w.c.s. - Development #65744

Performance: formdata.store va faire un update sur toutes les instances d'evolution

30 mai 2022 08:13 - Pierre Ducroquet

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

Description

Même sans modification de formdata.evolutions, un update est fait systématiquement sur chaque Evolution lors d'un store sur formdata.
Or un UPDATE, même s'il ne change pas les données, n'est par défaut pas optimisé dans PostgreSQL afin de conserver ses effets de bord.
Étant donné qu'on a plusieurs cas où des milliers, voire des dizaines de milliers d'objets Evolution rattachés à un formdata, on arrive sur notamment l'environnement saas.test à une utilisation en permanence d'un CPU sur le serveur PostgreSQL uniquement pour traiter ces UPDATE en boucle.

Demandes liées:

Lié à w.c.s. - Development #65745: test.saas: tables __evolutions avec des di...	**Fermé**	**30 mai 2022**

Révisions associées

Révision aebf5a59 - 30 juin 2022 13:47 - Pierre Ducroquet

sql: only the last Evolution object can be changed (#65744)

Historique

#1 - 30 mai 2022 08:13 - Pierre Ducroquet

- Assigné à mis à Pierre Ducroquet

#2 - 30 mai 2022 08:21 - Pierre Ducroquet

- Lié à Development #65745: test.saas: tables __evolutions avec des dizaines de milliers d'entrées par formdata ajouté

#3 - 30 mai 2022 18:06 - Pierre Ducroquet

- Fichier 0001-sql-don-t-update-when-Evolution-object-is-unchanged-.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Voilà le patch. Il passe les tests unitaires, et a priori devrait réduire la charge sur les PGs. Avec le pgbouncer des machines de test, on le verra facilement.

#4 - 30 mai 2022 18:25 - Benjamin Dauvergne

Il me semble que le premier et nouveau Evolution.__init__ est écrasé par le deuxième (j'ai juste regarder le patch vite fait).

#5 - 30 mai 2022 21:26 - Pierre Ducroquet

- Fichier 0002-Fix-duplicate-__init__.patch ajouté

Silly me, comme on dit.
En complément du coup (et ça passe toujours le jenkins)

#6 - 31 mai 2022 10:12 - Frédéric Péters

Je n'aime pas trop la multiplication des getter/setter, sur un tas d'attributs qui ne peuvent pas bouger.

Par contre, parts est une liste dont le contenu peut bouger, et le getter/setter va passer à côté de ça.

Je serais pour une approche moins technique et plutôt analyser ce qui peut effectivement bouger sur un objet Evolution (sans regarder je dirais que ça concerne juste last_jump_datetime et le contenu de parts).

#7 - 31 mai 2022 10:24 - Frédéric Péters

> Je serais pour une approche moins technique et plutôt analyser ce qui peut effectivement bouger sur un objet Evolution (sans regarder je dirais que ça concerne juste last_jump_datetime et le contenu de parts).

Et noter que ce sera toujours le dernier evolution qui sera modifié, jamais les précédents, et peut-être que juste se baser là-dessus serait totalement satisfaisant.

#8 - 07 juin 2022 10:00 - Pierre Ducroquet

- Fichier 0001-sql-only-the-last-Evolution-object-can-be-changed-65.patch ajouté

Patch modifié du coup : update uniquement sur la dernière instance de l'objet evolution, et on oublie toute tentative de savoir si les objets ont été modifiés ou non.

#9 - 19 juin 2022 15:04 - Frédéric Péters

J'ai l'impression qu'on peut plus simplement :

```
+            # iterate in reverse order to only save new and last evolution
+            for evo in reversed(self._evolution):
...
-                evo._sql_id = cur.fetchone()[0]
+                if hasattr(evo, '_sql_id'):
+                    break
+                else:
+                    evo._sql_id = cur.fetchone()[0]
```

#10 - 19 juin 2022 17:55 - Pierre Ducroquet

Frédéric Péters a écrit :

> J'ai l'impression qu'on peut plus simplement :
>
> [...]

