w.c.s. - Development #57017

Locker quelque chose pendant les modifications de schéma

16 septembre 2021 17:08 - Benjamin Dauvergne

Statut:	Rejeté	**Début:**	16 septembre 2021
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Au début de chaque point d'entré[1] dans les méthodes de migration (migrate, do_formdef_tables appelé par FormDef.data_class()) il faudrait locker un table globale, pour s'assurer qu'on ne migre pas dans plusieurs processus à la fois, peut-être pg_catalog.pg_class.

[1]

```
$ git grep 'sql\.[a-z]\w*(' wcs  | grep -v get_conn | grep -v 'totals\|count\|get_field_id\|get_ca
rddef_new\|wipe\|new_id\|cleanu'
wcs/admin/settings.py:            sql.do_user_table()
wcs/carddef.py:            table_name = sql.get_formdef_table_name(self)
wcs/carddef.py:            actions = sql.do_formdef_tables(self)
wcs/ctl/management/commands/convert_to_sql.py:            sql.do_formdef_tables(formdef, rebuild_v
iews=True, rebuild_global_views=True)
wcs/formdef.py:            self.table_name = sql.get_formdef_table_name(self)
wcs/formdef.py:            sql.do_global_views(conn, cur)
wcs/formdef.py:            table_name = sql.get_formdef_table_name(self)
wcs/formdef.py:            actions = sql.do_formdef_tables(self)
wcs/formdef.py:            sql.do_formdef_tables(self, rebuild_views=True, rebuild_global_views=Tr
ue)
wcs/formdef.py:            sql.do_formdef_tables(self, rebuild_views=True, rebuild_global_views=Tr
ue)
wcs/publisher.py:        sql.do_session_table()
wcs/publisher.py:        sql.do_user_table()
wcs/publisher.py:        sql.do_role_table()
wcs/publisher.py:        sql.do_tracking_code_table()
wcs/publisher.py:        sql.do_custom_views_table()
wcs/publisher.py:        sql.do_snapshots_table()
wcs/publisher.py:        sql.do_loggederrors_table()
wcs/publisher.py:        sql.do_meta_table()
wcs/publisher.py:        sql.drop_views(None, conn, cur)
wcs/publisher.py:            sql.do_formdef_tables(_formdef)
wcs/publisher.py:        sql.migrate_global_views(conn, cur)
wcs/publisher.py:        sql.migrate()
wcs/publisher.py:        sql.reindex()
```

Demandes liées:

Lié à w.c.s. - Development #57118: faire les modifications de schéma à la sau...	**Fermé**	**20 septembre 2021**

Historique

#2 - 16 septembre 2021 19:21 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

#3 - 16 septembre 2021 19:21 - Benjamin Dauvergne

- Fichier 0001-sql-lock-wcs_meta-during-schema-updates-57017.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

#4 - 16 septembre 2021 19:59 - Benjamin Dauvergne

C'est pylint qui fait des siennes.

#5 - 16 septembre 2021 20:06 - Frédéric Péters

do_formdef_tables (particulièrement) est le plus souvent appelé sans action de modification derrière; y aurait-il possibilité de limiter l'envoi du LOCK uniquement en cas de modification ? (cela écrit sans aucune idée du coût de l'appel).

#6 - 16 septembre 2021 23:14 - Benjamin Dauvergne

- Fichier 0001-sql-lock-wcs_meta-during-schema-updates-57017.patch ajouté

Voilà, ContextVar c'est du threading.local() moderne.

#7 - 19 septembre 2021 17:48 - Frédéric Péters

Je ne capte pas comment on passe du premier patch, qui lance juste "LOCK wcs_meta" peut-être un peu trop souvent (c'est mon commentaire en tout cas) au deuxième patch, avec décorateur et un truc de profondeur et de réinitialisation de lock.

Je ne sais pas si en soit le premier patch était trop naïf et ne marchait en fait pas, ou si le deuxième a une complexité inexpliquée inutile.

~~

À tester de mon côté, déjà le comportement : le second LOCK il fait attendre, il ne lève pas d'erreur etc. ça me laisse penser que le premier patch serait fonctionnellement ok.

Sur la forme par contre, je n'étais donc pas fan, mais peut-être encoe moins fan du deuxième.

Mon approche alternative (poussée dans une branche pas testée plus que ça, pour au moins avoir l'idée), ce serait de détecter les moments de modifications de schéma au niveau du curseur psycopg2, qqch comme ça :

```
+class LockableCursor(psycopg2.extensions.cursor):
+    locked = False
+
+    def execute(self, sql, args=None):
+        if not self.locked and (
+            sql.startswith('ALTER') or sql.startswith('CREATE') or sql.startswith('DROP')
+        ):
+            # hold a lock (on wcs_meta as it will be the first table we create) during schema
+            # changes; this should avoid concurrent operations and potential deadlocks.
+            super().execute('LOCK TABLE wcs_meta')
+            self.locked = True
+        super().execute(sql, args)
+
+
 def get_connection_and_cursor(new=False):
     conn = get_connection(new=new)
     try:
-        cur = conn.cursor()
+        cur = conn.cursor(cursor_factory=LockableCursor)
```

C'est isolé, ça évite d'avoir à penser à poser des lock_schemas() ou @guard_lock et lock().

