django-tree-queries Documentation

Release 0.15-2-g859bf4d

Feinheit AG

Jul 10, 2023

Contents

1 Features and limitations **3**

2 Usage **5**

3 Recipes **7**
- 3.1 Basic models . . . 7
- 3.2 Querying the tree . . . 8
- 3.3 Form fields . . . 9

4 Change log **11**
- 4.1 Next version . . . 11
- 4.2 0.15 (2023-06-19) . . . 11
- 4.3 0.14 (2023-01-30) . . . 11
- 4.4 0.13 (2022-12-08) . . . 12
- 4.5 0.12 (2022-11-30) . . . 12
- 4.6 0.11 (2022-06-10) . . . 12
- 4.7 0.10 (2022-06-07) . . . 12
- 4.8 0.9 (2022-04-01) . . . 12
- 4.9 0.8 (2022-03-09) . . . 12
- 4.10 0.7 (2021-10-31) . . . 12
- 4.11 0.6 (2021-07-21) . . . 12
- 4.12 0.5 (2021-05-12) . . . 13
- 4.13 0.4 (2020-09-13) . . . 13
- 4.14 0.3 (2018-11-15) . . . 13
- 4.15 0.2 (2018-10-04) . . . 13
- 4.16 0.1 (2018-07-30) . . . 14

Query Django model trees using adjacency lists and recursive common table expressions. Supports PostgreSQL, sqlite3 (3.8.3 or higher) and MariaDB (10.2.2 or higher) and MySQL (8.0 or higher, if running without `ONLY_FULL_GROUP_BY`).

Supports Django 2.2 or better, Python 3.6 or better. See the GitHub actions build for more details.

CHAPTER 1

Features and limitations

- Supports only integer and UUID primary keys (for now).
- Allows specifying ordering among siblings.
- Uses the correct definition of depth, where root nodes have a depth of zero.
- The parent foreign key must be named `"parent"` at the moment (but why would you want to name it differently?)
- The fields added by the common table expression always are `tree_depth`, `tree_path` and `tree_ordering`. The names cannot be changed. `tree_depth` is an integer, `tree_path` an array of primary keys and `tree_ordering` an array of values used for ordering nodes within their siblings.
- Besides adding the fields mentioned above the package only adds queryset methods for ordering siblings and filtering ancestors and descendants. Other features may be useful, but will not be added to the package just because it's possible to do so.
- Little code, and relatively simple when compared to other tree management solutions for Django. No redundant values so the only way to end up with corrupt data is by introducing a loop in the tree structure (making it a graph). The `TreeNode` abstract model class has some protection against this.
- Supports only trees with max. 50 levels on MySQL/MariaDB, since those databases do not support arrays and require us to provide a maximum length for the `tree_path` and `tree_ordering` upfront.

Here's a blog post offering some additional insight (hopefully) into the reasons for django-tree-queries' existence.

CHAPTER 2

Usage

- Install `django-tree-queries` using pip.
- Extend `tree_queries.models.TreeNode` or build your own queryset and/or manager using `tree_queries.query.TreeQuerySet`. The `TreeNode` abstract model already contains a `parent` foreign key for your convenience and also uses model validation to protect against loops.
- Call the `with_tree_fields()` queryset method if you require the additional fields respectively the CTE.
- Call the `order_siblings_by("field_name")` queryset method if you want to order tree siblings by a specific model field. Note that Django's standard `order_by()` method isn't supported – nodes are returned according to the depth-first search algorithm. It's not possible to order siblings by more than one field either.
- Create a manager using `TreeQuerySet.as_manager(with_tree_fields=True)` if you want to add tree fields to queries by default.
- Until documentation is more complete I'll have to refer you to the test suite for additional instructions and usage examples, or check the recipes below.

CHAPTER 3

Recipes

3.1 Basic models

The following two examples both extend the `TreeNode` which offers a few agreeable utilities and a model validation method that prevents loops in the tree structure. The common table expression could be hardened against such loops but this would involve a performance hit which we don't want – this is a documented limitation (non-goal) of the library after all.

3.1.1 Basic tree node

```
from tree_queries.models import TreeNode

class Node(TreeNode):
    name = models.CharField(max_length=100)
```

3.1.2 Tree node with ordering among siblings

Nodes with the same parent may be ordered among themselves. The default is to order siblings by their primary key but that's not always very useful.

```
from tree_queries.models import TreeNode

class Node(TreeNode):
    name = models.CharField(max_length=100)
    position = models.PositiveIntegerField(default=0)

    class Meta:
        ordering = ["position"]
```

3.1.3 Add custom methods to queryset

```
from tree_queries.models import TreeNode
from tree_queries.query import TreeQuerySet

class NodeQuerySet(TreeQuerySet):
    def active(self):
        return self.filter(is_active=True)

class Node(TreeNode):
    is_active = models.BooleanField(default=True)

    objects = NodeQuerySet.as_manager()
```

3.2 Querying the tree

All examples assume the `Node` class from above.

