MAKINA CORPUS

Jobs Talend

Table des Matières

1 Contenu du document

Ce document contient une description précise des Jobs Talend Gr@ce. Il a été écrit à partir du document original fourni lors du premier marché. Il convient d'avoir des connaissances Talend avancée pour comprendre le fonctionnement de ceux-ci.

2 Jobs Talend

[2.1] Installation

[2.1.1] Import du projet

Le fichier projet est compacté en un seul fichier ZIP. Il contient l'ensemble des jobs talend et des paramètres nécessaires à l'exécution des tâches d'intégration des données (fichiers géographiques xxx_GEO associés au projet Gr@ce).

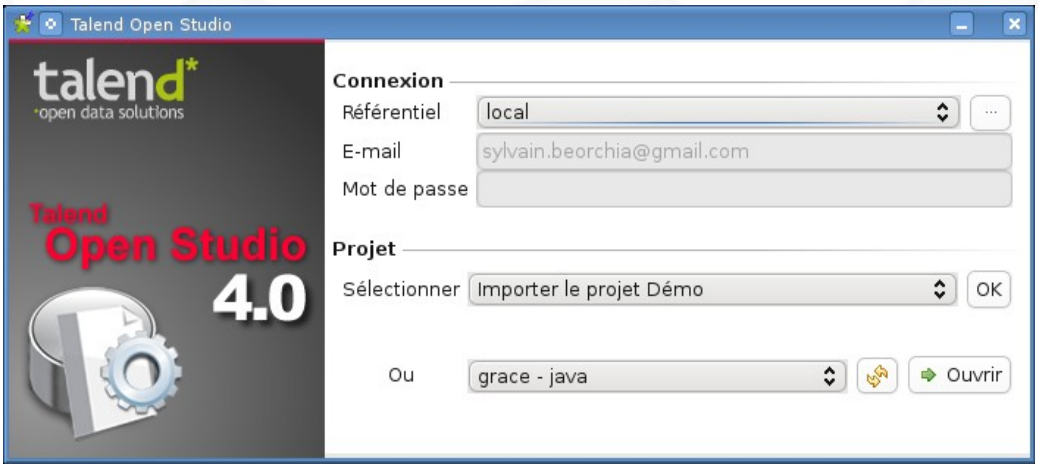
Sous windows, le lancement de Talend se fait via un raccourci sur le bureau.
Sous linux, la commande est la suivante :

./TalendOpenStudio-linux-gtk-x86.sh.

La version utilisée est la version Talend SDI 4.0 (cette version est disponible à l'adresse suivante :
http://sourceforge.net/projects/sdispatialetl/files/sdispatialetl/TOS.spatial.4.0.1/.

Au premier lancement de Talend, il est nécessaire de créer une connexion locale. Cette opération n'est plus nécessaire par la suite.

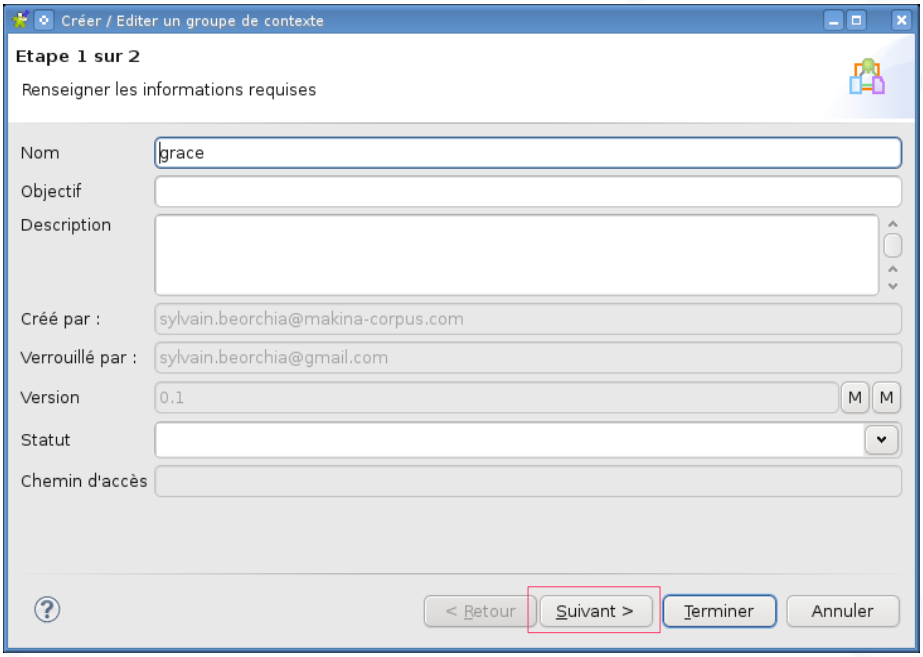
L'import du projet se fait sur le premier écran :