#8 - 20 septembre 2021 09:15 - Benjamin Dauvergne

- Statut changé de Solution proposée à Information nécessaire

Frédéric Péters a écrit :

> Je ne capte pas comment on passe du premier patch, qui lance juste "LOCK wcs_meta" peut-être un peu trop souvent (c'est mon commentaire en tout cas) au deuxième patch, avec décorateur et un truc de profondeur et de réinitialisation de lock.

Oui j'aurai du expliquer; si on lock en début de transaction, i.e. avant le code de lecture qui décide ou pas de modifier le schéma, on a pas de souci mais effectivement on perd en performance parce qu'on lock tout le temps et donc on sérialise tous chemins du code qui ont besoin d'appeler do_formdef_tables; mais donc si on décide de ne locker qu'au moment des modifications de schéma on se trouve avec une race condition entre les transactions, car une fois que le lock se libère on est peut-être plus dans une situation qui demande la modification du schéma. Exemple : j'ai deux do_formdef_tables qui se lancent en paralèlle pour ajouter la colonne f2, le premier obtient le lock immédiatement et ajoute la colonne le deuxième détecte qu'il faut ajouter la colonne arrive sur le lock_schema() attend que le premier finisse puis tente d'ajouter la colonnne une deuxième fois; ça lève une erreur.

Quand on lock au dernier moment on doit redémarrer la transaction complète au début (avec les vérifications dans information_schema du besoin de modifier le schéma) pour ne pas se retrouver dans cette situation.

> Je ne sais pas si en soit le premier patch était trop naïf et ne marchait en fait pas, ou si le deuxième a une complexité inexpliquée inutile.

Si les deux marchent mais ça dépend si on cherche la simplicité ou la performance (pseudo-performance parce qu'on a pas mesuré l'impact réel de

locker en début de transaction).

~~

> À tester de mon côté, déjà le comportement : le second LOCK il fait attendre, il ne lève pas d'erreur etc. ça me laisse penser que le premier patch serait fonctionnellement ok.

Oui je pense qu'il l'était aussi, j'ai juste voulu répondre à ton inquiétude sur les performances de do_formdef_tables.

> Sur la forme par contre, je n'étais donc pas fan, mais peut-être encore moins fan du deuxième.
>
> Mon approche alternative (poussée dans une branche pas testée plus que ça, pour au moins avoir l'idée), ce serait de détecter les moments de modifications de schéma au niveau du curseur psycopg2, qqch comme ça :
>
> [...]
>
> C'est isolé, ça évite d'avoir à penser à poser des lock_schemas() ou @guard_lock et lock().

Oui mais ça a le problème dont je parle plus haut, ça ne résout pas les soucis de concurrence entre workers, mais par contre je peux combiner nos deux patchs pour éviter de saupoudrer le code de lock() un peu partout, voir garder l'approche lock en début de transaction pour la plupart des fonctions et n'utiliser guard_lock que pour do_formdef_tables et do_global_views qui semblent les deux seuls méthodes de modification des schémas qui sont appelées un peu n'importe quand.

#9 - 20 septembre 2021 17:56 - Frédéric Péters

> que pour do_formdef_tables et do_global_views qui semblent les deux seuls méthodes de modification des schémas qui sont appelées un peu n'importe quand.

Pour do_formdef_tables je vois assez, je vais faire un ticket dédié, je serais plutôt pour commencer par réduire ça. (et plutôt automatiquement ça devrait libérer ce besoin de locker, il me semble).

#10 - 20 septembre 2021 18:01 - Frédéric Péters

- Lié à Development #57118: faire les modifications de schéma à la sauvegarde de formdef/etc. plutôt qu'au moment du data_class() ajouté

#11 - 21 septembre 2021 07:41 - Frédéric Péters

> Pour do_formdef_tables je vois assez, je vais faire un ticket dédié, je serais plutôt pour commencer par réduire ça. (et plutôt automatiquement ça devrait libérer ce besoin de locker, il me semble).

#57118 et je pense qu'avec ça, si on veut locker, ça peut juste être un cur.execute('LOCK wcs_meta') exécuté en haut de do_formdef_tables.

#12 - 28 septembre 2021 09:36 - Benjamin Dauvergne

- Fichier 0001-sql-lock-wcs_meta-table-during-schema-update-57017.patch ajouté

- Statut changé de Information nécessaire à Solution proposée

Voilà, lock posé uniquement sur migrate (pour éviter que ça se marche sur les pieds avec do_formdef_table sait-on jamais) et do_formdef_tables.

#13 - 28 janvier 2022 15:30 - Benjamin Dauvergne

Rebasé à l'instant.

#14 - 09 novembre 2022 09:32 - Benjamin Dauvergne

- Statut changé de Solution proposée à Rejeté

Ça ne m'a plus l'air trop fréquent.

Fichiers

0001-sql-lock-wcs_meta-during-schema-updates-57017.patch	3,35 ko 16 septembre 2021	Benjamin Dauvergne
0001-sql-lock-wcs_meta-during-schema-updates-57017.patch	18,4 ko 16 septembre 2021	Benjamin Dauvergne
0001-sql-lock-wcs_meta-table-during-schema-update-57017.patch	1,61 ko 28 septembre 2021	Benjamin Dauvergne