Passerelle - Development #43501

avoir une commande ensure-jsonb pour vérifier et corriger que les colonnes JSONB en sont bien

01 juin 2020 10:10 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	01 juin 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Serghei Mihai	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Voir #43492 pour les rapports sur l'état des tables csvdatasource en recette et prod de notre SaaS.

Révisions associées

Révision 5b4dd243 - 17 juin 2020 16:29 - Serghei Mihai

management: add command to ensure all JSONField fields have correct db type (#43501)

Historique

#2 - 01 juin 2020 10:30 - Benjamin Dauvergne

- Sujet changé de avoir une commande ensure-jsonb pour vérifier et corriger que les colonnes JSONB en son bien à avoir une commande ensure-jsonb pour vérifier et corriger que les colonnes JSONB en sont bien

#4 - 02 juin 2020 14:40 - Serghei Mihai

- Statut changé de Nouveau à En cours

- Assigné à mis à Serghei Mihai

#5 - 02 juin 2020 15:03 - Valentin Deniaud

- Fichier 0001-wip-jsonb-command.patch ajouté

- Patch proposed changé de Non à Oui

Je regardais en parallèle, j'ai fait la partie relou ie lister tous les champs concernés si ça peut t'être utile (pas tenté d'exécuter le code à ce stade qui a 0 chance de marcher).

#6 - 02 juin 2020 15:26 - Benjamin Dauvergne

Pourquoi ne pas utiliser l'attribut _meta des modèles pour trouver ces champs (ça t'aurait éviter la partie relou) ?

https://docs.djangoproject.com/fr/1.11/ref/models/meta/

#7 - 02 juin 2020 15:33 - Serghei Mihai

Je suis en train de bosser dessus.
Valentin m'a filé le patch avec son idée, qui ne sera pas forcément la mienne.

#8 - 02 juin 2020 15:43 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Pourquoi ne pas utiliser l'attribut _meta des modèles pour trouver ces champs (ça t'aurait éviter la partie relou) ?
>
> https://docs.djangoproject.com/fr/1.11/ref/models/meta/

Je ne connaissais pas, merci pour la doc (même si en l'occurrence on peut trouver des avantages à avoir l'info en statique, restriction explicite aux seuls champs potentiellement buggés, moins de risque de comportement inattendu, rapidité, etc).

#9 - 02 juin 2020 17:01 - Serghei Mihai

- Fichier 0001-management-add-command-to-ensure-all-JSONField-field.patch ajouté

Il manque les tests, mais un premier jet.

#10 - 08 juin 2020 09:43 - Serghei Mihai

- Fichier 0001-management-add-command-to-ensure-all-JSONField-field.patch ajouté

- Statut changé de En cours à Solution proposée

Il faut taper dans la base pour connaître le vrai type du champ, l'attribut db_type est toujours de celui que Django croit avoir tapé.

#11 - 08 juin 2020 15:57 - Valentin Deniaud

Sur le fond tout me paraît bien, sur la forme les imports ne sont pas dans le bon ordre et il y a trop de lignes blanches.

Idée d'amélioration, ça serait bien que la migration (dans utils/db.py), qui fait la même chose que la commande, utilise aussi le même code (notamment le check du db_type avant la conversion était présent, puis a été retiré car buggé, c'est l'occasion de le remettre). Donc juste copier coller du code de la commande vers la migration.

#12 - 09 juin 2020 10:03 - Serghei Mihai

- Fichier 0001-management-add-command-to-ensure-all-JSONField-field.patch ajouté

A mon avis ça ne sert à rien de copier/coller le code de vérification du type de la colonne car:

```
ALTER TABLE {table} ALTER COLUMN {col} TYPE jsonb USING {col}::jsonb;
```

passe même si la colonne a le bon type.

Ainsi on tape une seule requete par champ lors des migrations au lieu de, potentiellement, deux.

Imports, etc corrigés.