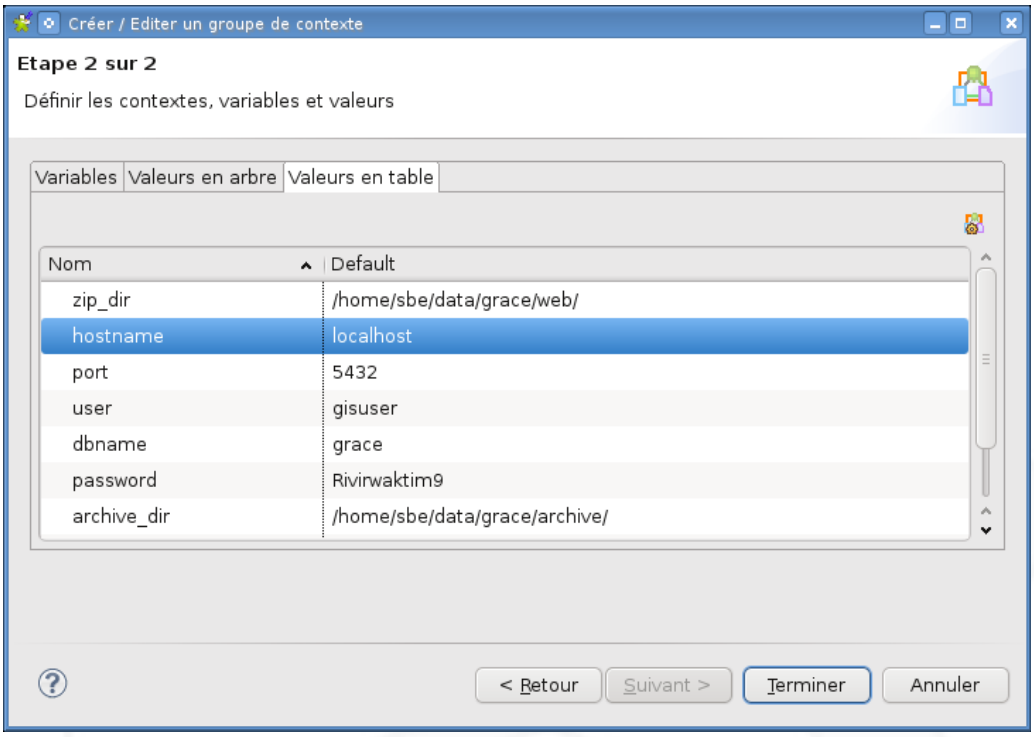
Il faut choisir l'option « Importer le ou les projets existants en local » et valider. Puis il faut alors choisir le fichier ZIP du projet. Lorsque cette opération aura été validée une fois, il ne sera plus nécessaire de la refaire, le projet sera disponible dans la liste des projets existants. L'ouverture du projet est plus ou moins longue selon les environnements de travail utilisés.