3.2.1 Basic usage

```
# Basic usage, disregards the tree structure completely.
nodes = Node.objects.all()

# Fetch nodes in depth-first search order. All nodes will have the
# tree_path, tree_ordering and tree_depth attributes.
nodes = Node.objects.with_tree_fields()

# Fetch any node.
node = Node.objects.order_by("?").first()

# Fetch direct children and include tree fields. (The parent ForeignKey
# specifies related_name="children")
children = node.children.with_tree_fields()

# Fetch all ancestors starting from the root.
ancestors = node.ancestors()

# Fetch all ancestors including self, starting from the root.
ancestors_including_self = node.ancestors(include_self=True)

# Fetch all ancestors starting with the node itself.
ancestry = node.ancestors(include_self=True).reverse()

# Fetch all descendants in depth-first search order, including self.
descendants = node.descendants(include_self=True)

# Temporarily override the ordering by siblings.
nodes = Node.objects.order_siblings_by("id")
```

3.2.2 Breadth-first search

Nobody wants breadth-first search but if you still want it you can achieve it as follows:

```
nodes = Node.objects.with_tree_fields().extra(
    order_by=["__tree.tree_depth", "__tree.tree_ordering"]
)
```

3.2.3 Filter by depth

If you only want nodes from the top two levels:

```
nodes = Node.objects.with_tree_fields().extra(
    where=["__tree.tree_depth <= %s"],
    params=[1],
)
```

3.3 Form fields

django-tree-queries ships a model field and some form fields which augment the default foreign key field and the choice fields with a version where the tree structure is visualized using dashes etc. Those fields are `tree_queries.fields.TreeNodeForeignKey`, `tree_queries.forms.TreeNodeChoiceField`, `tree_queries.forms.TreeNodeMultipleChoiceField`.

CHAPTER 4

Change log

4.1 Next version

4.2 0.15 (2023-06-19)

- Switched to ruff and hatchling.
- Dropped Django 4.0.
- Added Python 3.11.
- Added a `.without_tree_fields()` method which calls `.with_tree_fields(False)` in a way which doesn't trigger the flake8 boolean trap linter.

4.3 0.14 (2023-01-30)

- Changed the behavior around sibling ordering to warn if using `Meta.ordering` where ordering contains more than one field.
- Added Django 4.2a1 to the CI.
- Django 5.0 will require Python 3.10 or better, pruned the CI jobs list.
- Added quoting to the field name for the ordering between siblings so that fields named `order` can be used. Thanks Tao Bojlén!
- Narrowed exception catching when determining whether the ordering field is an integer field or not. Thanks Tao Bojlén.

4.4 0.13 (2022-12-08)

- Made it possible to use tree queries with multiple table inheritance. Thanks Olivier Dalang for the testcases and the initial implementation!

4.5 0.12 (2022-11-30)

- Removed compatibility with Django < 3.2, Python < 3.8.
- Added Django 4.1 to the CI.
- Fixed `.with_tree_fields().explain()` on some databases. Thanks Bryan Culver!

4.6 0.11 (2022-06-10)

- Fixed a crash when running `.with_tree_fields().distinct().count()` by 1. avoiding to select tree fields in distinct subqueries and 2. trusting the testsuite.

4.7 0.10 (2022-06-07)

- Fixed ordering by string fields to actually work correctly in the presence of values of varying length.

4.8 0.9 (2022-04-01)

- Added `TreeQuerySet.order_siblings_by` which allows specifying an ordering for siblings per-query.

4.9 0.8 (2022-03-09)

- Added pre-commit configuration to automatically remove some old-ish code patterns.
- Fixed a compatibility problem with the upcoming Django 4.1.

4.10 0.7 (2021-10-31)

- Added a test with a tree node having a UUID as its primary key.

4.11 0.6 (2021-07-21)

- Fixed `TreeQuerySet.ancestors` to support primary keys not named `id`.
- Changed the tree compiler to only post-process its own database results.
- Added `**kwargs`-passing to `TreeQuery.get_compiler` for compatibility with Django 4.0.

4.12 0.5 (2021-05-12)

- Added support for adding tree fields to queries by default. Create a manager using `TreeQuerySet.as_manager(with_tree_fields=True)`.
- Ensured the availability of the `with_tree_fields` configuration also on subclassed managers, e.g. those used for traversing reverse relations.
- Dropped compatibility with Django 1.8 to avoid adding workarounds to the testsuite.
- Made it possible to use django-tree-queries in more situations involving JOINs. Thanks Safa Alfulaij for the contribution!

4.13 0.4 (2020-09-13)

- Fixed a grave bug where a position of `110` would be sorted before `20` for obvious reasons.
- Added a custom `TreeNodeForeignKey.deconstruct` method to avoid migrations because of changing field types.
- Removed one case of unnecessary fumbling in `Query`'s internals making things needlessly harder than they need to be. Made django-tree-queries compatible with Django's master branch.
- Removed Python 3.4 from the Travis CI job list.
- Dropped the conversion of primary keys to text on PostgreSQL. It's a documented constraint that django-tree-queries only supports integer primary keys, therefore the conversion wasn't necessary at all.
- Reverted to using integer arrays on PostgreSQL for ordering if possible instead of always converting everything to padded strings.

4.14 0.3 (2018-11-15)

- Added a `label_from_instance` override to the form fields.
- Removed the limitation that nodes can only be ordered using an integer field within their siblings.
- Changed the representation of `tree_path` and `tree_ordering` used on MySQL/MariaDB and sqlite3. Also made it clear that the representation isn't part of the public interface of this package.

4.15 0.2 (2018-10-04)

- Added an optional argument to `TreeQuerySet.with_tree_fields()` to allow reverting to a standard queryset (without tree fields).
- Added `tree_queries.fields.TreeNodeForeignKey`, `tree_queries.forms.TreeNodeChoiceField` and `tree_queries.forms.TreeNodeMultipleChoiceField` with node depth visualization.
- Dropped Python 3.4 from the CI.

4.16 0.1 (2018-07-30)

- Initial release!