Moi aussi j'y ai cru... mais dans ce cas les id ne seront plus séquentiels. Je n'ai donc pas voulu casser cette supposition (qui, je crois, est testée dans un test unitaire)

#11 - 19 juin 2022 19:09 - Frédéric Péters

> Moi aussi j'y ai cru... mais dans ce cas les id ne seront plus séquentiels. Je n'ai donc pas voulu casser cette supposition (qui, je crois, est testée dans un test unitaire)

Ok, ça me semble quelque chose qui mérite d'être changé, je vois hors tests, l'ORDER BY devrait vraiment être adapté.

```
        sql_statement = (
            '''SELECT id, who, status, time, last_jump_datetime,
                                    comment, parts FROM %s_evolutions
                               WHERE formdata_id = %%(id)s
                            ORDER BY id'''
            % self._table_name
        )
```

et peut-être d'autres.

#12 - 20 juin 2022 08:32 - Pierre Ducroquet

Le champ time des objets evolution n'est précis (côté python) qu'à la seconde. Du coup en cas d'insertion de plusieurs objets créés sur un intervalle court, on ne pourra pas trier avec ce seul critère, il faudra un autre critère pour les distinguer.
Actuellement, sur la base de test, j'ai trouvé sans difficultés une table evolution où on a deux objets evolutions pour le même formdata et le même time.

#13 - 20 juin 2022 09:19 - Frédéric Péters

Ok, on ne se lance pas là-dedans du coup; je serais quand même pour revoir le patch, pas fan du for i in range(len(self._evolution) - 1, -1, -1):, plutôt pour quelque chose type,

```
            for idx, evo in enumerate(self._evolution):
                if not hasattr(evo, '_sql_id'):
                    break
            for evo in self._evolution[idx:]:
                # le code actuel
```

(qui ne doit pas être correct, c'est pour l'idée)

#14 - 20 juin 2022 12:14 - Pierre Ducroquet

- Fichier 0001-sql-only-the-last-Evolution-object-can-be-changed-65.patch ajouté

Le code me semble correct et passe les tests unitaires.

#15 - 30 juin 2022 12:20 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

> Et noter que ce sera toujours le dernier evolution qui sera modifié, jamais les précédents, et peut-être que juste se baser là-dessus serait totalement satisfaisant.

~~Je ne vois pas dans ce dernier patch ce qui fait que la dernière évolution est toujours sauvegardée, on est certain qu'elle n'a pas de _sql_id ?~~

Ok on repart de idx idx qui est au maximum len(evo)-1 donc c'est bon. Validé.

#16 - 30 juin 2022 13:48 - Pierre Ducroquet

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

Merci ! mergé

```
commit aebf5a59b50650943b722f86e910f7a75fd824f2 (HEAD -> main, origin/main, origin/HEAD, wip/65744-evolution-n
eedless-update)
Author: Pierre Ducroquet <pducroquet@entrouvert.com>
Date:   Mon May 30 08:52:50 2022 +0200

    sql: only the last Evolution object can be changed (#65744)
```

