w.c.s. - Development #65745

test.saas: tables __evolutions avec des dizaines de milliers d'entrées par formdata

30 mai 2022 08:13 - Pierre Ducroquet

Statut:	Fermé	**Début:**	30 mai 2022
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	Non

Description

On a sur test.saas plusieurs environnements qui montrent des comportements aberrants avec des tables __evolutions qui montent en volume en permanence.

```
test_montpellier2_test_eservices_montpellier3m_fr=# select formdata_id, count(*) from formdata_16_
incident_sur_le_reseau_d_eaux__evolutions group by 1;
 formdata_id | count
-------------+--------
           1 |  72098
           3 | 116918
           4 |  43303
           5 | 112681
           6 | 116704
           7 |  50521
           8 | 116684
           9 |  10188
(8 lignes)
```

À noter : ces chiffres augmentent en permanence, toutes les heures, suite à un cron de wcs.
Je suppose que cela est dû à un mauvais paramétrage dans l'application, mais je pense qu'une limite devrait être mise pour passer ces cas en erreur avant d'accumuler tant d'entrées inutilisables dans la base.
(Et prenons les choses du bon côté : cela permet de mettre en lumière les points optimisables dans l'application par rapport à la base de données, cf #65744 par exemple)

Demandes liées:

Lié à w.c.s. - Development #65744: Performance: formdata.store va faire un up...	**Fermé**	**30 mai 2022**

Historique

#1 - 30 mai 2022 08:21 - Pierre Ducroquet

- Lié à Development #65744: Performance: formdata.store va faire un update sur toutes les instances d'evolution ajouté

#2 - 30 mai 2022 08:44 - Frédéric Péters

Ici sur la plateforme de recette de ce que je vois rapidement c'est supposé monter la criticité après 3 jours mais, 1/ ça a été raccourci à 20 minutes sans doute pour tester, 2/ ça le refait sans fin plutôt que s'arrêter une fois la criticité atteinte.

Mais clairement il faudrait détecter/remonter ces workflows qui tournent sans fin.

#10 - 05 juin 2022 16:25 - Benjamin Dauvergne

Il me semble qu'avant les sauts sur le même statut ne généraient pas de nouvelles évolutions quand certaines conditions étaient réunies, dans le code :

```
        elif (
            self.status == status
            and self.evolution[-1].status == status
            and not self.evolution[-1].comment
            and not [
                x for x in self.evolution[-1].parts or [] if not isinstance(x, ActionsTracingEvolutionPart)
            ]
        ):
```

```
            # if status do not change and last evolution is empty,
            # just update last jump time on last evolution, do not add one
            self.evolution[-1].last_jump_datetime = datetime.datetime.now()
            self.store()
            return
```

serait-il possible qu'à cause d'un changement cette condition soit moins souvent remplie ?

~~Si je prends le cas Isère il semble que ce soit l'action Alerte qui cause souci, sur Strasbourg c'est l'utilisation de deux status au lieu de boucler sur le même statut pour le polling. Une source possible d'augmentation des évolution c'est quand "Enregistrer les erreurs" est coché et que sur un cas non passant du polling le web-service retourne {'err': 1} au lieu d'une réponse normale (mais je n'ai pas identifié de tel problème sur les deux cas remontés).~~

Pour Isère je dis des bêtises, Alerte n'enregistre pas de Part, c'est simplement les ActionsTracingEvolutionPart qui s'accumulent dans le même statut.

#11 - 09 juillet 2023 21:12 - Frédéric Péters

- Statut changé de Nouveau à Fermé

> c'est simplement les ActionsTracingEvolutionPart qui s'accumulent dans le même statut.

Les traces sont désormais conservées à part (#72802).

Pour les autres situations, des tickets spécifiques semblent avoir été créés (comme suggéré dans #65745#note-6).