# w.c.s. - Development #57009

## Les champs avec une apostrophe unicode sont mal indexés (l'article est collé au mot)

16 septembre 2021 15:51 - Benjamin Dauvergne

| | | | |
|---|---|---|---|
| **Statut:** | Fermé | **Début:** | 16 septembre 2021 |
| **Priorité:** | Normal | **Echéance:** | |
| **Assigné à:** | Frédéric Péters | **% réalisé:** | 0% |
| **Catégorie:** | | **Temps estimé:** | 0:00 heure |
| **Version cible:** | | | |
| **Patch proposed:** | Oui | **Planning:** | Non |

**Description**

apostoprhe unicode (https://unicode-table.com/fr/2019/)

Cf. #57006

C'est due à la normalisation unicode par FtsMatch.get_fts_value()[1] :

```
In [3]: FtsMatch.get_fts_value('Allée de l’Allier 13013 Marseille')
Out[3]: 'Allee de lAllier 13013 Marseille'
```

Ça supprime l'apostrophe unicode et on se retrouve à indexer "lAllier" au lieu de "Allier". Je me demande si la fonction ts_vector() toute seule ne ferait pas un meilleur boulot sans qu'on lui mâche:

```
wcs_demarches_departement13_test_entrouvert_org=# select to_tsvector('Allée de l’Allier 13013 Mars
eille');
-[ RECORD 1 ]-------------------------------------
to_tsvector | '13013':5 'alli':4 'allé':1 'marseil':6
```

(bon visiblement pas sur les accents)

ou alors unidecode :

```
In [1]: import unidecode

In [2]: unidecode.unidecode('Allée de l’Allier 13013 Marseille')
Out[2]: "Allee de l'Allier 13013 Marseille"
```

[1]

```
    @classmethod
    def get_fts_value(cls, value):
        return unicodedata.normalize('NFKD', value).encode('ascii', 'ignore').decode('ascii')
```

### Révisions associées

**Révision c1ccadde - 20 octobre 2021 15:09 - Frédéric Péters**

sql: switch fts normalization to unidecode (#57009)

### Historique

**#2 - 16 septembre 2021 15:51 - Benjamin Dauvergne**

*- Description mis à jour*

**#3 - 17 septembre 2021 11:10 - Anonyme**

Benjamin Dauvergne a écrit :

> Ça supprime l'apostrophe unicode et on se retrouve à indexer "lAllier" au lieu de "Allier". Je me demande si la fonction ts_vector() toute seule ne ferait pas un meilleur boulot sans qu'on lui mâche:
>
> [...]

(bon visiblement pas sur les accents)

Les fonctions FTS de PostgreSQL dépendent d'une configuration. Par défaut plusieurs langues sont définies. Quand on ne précise pas de configuration, c'est par défaut l'anglais qui est utilisé (`default_text_search_config` sur la base de données). Ici, vu les résultats, c'est déjà le français. Par contre la configuration par défaut ne fait pas de travail de transformation sur les accents. On peut passer sur une configuration FTS plus avancée qui contienne la suppression des accents, si vous considérez ces derniers comme indésirables.

0) Charger l'extension unaccent (postgresql-contrib)
```
test=# CREATE EXTENSION unaccent;
```

1) Créer une configuration alternative

```
test=# CREATE TEXT SEARCH CONFIGURATION fr ( COPY = french );
CREATE TEXT SEARCH CONFIGURATION
Time: 19.081 ms
test=# ALTER TEXT SEARCH CONFIGURATION fr ALTER MAPPING FOR hword, hword_part, word WITH unaccent, french_stem;
ALTER TEXT SEARCH CONFIGURATION
Time: 2.915 ms
```

2) L'utiliser…

```
test=# SELECT to_tsvector('fr', 'Allée de l'Allier 13013 Marseille');
to_tsvector
-----------------------------------------
'13013':5 'alle':1 'alli':4 'marseil':6
(1 row)

Time: 0.247 ms
```

On peut changer la règle par défaut de la base également :
```
ALTER DATABASE test SET default_text_search_config TO fr;
```

Par contre il ne faut pas oublier que si des index ont été définis sur la configuration «french», il faudra les recréer avec la configuration «fr».

