Passerelle - Documentation #26331

Faire passer le message comme quoi le connecteur CSV n'est pas approprié pour des fichiers avec des dizaines de milliers de lignes

12 septembre 2018 09:33 - Frédéric Péters

Statut:	Nouveau	**Début:**	12 septembre 2018
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	

Description

Parce qu'actuellement c'est parfois utilisé ainsi et c'est n'importe quoi et il faudrait le dire.

Historique

#1 - 10 octobre 2018 11:47 - Mikaël Ates (de retour le 29 avril)

Il arrive que soit fournis des fichiers tableurs conséquents par certaines sources,
https://www.data.gouv.fr/fr/datasets/liste-et-adresses-des-etablissements-scolaires/ par exemple.

Quelles sont les alternatives ? Est-ce que si ces données étaient dans un JSON Data Store ce serait plus rapide ? Est-il envisageable d'alimenter un JDS à partir d'un fichier CSV ?

#2 - 10 octobre 2018 14:16 - Thomas Noël

Je pense que ce n'est pas tant une question de performance (aujourd'hui les csv sont interprétés et stockés en base) qu'une question de stabilité de la source. Une incohérence ou inconsistance dans un CSV ça arrive très vite, et boum. Alors sur des dizaines de milliers de lignes, c'est pire.

Mon avis : pour les données qui viennent de https://www.data.gouv.fr comme celles qui viennent de api.gouv.fr, on devrait avoir des connecteurs «dédiés» et proposés par défaut, un peu comme pour la BAN.

#4 - 06 mars 2019 12:37 - Benjamin Dauvergne

On peut aussi s'auto-suggérer que c'est la loose de ne pas savoir gérer une liste de 36000 trucs en 2019 correctement (dans le sens où on maintiendrait un dico en RAM avec deux dicos d'index et ça marcherait); en Django 1.11 on devrait facilement s'en sortir avec un JSONField indexé. Un plan pourrait être d'importer la couche de compatibilité JSONField d'authentic et de l'utiliser, puis quand on détecter django>1.11 d'ajouter une déclaration d'index à TableRow:

```
class Meta:
    if DJ111:
        indexes = [
            django.contrib.postgres.indexes.GinIndex(fields=('data',')),
        ]
```

ça c'est la partie facile, ensuite il faudra voir pour détecter les filtres simples du style code_postal == query.get('code_postal') pour en faire des TableRow.objects.filter(data__xxx=query.get('blabla')), ça peut dans un premier n'être qu'une regex, pour le cas 99% qui nous intéresse (je cherche un code postal).

#5 - 06 mars 2019 13:16 - Benjamin Dauvergne

Bien sûr ça n'empêche pas d'avoir aussi des connecteurs spécifiques du style liste des écoles et des codes postaux aussi bien sûr (mais plus pour des questions de simplicité à trouver/mettre en place que pour les performances, j'espère).