использующих международные номера IBAN, следует присвоить этому параметру значение переменной IBAN\_SEPA\_COUNTRIES ИЗ модуля localflavor. Generic, countries. Sepa.

Значение по умолчанию — None (в поле разрешается заносить значения международных банковских’номеров из любых стран, даже не использующих международные номера IBAN);

• use nordea extensions — если True, то в поле также можно сохранять номера счета в банке Nordea, если False —нет (по умолчанию — False).

□ bicfieid — хранит банковский идентификационный код (БИК) в строковом виде.

**Поля формы, предоставляемые django-localflavor**

Следующие два класса полей объявлены в модуле localflavor.generic, forms:

□ ibanformfieid — номер международного банковского счета (IBAN) в виде строки. Поддерживает дополнительные параметры include\_countries и use\_nordea\_Extensions, используется полем модели ibanfieid по умолчанию.

□ bicformfieid — банковский идентификационный код (БИК) в виде строки. Используется полем модели bicfield по умолчанию.