[2.1.2] Configuration du projet

Au premier lancement du projet, il est nécessaire de configurer les variables de contexte en fonction de l'environnement de travail utilisé. Le contexte « grace 0.1 » est disponible dans la marge à gauche (onglet référentiel). Double cliquer dessus pour l'ouvrir. Le premier écran n'est pas utilisé :



Dans le suivant, il faut cliquer sur « Valeur en table » et modifier les valeurs :

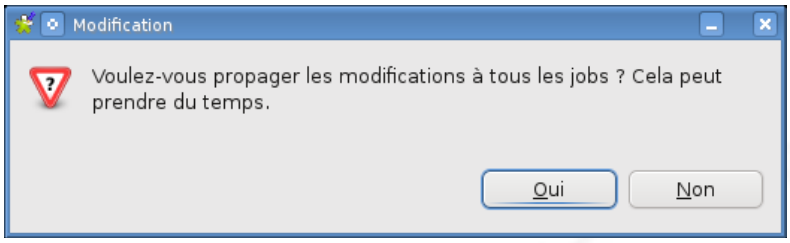


Le tableau ci-dessous présente les différentes variables :

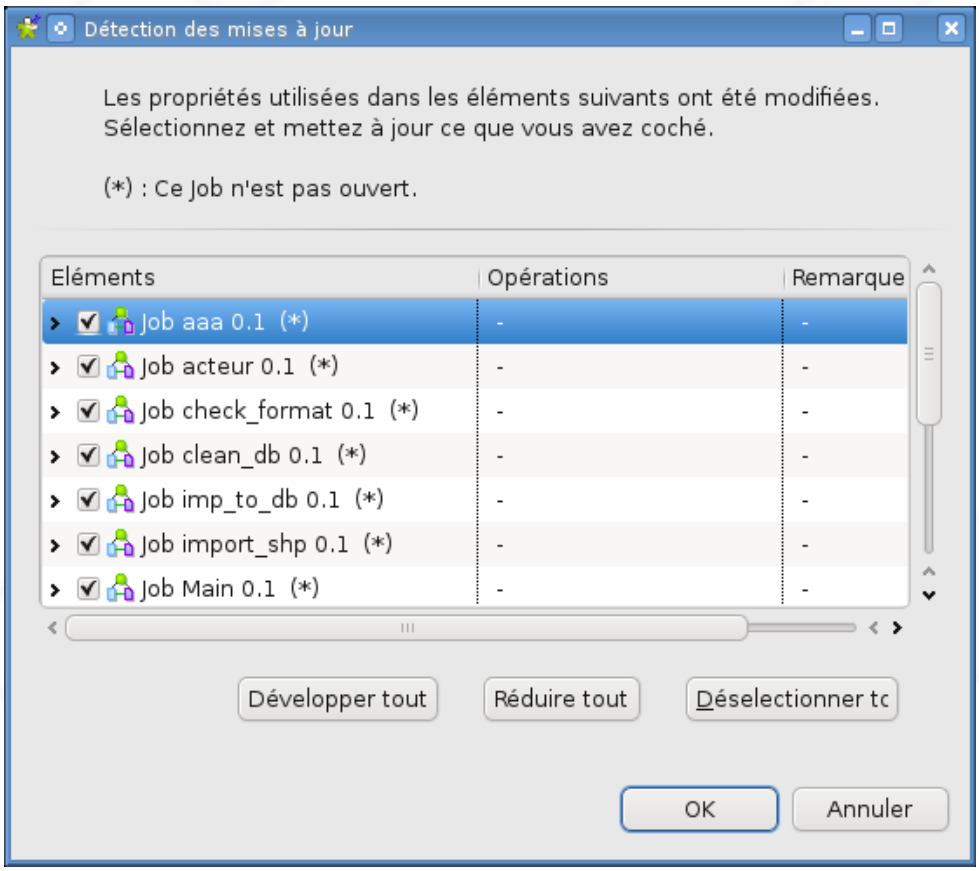
Nom de la variable	Description	Exemple
zip_dir	Répertoire racine où sont contenus les fichiers ZIP des données entrantes	C:\data\zip\
hostname	Nom du serveur de la base de données	Localhost
port	Port de la base de données	5432
user	Utilisateur de la base de données	gisuser
dbname	Nom de la base de données	grace
password	Mot de passe de l'utilisateur de la base de données	xxxxx
archive_dir	Répertoire dans lequel seront copiés les fichiers ZIP ayant été traités	C:\data\archive\
currentPartner	Variable contenant le nom du ZIP de traitement.	NULL
source_maj	Nom d'intégrateur des données	CRA
departement	Clé du département	2472
zip_name	Nom du fichier zip	Grace.zip

zone	Zones autorisées pour l'utilisateur	40003,4004

Lorsque les variables sont correctement paramétrées, il faut valider la fenêtre en cliquant sur Terminer. Un message propose de propager les modifications à tous les job.



