Reporting

Direction des Systèmes d’Information
Pôle Géomatique

9 juillet 2007

Introduction

L'objectif de ce document est de décrire la chaine de traitement qui a été utilisé pour générer le catalogue topographique.

Le catalogue topographique Le produit final est un fichier pdf. En ce qui concerne le contenu, il est extrait d'une Base de Données Postgresql. Des images sont aussi insérées dans ce catalogue.

Un schéma de la procédure Le fichier pdf est généré à l'aide de l'outil LaTeX.

Les images insérées dans ce document ont été converties à l'aide de plusieurs utilitaires dont ImageMagick et swftools. Les extraits de cartes ont quant à eux, été réalisé grace à mapyrus.

Les données attributaires ont pu être incorporées dans le document latex à l'aide de l'outil latexbd. Ces données sont extraites d'une base de données Postgresql. Initialement au format Excel, elles ont été insérées dans la base Postgresql à l'aide des utilitaires cvsToSql et psql.

Les outils Pour réaliser ce catalogue, nous avons donc utiliser :

1. Postgresql
2. LaTeX
3. ainsi que divers autres outils complémentaires
   (a) recode (pour la gestion des caractères accentuées)
   (b) latexbd (utilitaire permettant d'interfacer LaTeXet Postgresql)
   (c) csv2sql (pour transferer des données excel dans la base postgresql)

Chapitre 1

Pour aller vite...

1.1 Description de la procédure "‘données attributaires"’

1. Mise à jour des fichiers Excel
2. Transfert des données Excel dans la base Postgresql
   (a) Utilisation de csvToSql
   (b) Execution d’un script de transfert
3. Compilation du document .tex

1.2 Mise à jour des fichiers Excel

Les fichiers à modifier sont sur le réseau, dans le répertoire

```
F:\INFO\geomatique\dossiers_en_cours\donnees_topo
```

Ces fichiers sont :
- Theme.xls
- Domaine.xls
- Classe.xls
- Objet.xls
- Representation.xls
- Representation_covadis.xls

1.3 Transfert des données Excel dans la base Postgresql

1.3.1 Utilisation de csvToSql

Cet utilitaire se lance à partir de la commande suivante :

```
C:\libre\csvtosql_jdk50\bin\rungui.bat
```

L’utilisation de l’outil est très simple, on charge un fichier .xml qui décrit les différents paramètres de la conversion, puis on lance l’exécution de ce transfert selon ces paramètres.

Pour de plus amples informations sur csvToSql, il faut aller sur le site

```
http://csvtosql.sourceforge.net/
```

1.3.2 Execution d'un script de transfert

Sous linux, dans le répertoire

`/catalogue_topo/bin`

un script shell nommé

`fichier_commande`

est à exécuter. Ce script effectue les différentes manipulations spécifiques du transfert de fichier texte vers la base Postgresql

- ré-encodage des fichiers,
- execution de commandes SQL.

1.4 Compilation du document .tex

Ici aussi, on va utiliser un script. Ce script lance la commande latexbd qui s'occupe de générer le fichier .pdf en sortie. Ce script est :

`/catalogue_topo/bin/reporting_test.sh`

1.5 Commentaires : les limites de la procédure

Voici les différents problèmes rencontrés :

- utilisation des caractères accentués : problème résolu
- utilisation du caractère _ : problème résolu

Chapitre 2

Installation de la chaîne de traitement

2.1 Introduction

La chaine de traitement va nous permettre de faire du reporting de Postgresql vers un fichier .pdf. Nous pouvons faire un parallèle très grossier entre LaTeXet les états de MS Access. Nous allons

- utiliser différents maillons
  - Postgresql
  - recode
  - latexbd
- gérer les caractères accentués

On remonte la chaîne à partir de la fin

2.2 Les maillons de la chaîne

2.2.1 Le document catalogue-topo-but.tex

Ce document est généré par latxbd à partir du document catalogue-topo.tex

Ce fichier -but.tex est un pur LaTeX. Il peut comporter des caractères accentués, mais il doit avoir un caractère d'échappement avant le caractère souligné. Sous Linux, l'en-tête du document contient :

- `\usepackage[utf8]{inputenc}`
- `\usepackage[T1]{fontenc}`
- `\usepackage[french]{babel}`

2.2.2 Le document catalogue-topo.tex

Ce document n'est pas un document LaTeX. C'est un document source pour latexbd. Latexbd est un utilitaire récupéré sur ce site :

`http://perso.orange.fr/Etienne.Marache/latex/lbd/latexbd.html`

Il permet de faire la liaison entre LaTeXet une base de données.