#17 - 30 juin 2022 22:14 - Transition automatique

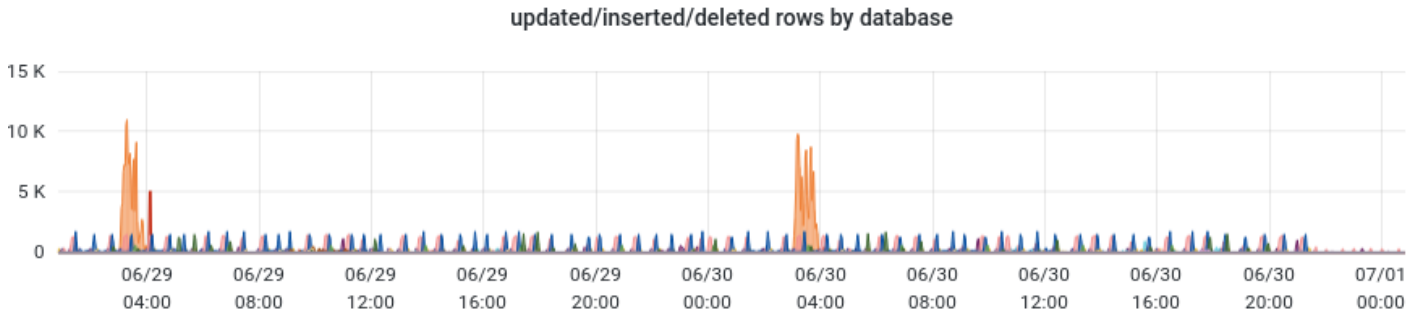
- Statut changé de Résolu (à déployer) à Solution déployée

#18 - 01 juillet 2022 00:53 - Pierre Ducroquet

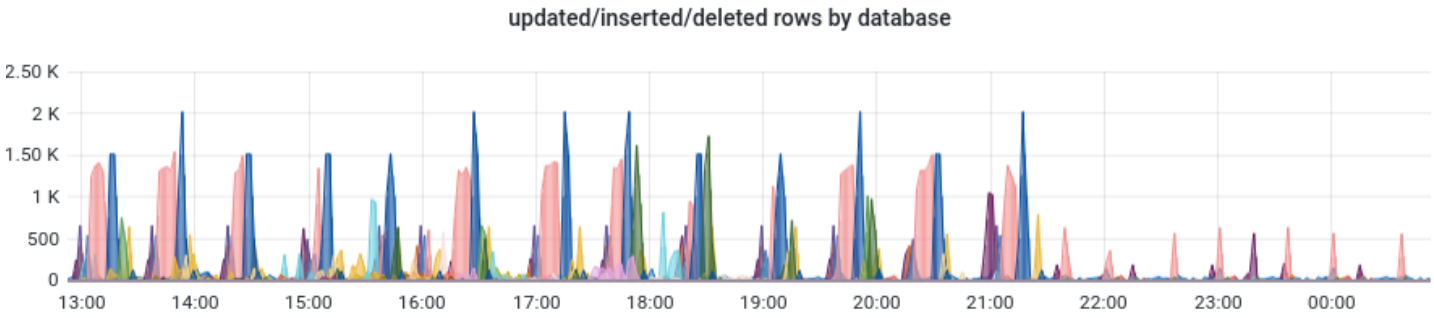
- Fichier Screenshot_20220701_005142.png ajouté

- Fichier Screenshot_20220701_005240.png ajouté

On voit nettement le déploiement...



Si on zoome sur les 12 dernières heures...



J'ai hâte de voir le résultat sur les WAL et le bloat...

#19 - 06 juillet 2022 15:36 - Pierre Ducroquet

Sur une semaine, le volume de WAL a été divisé par deux en recette.

#20 - 11 septembre 2022 04:42 - Transition automatique

Automatic expiration

Fichiers

0001-sql-don-t-update-when-Evolution-object-is-unchanged-.patch	4,07 ko	30 mai 2022	Pierre Ducroquet
0002-Fix-duplicate-__init__.patch	824 octets	30 mai 2022	Pierre Ducroquet
0001-sql-only-the-last-Evolution-object-can-be-changed-65.patch	1,2 ko	07 juin 2022	Pierre Ducroquet
0001-sql-only-the-last-Evolution-object-can-be-changed-65.patch	1,01 ko	20 juin 2022	Pierre Ducroquet
Screenshot_20220701_005142.png	28 ko	30 juin 2022	Pierre Ducroquet
Screenshot_20220701_005240.png	52,1 ko	30 juin 2022	Pierre Ducroquet