Cette étape est nécessaire pour la bonne personnalisation du job.



[2.2] Jobs

[2.2.1] Main

- Connexion à la base

Ce composant teste la connexion à la base. Si la connexion échoue, le traitement s'arrête.

- Pré-traitement, pour récupérer le nom du zip ainsi que le nom du partenaire courant.

- uploadedZips / process / Extraction du zip

Ce groupe de composants permet de lire le répertoire ZIP contenant les fichiers entrants. Pour chacun des fichiers ZIP, il en extrait les fichiers XXX_GEO et lance l'intégralité du traitement (c'est-à-dire tous sous-jobs) avant de passer au suivant. Les fichiers ZIP entrants doivent contenir les fichiers shape suivants : ARTERE_GEO, NOEUD_GEO et TRANCHEE_GEO. Ceux-ci doivent être accompagnés des fichiers annexes au SHP (dbf, shx, prj) et respecter le format attendu par le JOB.

- import_shp (cf chapitre correspondant)

Ce job importe les Shapefile dans des tables temporaires table_shp_imp.

- tmp_to_imp (cf chapitre correspondant)

Ce job importe les tables temporaires dans des tables intermédiaires table_imp.

- check_format (cf chapitre correspondant)

Ce job vérifie le format des tables.

- acteur (cf chapitre correspondant)

Ce job recherche et ajoute dans la base, s'ils n'ont pas déjà été renseignés, de nouveaux acteurs trouvés dans les données.

- imp_to_db (cf chapitre correspondant)

Ce job importe les données temporaires dans les tables définitives.

- Création des linéaires

Ce groupe de composants réparti une partie des informations depuis la table ARTERE vers les tables FOURREAU, CABLE ET SOUS-TUBAGE. En entrée, un composant **`tPostgresqlInput`** sélectionne des données (connexion à la base, lecture de la table ARTERE selon le SQL précisé dans l'onglet composant). Le composant **`tFlowToIterate`** permet de passer en revue tous les enregistrements de la sélection. Trois composants **`tLoop`** permettent de boucler sur « nbr_cable_artere », « nbr_four_artere » et « nbr_tube_artere » pour créer autant d'objet que précisé dans les données d'entrées. Les composants **`tMap`** et **`tPostgresqlOutput`** permettent ensuite de faire les correspondances de champs.
*Remarques : les composant **tMap** utilisent des fonctions particulières, définies dans le chapitre Code Custom.*

- Traitements A

Ce traitement appelle la méthode « `majFourCableST` » (cf description précise dans le chapitre code custom).

- Traitements des nœuds

Ce groupe de composant réparti une partie des informations depuis la table NOEUD vers les tables CHAMBRE, LOCAL_TECHNIQUE (LT) et SITES_EMISSION (SE). La table NOEUD_OLDNEW, utilisée dans le traitement, permet de faire la correspondance entre les nouveaux identifiants générés et les identifiants créés par défaut lors de l'import des données. Le composant **tMap** dispatche les informations.

- maj_annexe

Ce traitement lance les méthodes « `majAnnexesArtere` » et « `majAnnexesNoeud` ». Ces dernières sont décrites dans le chapitre code custom .

- noeud_ep (cf chapitre correspondant)

Ce job crée les équipements passifs à partir des informations de NOEUD_IMP.

- Traitements B

Ce traitement lance la méthode « `majBranchCable` ». Celle-ci est décrite dans le chapitre code custom.

- Traitements C

Ce traitement lance la méthode « `majLiaisonEP` ». Celle-ci est décrite dans le chapitre code custom.