#13 - 09 juin 2020 12:09 - Valentin Deniaud

Serghei Mihai a écrit :

> A mon avis ça ne sert à rien de copier/coller le code de vérification du type de la colonne car:
> [...]
> passe même si la colonne a le bon type.
>
> Ainsi on tape une seule requete par champ lors des migrations au lieu de, potentiellement, deux.

OK mais mon problème c'est plutôt de rendre clair le fait que les deux font la même chose, le code est quand même sacrément différent actuellement. Notamment la boucle for que la commande a en plus, on se demande bien pourquoi il y en a besoin.
Et en me posant cette question je me rends compte qu'il n'y en a pas besoin :

```
'SELECT table_schema, data_type FROM information_schema.columns WHERE table_name=%s AND column_name=%s'
```

pourrait être

```
'SELECT data_type FROM information_schema.columns WHERE table_name=%s AND column_name=%s AND table_schema=%s'
% (table_schema=connection.schema_name, ...)
```

Et normalement juste un résultat, plus de boucle for.
Et il y a même un bug sur comment c'est fait actuellement, parce que ce que la commande lancée sur un tenant opère sur tous les tenants, c'est mal (et lancée avec --all-tenants, elle fait 1 fois les n opérations nécessaires, puis n - 1 itérations inutiles). Bon, je dis ça sans avoir beaucoup testé.

> Imports, etc corrigés.

Je ne sais pas comment tu classes les imports, pour moi ça doit ressembler à

```
import pytest

from django.core.files import File
from django.core.management import call_command
from django.db import connection
from django.utils.six import BytesIO

from passerelle.apps.csvdatasource.models import CsvDataSource
```

```
from passerelle.contrib.teamnet_axel.models import TeamnetAxel
```

par exemple dans le fichier de test, pas de double saut de ligne après l'import de pytest, imports django par ordre alphabétique et sans sauts de lignes entre.

#14 - 09 juin 2020 17:02 - Serghei Mihai

Valentin Deniaud a écrit :

> Et en me posant cette question je me rends compte qu'il n'y en a pas besoin :
> [...]
> pourrait être
> SELECT data_type FROM information_schema.columns WHERE table_name=%s AND column_name=%s AND table_schema=%s' %
> (table_schema=connection.schema_name, ...)

Dans ce contexte connection.schema_name n'existe pas. D'ou le SELECT table_schema, data_type pour avoir le schema et le type de la colonne et ceci sans avoir à vérifier si on est en contexte multitenant ou pas.

#15 - 09 juin 2020 17:30 - Valentin Deniaud

OK je comprends, la commande n'est pas faite pour marcher avec tenant_command. Je sais pas si ça passe, toutes les commandes que j'ai croisées se lancent avec tenant_command, pour moi la bonne approche c'est celle-ci en exploitant connection.schema_name.
Je vais laisser quelqu'un d'autre valider :/

#16 - 09 juin 2020 17:34 - Serghei Mihai

Elle fonctionne avec tenant_command aussi. Et comme toi au début je ne réflechissais que multitenant (connection.get_tenant().schema_name dans le premier patch brouillon joint au ticket).
Mais la commande doit pouvoir s'éxecuter aussi sans être en multitenant.

#17 - 17 juin 2020 16:26 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Pousse donc ça ainsi.

#18 - 17 juin 2020 16:29 - Serghei Mihai

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 5b4dd2432c36d7a4f1f97af6688978bae2b06d66 (HEAD -> master, origin/master, origin/HEAD)
Author: Serghei Mihai <smihai@entrouvert.com>
Date:   Tue Jun 2 17:00:17 2020 +0200

    management: add command to ensure all JSONField fields have correct db type (#43501)
```

#19 - 19 juin 2020 08:16 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

0001-wip-jsonb-command.patch	3,28 ko	02 juin 2020	Valentin Deniaud
0001-management-add-command-to-ensure-all-JSONField-field.patch	2,7 ko	02 juin 2020	Serghei Mihai
0001-management-add-command-to-ensure-all-JSONField-field.patch	5,48 ko	08 juin 2020	Serghei Mihai
0001-management-add-command-to-ensure-all-JSONField-field.patch	5,47 ko	09 juin 2020	Serghei Mihai