#### #4 - 17 septembre 2021 11:18 - Benjamin Dauvergne

Misère :) Je sens que c'est la vrai bonne solution, mais c'est beaucoup plus compliqué que d'utiliser unidecode puis reindex (par lequel on devra passer vu que la chaîne source à indexer est générée coté python) :/

#### #5 - 17 septembre 2021 11:24 - Anonyme

Benjamin Dauvergne a écrit :

> Misère :) Je sens que c'est la vrai bonne solution, mais c'est beaucoup plus compliqué que d'utiliser unidecode puis reindex (par lequel on devra passer vu que la chaîne source à indexer est générée coté python) :/

Hum, j'avoue avoir du mal à saisir la complexité de cette solution. Ce n'est que créer un objet dans la base. Après si c'est ce point qui bloque, unaccent propose aussi des fonctions.

```
test=# select to_tsvector('Allée de l'Allier 13013 Marseille');
               to_tsvector
-----------------------------------------
 '13013':5 'alli':4 'allé':1 'marseil':6
(1 row)

test=# select to_tsvector(unaccent('Allée de l'Allier 13013 Marseille'));
               to_tsvector
-----------------------------------------
 '13013':5 'alle':1 'alli':4 'marseil':6
(1 row)
```

#### #6 - 17 septembre 2021 11:35 - Benjamin Dauvergne

Pierre Ducroquet a écrit :

> Benjamin Dauvergne a écrit :
>
> > Misère :) Je sens que c'est la vrai bonne solution, mais c'est beaucoup plus compliqué que d'utiliser unidecode puis reindex (par lequel on devra passer vu que la chaîne source à indexer est générée coté python) :/
>
> Hum, j'avoue avoir du mal à saisir la complexité de cette solution. Ce n'est que créer un objet dans la base. Après si c'est ce point qui bloque, unaccent propose aussi des fonctions.
>
> [...]

On a une base par client sur le SaaS et divers déploiements on premise, donc ça suppose en vrai d'écrire une migration correspondante dans w.c.s. (en espérant que tout ça puisse être fait avec l'utilisateur de la base et pas un utilisateur root, mais je pense que c'est le cas) pour :
1. créer la nouvelle configuration FTS
2. la coller par défaut sur la base
3. dropper l'index FTS
4. recréer l'index FTS
5. updater tous les formdata_*.fts avec le ts_vector() calculé qui va bien (ça veut lire relire toutes les lignes de toutes ces tables et faire un FormData.store()) parce qu'on a pas de source du texte indexé en dehors de cette colonne "fts"

En adoptant unidecode comme "pré-filtre" FTS on a juste à modifier le pré-filtre actuel (qui en gros fait salement fts_string.encode('ascii', 'ignore').decode()) et jouer le point 5. Évidemment c'est moins correct que de se reposer sur les capacités natives de PG, mais c'est pour ça que je dis que c'est plus simple (de notre point de vue, on est pas des gros fans d'aller bosser dans postgres en fait, c'est pour ça qu'on t'embauche :) ).

#### #7 - 20 octobre 2021 09:01 - Frédéric Péters

*- Fichier 0001-sql-switch-fts-normalization-to-unidecode-57009.patch ajouté*

*- Statut changé de Nouveau à Solution proposée*

*- Assigné à mis à Frédéric Péters*

*- Patch proposed changé de Non à Oui*

> juste à modifier le pré-filtre actuel

Le patch en question.

#### #8 - 20 octobre 2021 09:56 - Thomas Noël

*- Statut changé de Solution proposée à Solution validée*

Nettement plus joli/rassurant.

#### #9 - 20 octobre 2021 15:11 - Frédéric Péters

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

```
commit c1ccadde40c74c59088845041eacab2d221c8d61
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Wed Oct 20 08:58:49 2021 +0200

    sql: switch fts normalization to unidecode (#57009)
```

#### #10 - 20 octobre 2021 19:17 - Frédéric Péters

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

### Fichiers

| | | | |
|---|---|---|---|
| 0001-sql-switch-fts-normalization-to-unidecode-57009.patch | 2,47 ko | 20 octobre 2021 | Frédéric Péters |