- maj_id_pere

Ce composant est une exécution de SQL :

```
UPDATE chambre a SET id_lt_amont = b.id_lt
FROM local_technique b, noeud n
WHERE b.id_noeud = n.id_noeud
  AND n.nom_noeud = a.id_lt_amont
  AND a.id_noeud IN (SELECT id_noeud FROM integr."+context.getProperty("currentPartner")
+"_"+context.zip_name+"_noeud_oldnew);

UPDATE local_technique a SET id_lt_pere = (
    SELECT b.id_lt FROM local_technique b WHERE b.id_noeud = (select id_noeud from
integr."+context.getProperty("currentPartner")+"_"+context.zip_name+"_noeud_oldnew WHERE nom =
a.id_lt_pere)
    )
    WHERE a.id_noeud IN (SELECT id_noeud FROM integr."+context.getProperty("currentPartner")
+"_"+context.zip_name+"_noeud_oldnew);
```

Il renseigne les champs id_lt_amont de la table CHAMBRE et id_lt_pere de la table LT.

- maj_diam_int

Ce composant est une exécution de SQL. Il renseigne les diamètres intérieurs à partir d'un gabarit.

- maj_zone_lt

Ce composant est une exécution de SQL. Il renseigne les zones courantes dans les locaux techniques.

- maj_artere_insee2

Ce composant est une exécution de SQL. Il renseigne le champ `id_com_insee2_artere` de la table artère.

- link_tranchee_artere (cf chapitre correspondant)

Ce job crée la liaison entre les tranchées et les artères.

- process_comments (cf chapitre correspondant)

Ce job traite les éventuelles informations complémentaires contenues dans les champs commentaires du fichier ARTERE_GEO.

- srid (cf chapitre correspondant)

Ce job corrige les SRID sur les tables géométriques.

- clean_db (cf chapitre correspondant)

Ce job nettoie la base des tables temporaires créées pour les besoins des traitements.

- move_to_archive

Ce traitement appelle la méthode « `moveDir` » pour déplacer le(s) ZIP traité(s) dans le répertoire archive.

A noter que tous les composant tWarn, tError servent à inscrire des informations de progression ou d'échec dans la table log et dans le fichier de log. Ces derniers sont utilisés pour informer l'utilisateur du déroulement des opérations.

[2.2.2] import_shp

Ce job prend en entrée les fichiers ARTERE_GEO, NOEUD_GEO et TRANCHEE_GEO et insère les données dans des tables temporaires correspondantes ARTERE_SHP_IMP, NOEUD_SHP_IMP et TRANCHEE_SHP_IMP.
Ce job recrée à l'identique la structure des fichiers GEO dans la base de données.

[2.2.3] tmp_to_imp

Ce job intermédiaire permet de basculer les données des tables XXX_SHP_IMP vers des tables xxx_IMP. Les données sont basculées sans traitement ni changement de structure. Pourquoi ce job alors ? Il est indispensable, car il permet de modifier la structure d'entrée des données dans le JOB sans avoir à modifier l'intégralité du JOB (c'est ce dernier qu'il faudrait modifier si jamais la structure des fichiers GEO venait à changer).

[2.2.4] check_format

Ce job vérifie la bonne conformité des schémas des tables.

[2.2.5] Acteur

Ce job lit les données entrantes (tables geo : artère, nœud et tranchée), et de crée et renseigne de nouveaux acteurs dans la tables ACTEUR s'ils n'y sont pas présents.

[2.2.6] imp_to_db

Ce job permet de basculer les données depuis les tables temporaires vers les tables réelles.
Pour les nœuds, les données sont transférées depuis la table NOEUD_IMP vers une table NOEUD_INTERM, puis vers la table NOEUD. La table NOEUD_INTERM sert dans les traitements suivant à conserver une trace des import en cours (en effet, plus tard, des jobs font des sélections sur la table des NOEUD, il est donc nécessaire de filtrer cette sélection avec un ensemble de nœud entrant). Une table NOEUD_OLDNEW est également créée. Elle permet de garder une correspondance entre les identifiants des nœuds nouvellement créés et les identifiants créés par défaut lors de l'import de données dans les tables temporaires. En résumé, cette table permet de faire la liaison entre la table des NOEUD et la table temporaire NOEUD_IMP. La table NOEUD_OLDNEW est utilisée également pour remplir les champs NOEUD_A et NOEUD_B de la table ARTERE.
Le fonctionnement est similaire pour la table ARTERE.
Pour la table tranchée, il n'y a pas besoin d'avoir de table TRANCHEE_OLDNEW.
Enfin, le dernier composant du job, lance un SQL :

```
UPDATE noeud_oldnew a SET id_old_amont = b.id_noeud FROM noeud_oldnew b WHERE
a.id_old_amont=b.id_old ;
```

Ce SQL met à jour le champ id_old_amont de la table NOEUD_OLDNEW qui servira par la suite à remplir les tables intégrant la notion de père/fils.