Installation des paquets :

- texlive
- texlive-lang-french
- texmaker
- xpdf

- gv
- python-sqlalchemy
- python-psycopg2

On télécharge puis on recopie les fichiers de latexbd.

- Dans le répertoire /home/fred/catalogue_topo/reporting/tutorial, on place
  - les fichiers .tex
  - le fichier exemples-lbd.sql
- Dans le répertoire /usr/local/bin, on place les fichiers
  - latexbd
  - latexbd.py
  - lbda.py

Dans le répertoire /usr/local/bin, on change les droits sur les fichiers :

```
chmod 755 latexbd.py
chmod 755 latexbd
chmod 755 lbda.py
```

Attention, le document catalogue-topo.tex, comme le document catalogue-topo-but.tex ne doivent pas, contenir le caractère souligné.

En ce qui concerne le fichier catalogue-topo.tex, si les noms de colonnes de la base contiennent le caractère souligné, alors on utlisera les alias.

D'autre part, nous allons faire une petite modification dans le script shell /usr/local/bin/latexbd afin que celui-ci produise des documents sans _ mais avec
_. Modification du script shell /usr/local/bin/latexbd

```
# après $lb $fichier
awk '{gsub("\_","\\\_",$0); print}' $fichier-but.tex > $fichier-temp.tex
# cette ligne est equivalente à
sed "s/\_/\\\_/g" $fichier-but.tex > $fichier-temp.tex
# puis ensuite on recopie le -temp en -but
cp $fichier-temp.tex $fichier-but.tex
```

2.2.3 La Base Postgresql doit être en UTF8

Latexbd va aller piocher les informations dans une base de données appelée topographie. Cette base de données doit être en UTF8.

Pour créer une base de données en UTF8, on peut utiliser les commandes suivantes

```
psql template1 -c 'DROP DATABASE "topographie";'
psql template1 -c 'CREATE DATABASE "topographie" WITH ENCODING='\''UTF8\'';'
```

2.2.4 Les fichiers SQL

Les fichiers SQL sont générés par l'utilitaire csvToSql (sous windows). Ils sont ensuite passés sous Linux. Le script fichier_commande s'occupe de re-encoder les fichiers SQL avant de lancer leur execution sous Postgresql.

```
recode l1..utf8 < attribut_valeur.sql > attribut_valeur2.sql
recode l1..dump < attribut_valeur.sql | pager
recode l1..count < attribut_valeur.sql
```

Chapitre 3

Pour aller plus loin...

3.1 Les commandes latexbd et lbdpython

3.2 Les problèmes d'encodage

3.3 Les commandes shell

3.4 L'utilisation de Postgresql

3.4.1 Pour supprimer une base de données

```
psql template1 -c 'DROP DATABASE "exemples-lbd";'
```

3.4.2 Création d'une base Postgresql

```
createdb topographie
psql template1 -c 'CREATE DATABASE "topographie" WITH ENCODING='\''UTF8\'';'
psql template1 -c 'CREATE DATABASE "exemples-lbd" WITH ENCODING='\''UTF8\'';'
```

3.4.3 Ajout du langage plpgsql à cette base

```
createlang plpgsql topologie
```

3.4.4 Ajout de fonctionnalités postgis à la base topologie

```
psql topologie
     -f /usr/share/postgresql-8.1-postgis/lwpostgis.sql
psql topologie
     -f /usr/share/postgresql-8.1-postgis/spatial-ref-sys.sql
psql topologie
     -f /usr/share/postgresql-8.1-postgis/lwpostgis-upgrade.sql
```

3.4.5 Pour consulter et interroger le serveur Postgresql

A l'aide de pgsql

```
\l liste les bases
\dt liste les tables de la base courante
```

```
\du liste les roles
```

à l’aide de henplus

```
connect jdbc:postgresql://localhost/topologie
```

Conclusion

Annexes

csvToSql

A titre d'information, voici le contenu d'un fichier descripor.xml.

