Combo - Development #60018

Page.objects.get_by_natural_key ne correspond pas à Page.natural_key

22 décembre 2021 17:40 - Valentin Deniaud

Statut:	En cours	**Début:**	22 décembre 2021
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

J'ai l'impression que le bug est présent depuis l'implémentation initiale en 2016, #8598.

Si on a une hiérarchie A > B > C, C.natural_key() renvoie 'a/b', pourtant get_by_natural_key ne regarde que 'b'.
Donc gros fail si à côté il y a D > E > F avec E qui a le même slug que B, F.natural_key() renvoie 'd/b' et crash au moment de get_by_natural_key.

Historique

#1 - 22 décembre 2021 17:53 - Frédéric Péters

En passant, si tu veux taper une vraie colonne "path" ça évitera sans doute des bugs de ce type et devrait réduire les requêtes. (aussi on peut s'inspirer de choses comme https://406.ch/writing/django-tree-queries/ mais n'ajoutons pas de dépendance).

#2 - 23 décembre 2021 11:42 - Valentin Deniaud

Frédéric Péters a écrit :

> En passant, si tu veux taper une vraie colonne "path" ça évitera sans doute des bugs de ce type et devrait réduire les requêtes.

OK pour voir les choses en plus grand, j'avais tapé vite fait

```
     def get_by_natural_key(self, path):
         parts = [x for x in path.strip('/').split('/') if x] or ['index']
-        return self.get(slug=parts[-1])
+        lookups = {'slug': parts[-1]}
+        for i, part in enumerate(parts[:-1], 1):
+            lookups['%s%s' % ('parent__' * (len(parts) - i), 'slug')] = part
+        return self.get(**lookups)
```

qui semblait fonctionner mais ça a une forte odeur de rustine.

Par contre tu as quoi en tête en écrivant « vraie colonne path » ? Colonne de quel type qui contiendrait quoi ?

> (aussi on peut s'inspirer de choses comme https://406.ch/writing/django-tree-queries/ mais n'ajoutons pas de dépendance).

Là je ne sais pas si cette parenthèse vient préciser l'approche « vraie colonne path » ou vient suggérer une alternative ?

En tout cas ce projet me semble adapté, il y a juste deux méthodes descendants/ancestors qui récupèrent en une requête ce qu'on fait actuellement en autant de requêtes qu'il y a de résultats. Tout cela grâce à une requête SQL écrite en dur, mais je suis incompétent pour comprendre comment elle marche. Par contre je peux la copier-coller et la faire marcher dans le code de combo sans problème, mais ça va peut-être au delà de ce que tu entendais par « s'inspirer » ?

#3 - 23 décembre 2021 12:32 - Frédéric Péters

Par vraie colonne path j'imaginais une path = models.CharField(length=...) et dedans il y aurait /foo/bar/ pour une page qui aurait comme "bar" et comme parente une page avec "foo" comme slug.

La référence éventuelle à django-tree-queries c'était pour dire que si dénormaliser ainsi bêtement l'info posait question il y avait peut-être là de l'inspiration.

#4 - 23 décembre 2021 18:17 - Valentin Deniaud

Frédéric Péters a écrit :

> Par vraie colonne path j'imaginais une path = models.CharField(length=...) et dedans il y aurait /foo/bar/ pour une page qui aurait comme "bar" et comme parente une page avec "foo" comme slug.
>
> La référence éventuelle à django-tree-queries c'était pour dire que si dénormaliser ainsi bêtement l'info posait question il y avait peut-être là de l'inspiration.

À y repenser je vais plutôt partir sur la base de la requête de django-tree-queries, on sera mieux à maintenir 50 lignes de SQL plutôt que de continuer à se traîner du bricolage à base de chaînes de caractères pour représenter des liens entre les objets.
Il y a même moyen que ça fasse un patch plus petit et sans besoin de migration.

#5 - 11 janvier 2022 11:17 - Valentin Deniaud

- Fichier 0001-general-use-tree-queries-for-handling-page-hierarchy.patch ajouté

- Fichier 0002-data-fix-get_by_natural_key-for-page-60018.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Voilà pour ce que ça donne, c'est pas fou niveau nombre de requêtes mais je n'ai pour l'instant rien cherché à optimiser, il y aurait sûrement des simplifications possibles dans get_with_hierarchy_attributes ou get_page_from_url_parts.

Branche basée sur #59509, si le patch proposé là bas ne passe pas le test de 0002 n'est pas réaliste.

#6 - 25 juillet 2022 18:53 - Pierre Ducroquet

- Statut changé de Solution proposée à En cours

Je résume une discussion que nous avons eu avec Valentin au sujet de ce patch.
Le TreeQuerySet introduit ici est certes intéressant, mais sur les plus gros clients cela augmenterait considérablement le coût des requêtes parce que le code est trop limité.
Il s'agit tout de même, à chaque fois que cela est demandé, de calculer l'intégralité des relations de hiérarchie de chaque objet, pour ensuite en extraire la hiérarchie d'une page, et derrière pouvoir faire des requêtes (chercher les parents, les enfants...). De même pour la recherche par tableau de slugs, c'est l'ensemble des tableaux qui sont calculés alors que cela n'est nullement nécessaire.

J'entends bien que raw() ne soit pas apprécié (en tout cas pas à la juste valeur du SQL, mais passons :)), mais ici on pourrait avoir des requêtes bien plus efficaces et assez lisibles pour l'ensemble des besoins.

Autre alternative, si l'on veut éviter tout ça : il existe un type spécifique dans PostgreSQL pour représenter les hiérarchies, ltree. À plusieurs reprises je m'en suis servi pour traduire avec des triggers une relation parent_id / id. Cela pourrait être ici une solution : ajouter une colonne ltree maintenue par des triggers et effectuer des requêtes dessus. (et au passage, ajouter une contrainte d'unicité (parent_id, slug), parce que sinon ça va être compliqué)

#7 - 17 mai 2023 16:40 - Valentin Deniaud

- Assigné à Valentin Deniaud supprimé

Fichiers

0001-general-use-tree-queries-for-handling-page-hierarchy.patch	10,2 ko	11 janvier 2022	Valentin Deniaud
0002-data-fix-get_by_natural_key-for-page-60018.patch	4,27 ko	11 janvier 2022	Valentin Deniaud