[2.2.7] noeud_ep

Ce job crée les ELEMENT_BRANCHEMENT_PASSIF (EP). A partir de la table NOEUD_IMP, on boucle sur le champ NB_EP, et on crée pour chaque NOEUD le nombre d'EP correspondant. La table NOEUD_OLDNEW est donc utilisé à ce niveau.

[2.2.8] link_tranchee_artere

Ce job lance le SQL ci-dessous qui met à jour le champ id_artere des tranchées :

```
UPDATE tranchee t SET id_artere = (SELECT id_artere FROM artere a WHERE
Intersects(a.geom,ST_Envelope(t.geom)) ORDER BY
ST_Length(Intersection(a.geom,ST_Envelope(t.geom))) DESC limit 1) WHERE t.id_tranchee IN (SELECT
tranchee_interm.id_tranchee FROM tranchee_interm);
```

[2.2.9] process_comments

Ce job a été inséré lors du projet FMProjet. Il s'agit de compléter certains informations sur le réseau sans casser la structure des fichiers GEO. Pour cela, des colonnes commentaires sont ajoutées dans le fichier ARTERE_GEO si nécessaires. Si ces colonnes sont présentes, alors une procédure écrite en JAVA est déclenchée. Les traitements sont complexes et nécessitaient l'utilisation d'un composant Java.
Il s'agit d'une manière générale de parser les champs commentaires, et de redistribuer les câbles / fourreaux correctement (par rapport à l'exécution du job classique qui affecte les cables / fourreaux aux artères de manière arbitraire).

[2.2.10] Srid

Ce job lance le SQL ci-dessous qui met à jour les SRID :

```
UPDATE artere SET geom=st_setsrid(geom,2154) WHERE getsrid(geom) != 2154;UPDATE noeud SET
geom=st_setsrid(geom,2154) WHERE getsrid(geom) != 2154;UPDATE cable SET
geom=st_setsrid(geom,2154) WHERE getsrid(geom) != 2154;UPDATE fourreau SET
geom=st_setsrid(geom,2154) WHERE getsrid(geom) != 2154;UPDATE sous_tubage SET
geom=st_setsrid(geom,2154) WHERE getsrid(geom) != 2154 ;
```

[2.2.11] clean_db

Ce job lance le SQL ci-dessous qui détruit les tables temporaires :

```
DROP TABLE artere_imp ;
DROP TABLE noeud_imp ;
DROP TABLE tranchee_imp ;
DROP TABLE artere_interm ;
DROP TABLE noeud_interm ;
DROP TABLE tranchee_interm ;
DROP TABLE noeud_oldnew ;
DROP TABLE noeud_shp_imp ;
DROP TABLE artere_shp_imp ;
DROP TABLE tranchee_shp_imp ;
DROP TABLE element_branchement_passif_interm ;
DROP TABLE artere_oldnew;
```

[2.2.12] main_check

Ce job est une dérivation du main. Il constitue point de départ d'une pré-exécution du job main. Son but est de vérifier la qualité des informations à importer. Il reprend le début de la structure du job main avec en plus : check_file_exist, imp_to_db_check et noeud_check_geom. Une procédure de comptage des nœuds est également présente.

Elle sert à vérifier que tous les nœuds fournis sont bien situés dans la zone géographique autorisée à l'utilisateur courant.
Lors de l'exécution de ce job, des tWarn inscrivent des informations de suivi ou d'erreur dans la table log et le fichier log. Le job est exécuté jusqu'au bout, même si une erreur est détectée, et ce afin de collecter le maximum d'informations en retour pour l'utilisateur.

[2.2.13] check_file_exist

Ce job vérifie la présence des fichiers GEO dans le zip.