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
  <!-- CONFIGURATOR DATA -->
  <descriptor version="3"/>

  <!-- DEFINE HERE TABLE STRUCTURE -->
  <structure tablename="theme">
    <field name="theme_id"     type="VARCHAR">
     <param name="notnullvalue"  value="'mystring'"/>
    </field>
    <field name="theme_libelle"  type="VARCHAR"/>
  </structure>

  <!-- WHAT GRAMMAR USE -->
  <!-- ici, on a le choix entre -->
  <!-- mysql -->
  <!-- oracle -->
  <!-- standard -->
  <grammar class="net.sf.csv2sql.grammars.standard.GrammarFactory"/>

  <!-- TEMPORARY STORAGE -->
  <storage class="net.sf.csv2sql.storage.Memory"/>

  <!-- RENDERER CONFIGURATION -->
  <!-- ici, on a le choix entre -->
  <!-- SSqlInsertRenderer -->
  <!-- SqlInsertRendererFromExcel -->
  <render class="net.sf.csv2sql.renders.SqlInsertRendererFromExcel">
    <param name="inputfile"     value="theme.xls"/>
    <!--optional-->
    <param name="trimdata"           value="true"/>
    <param name="suppressheader"     value="true"/>
    <param name="removedoublequotes" value="false"/>
    <!-- decomment for add id autogeneated field -->
    <!--<param name="idGenerator"   value="net.sf.csv2sql.idgenerators.Incremental"/>-->
    <!--<param name="startrow" value="2"/>-->
```

```
    <!--<param name="stoprow" value="4"/>-->
    <!--<param name="presql" value="BEGIN TRANSACTION;"/>-->
    <param name="presql" value="
SELECT drop_table_if_exists ('theme',false);
CREATE TABLE theme
(
  theme_id character varying(256) NOT NULL,
  theme_libelle character varying(256) NOT NULL
)
WITH OIDS;
"/>
    <!--<param name="postsql" value="COMMIT;"/>-->
    <param name="postsql" value="
VACUUM ANALYSE theme;
"/>
  </render>

  <!-- WRITER CONFIGURATION -->
  <!-- ici, on a le choix entre -->
  <!-- SqlFileWriter -->
  <!-- StandardOutputWriter -->
  <!-- JDBCBatchWriter -->
  <!-- JDBCWriter -->
  <output>

    <writerAppender active="true"
        class="net.sf.csv2sql.writers.SqlFileWriter">
        <param name="filename" value="F:\INFO\geomatique\dossiers_en_cours\donnees_topo\the
    </writerAppender>

    <writerAppender active="true"
        class="net.sf.csv2sql.writers.StandardOutputWriter">
    </writerAppender>

    <!-- change parameters and activate it -->
    <writerAppender active="false"
        class="net.sf.csv2sql.writers.JdbcWriter">
        <param name="driver"    value="org.postgresql.Driver"/>
        <param name="url"       value="jdbc:postgresql://localhost/test"/>
        <param name="username"  value="fred"/>

        <!--optional-->
        <param name="password"  value="fred"/>
        <!--<param name="commit"    value="false"/>-->
        <param name="jdbcjar"   value="C:/libre/csvtosql_jdk50/lib/postgresql-8.2-505.jdbc4
    </writerAppender>

    <!-- change parameters and activate it -->
    <writerAppender active="false"
        class="net.sf.csv2sql.writers.JdbcBatchWriter">
```

```
        <param name="driver"        value="org.postgresql.Driver"/>
        <param name="url"           value="jdbc:postgresql://localhost/test"/>
        <param name="username"      value="fred"/>
        <param name="batchcount"    value="10"/>

        <!--optional-->
        <param name="password"      value="fred"/>
        <param name="commitbatchcount"  value="0"/>
        <param name="jdbcjar"       value="/root/myJdbcDriver.jar"/>
    </writerAppender>

  </output>

</root>
```

Table des matières