GraceTHD-Check - Manuel développeur

DRAFT !!!

Révision : 01/12/2016 - sby

Préambule

Notes

- GraceTHD-Check est un projet GraceTHD-Community.
  https://github.com/GraceTHD-community
- Cette documentation est rédigée sur le wiki du Groupe Experts Contrôle sur le Redmine GraceTHD.
  https://redmine.gracethd.org/redmine/projects/gracethd-check/wiki
- Rédacteurs : Stephane Byache (Aleno).
- Licence : CC-by-sa v2.0

Utilité

GraceTHD-Check permet d'augmenter une base de données GraceTHD-MCD avec un schéma gracethdcheck qui comporte des tables et des vues offrant des relevés unitaires d'anomalies (une vue par point de contrôle), des rapports d'anomalies (vues synthétisant les anomalies rencontrées dans toutes les tables pour un statut).

Structure

Même arborescence que GraceTHD-MCD, et donc Gracelite.
La racine contient les scripts exécutables. Les scripts concernés par ce manuel sont nommés gracethdcheck_pg_*.bat.
sql_postgis contient les scripts SQL qui sont appelés par les scripts exécutables à la racine.

Procédure développeur

Développement de points de contrôles « conformes » GraceTHD-Check.

- Nommage du point de contrôle « spécifique ». Ex : ad_code_9_s00123
- Soumettre des points de contrôle sous la forme d'un t_ct_cat.csv.
- Un point de contrôle validé intègre GraceTHD. Ex : ad_code_1_s09876

Développer un point de contrôle pour une utilisation interne, et en vue de le faire référencer par le projet GraceTHD-Check.

- Vous pouvez référencer la définition d'un point de contrôle dans la table t_ct_cat. Dans ce cas utiliser un code 9 au lieu du 1. Ex : ad_code_9_s00123.
- Cette seule définition peut déjà être soumise au projet GraceTHD-Check via une demande sur le Redmine de GraceTHD. Idéalement il peut s'agir d'un extrait de t_ct_cat en csv voire en SQL. Mais ça peut également être une simple demande écrite avec une définition du point de contrôle.
- Si elle est validée, un code définitif sera attribué. Ex : ad_code_1_s00123. C'est un moyen de contribuer au projet et quelque part de réserver au plus tôt le code définitif.
- Vous pouvez également développer ce point contrôle, pour Postgis et / ou Spatialite. Dans ce cas l'intégrer dans la table t_ct_code_pgs (Postgis) et / ou t_ct_code_spl (Spatialite).
- Vous pouvez soumettre ce code via une demande sur le Redmine GraceTHD. Ca peut prendre la forme d'un ou des SQL ou CSV conformes à t_ct_code_pgs et / ou t_ct_code_spl. Mais ça peut être également la fourniture de la seule requête SQL de contrôle sans les propositions d'attributs de la table t_ct_code*.

Note : La structure des points de contrôle doit toujours être identique.

- *La première ligne doit être le code du point de contrôle entre /* */*
- *La partie SELECT doit être absolument identique pour chaque requête de contrôle.*
- *Utiliser un outil qui permet de gérer correctement la saisie multi-lignes en base et qui respecte les indentations du code. pgAdmin gère cela correctement pour l'édition.*
- *Pour Postgis il est possible d'utiliser une syntaxe PL/pgSQL, toutefois tant que le SQL suffit c'est préférable car il est plus simple de porter du SQL Postgis sur Spatialite, et inversement, que du PL/pgSQL. A réserver pour les cas nécessaires.*

Note : t_ct_conf

- _t_ct_conf permet de stocker des variables réutilisables dans les requêtes des points de contrôle. Les utilisateurs peuvent ainsi adapter le comportement de certains points de contrôle à leur contexte.
- *t_ct_conf_user permet de stocker les variables modifiées pour le contexte. Toutefois ce n'est qu'une forme de sauvegarde, c'est toujours t_ct_conf qui est utilisée. t_ct_conf doit être mise à jour (sans écraser les autres variables) avec les variables de t_ct_conf_user. Exécuter : gracethdcheck_pg_maj_conf.bat*

Exemples de syntaxe de points de contrôle

Exemple Postgis

Attribut ad_code ayant une valeur NULL

```
/*ad_code_1_r00001*/
   SELECT
      'ad_code_1_r00001'::text AS ct_code,
      'r'::text AS ct_type,
      'Valeur NULL pour un attribut dont le remplissage est obligatoire. '::text AS ct_def,
      '1'::text AS ct_sensib,
      NULL::text  AS ct_detail,
      --datetime('now', 'localtime') AS ct_date, --sqlite
      NOW() AS ct_date,  --postgresql
      (SELECT valeur FROM t_ct_conf WHERE nom='ct_1_liv') AS ct_liv,
      *
   FROM t_adresse
   WHERE ad_code IS NULL
   ;
```

Exemple Spatialite

```
/*ad_code_1_r00001*/
   SELECT
      'ad_code_1_r00001' AS ct_code,
      'r' AS ct_type,
      'Valeur NULL pour un attribut dont le remplissage est obligatoire. ' AS ct_def,
      '1' AS ct_sensib,
      NULL AS ct_detail,
      datetime('now', 'localtime') AS ct_date, --sqlite
      --NOW() AS ct_date,  --postgresql
      (SELECT valeur FROM t_ct_conf WHERE nom='ct_1_liv') AS ct_liv,
      *
   FROM t_adresse
   WHERE ad_code IS NULL
   ;
```

Structure des tables de codes (t_ct_code_xxx) :

- ct_pgs_code : le code unique du point de contrôle dans la table. Devrait être égal à ct_pgs_cat_code, sauf si c'est un point de contrôle en cours de test et non référencé dans le catalogue.
- ct_pgs_cat_code : le code du point de contrôle dans le catalogue de points de contrôle (t_ct_cat).
- ct_pgs_statut : le statut de ce point de contrôle (l_ct_statut).
- ct_pgs_version : la version de l'implémentation du point de contrôle.
- ct_pgs_source : la source du point de contrôle, idéalement une adresse mail.
- ct_pgs_date : la date de la dernière révision du point de contrôle.
- ct_pgs_commentaire : commentaires.
- ct_pgs_script : le script du point de contrôle. Pour Postgis et Spatialite, le début du script doit être /*[le code point de contrôle]*/ (cf. exemple).