[2.2.14] imp_to_db_check

Ce job est identique au job imp_to_db à la différence près que les tables créées le sont dans un schéma particulier, qui permet de multiples imports simultanés de la part du même utilisateurs.

[2.2.15] noeud_check_geom

Ce job vérifie que les nœuds fournis sont bien aux mêmes positions géographiques que les extrémités des artères liées.

[2.3] Code custom

Du code particulier a du être créé pour répondre aux besoins spécifiques des traitements, il est visible dans la marge à gauche (Code/Routines/Custom/utils 0.1).
Ce code défini une classe « `utils` » contenant plusieurs méthodes :

[2.3.1] stringToInt

Convertit une chaine de caractère en entier.

[2.3.2] StringToDouble

Convertit une chaine de caractère en réel.

[2.3.3] StringToDate

Convertit une chaine en date.

[2.3.4] GetDate

Récupère la date du jour et la formate (AAAA-MM-JJ) selon le format attendu par la base de données.

[2.3.5] GenerateIdObject

Génère les identifiants des objets selon la syntaxe suivante :

```
codePartner + "_" + «  nom de la table »  + "_" + code aléatoire
```

Dans un tMap, l'appel à cette fonction se fera, par exemple, de la manière suivante :

```
utils.generateIdObject(context.getProperty("currentPartner"),"CABLE")
utils.generateIdObject(context.getProperty("currentPartner"),"NOEUD")
```

[2.3.6] moveDir

Déplace un répertoire.

[2.3.7] DeleteDir

Détruit un répertoire.

[2.3.8] DeleteZipFiles

Détruit le(s) fichier(s) ZIP.

[2.3.9] CreateConnexion

Crée une connexion à la base.

[2.3.10] MajFourCableST

Traite les mise à jour de liaison entre les tables FOURREAU, CABLE et SOUS-TUBAGE :
Les artères en cours d'intégration sont lues.
Trois dictionnaires cableArtere, tubeArtere et fourreauArtere sont remplis, un dictionnaire contient des couples d'identifiants, et permet de retrouver les associations fourreaux/artères, câbles/artères...
Pour chaque câble, id_four est mis à jour (via du SQL) ainsi que le id_st.

Pour chaque sous-tubage, id_four est mis à jour.

[2.3.11] MajBranchCable

Traite les mise à jour de liaison entre les tables BRANCHEMENT et CABLE.
Les artères en cours d'intégration sont lues.
Deux dictionnaires tabArtereNA et tabArtereNB sont créés, ils serviront à enregistrer les correspondances Noeuds/Artères.

```
Pour chaque artère :
      On lit les câbles et pour chaque câble:
            On récupère les id_ep de la table EP correspondant aux deux nœuds
            de l'artère.
            On met à jour le cable (id_ep_a, id_ep_b) à partir de ces
            informations.
```

[2.3.12] MajAnnexesArtere

Traite les mise à jour complémentaires (diamètres, utilisateurs...) des tables annexes à ARTERE (FOURREAU, CABLE, SOUS-TUBAGE).
Les artères en cours d'intégration sont lues.
Trois dictionnaires cableArtere, stArtere et fourArtere sont remplis.

```
Pour chaque artère :
      On récupère les fourreaux associés et pour chaque fourreau :
            On extrait (depuis la table GEO):
                  TYPE_FOUR
                  ETAT_FOUR
                  DIAM_FOUR
                  GEST_FOUR
                  UTIL_FOUR
                  PRO_FOUR
                  AR_ETAT
            On met alors à jour les fourreaux avec ces données.

      On récupère les câbles associés, et pour chaque câble :
            On extrait:
                  TYPE_CABLE
                  NB_FIBRE
                  NB_FIB_UTI
                  GEST_CABLE
                  UTIL_CABLE
                  PRO_CABLE
                  DIAM_CABLE
                  ETAT_CABLE
            On met alors à jour les câbles avec ces données.
            (un calcul est fait sur nbr_fibre_dispo)
```

```
On récupère les ST associés, et pour chaque ST :
      On extrait:
            TYPE_FST
            DIAM_FST
            ETAT_FST
            GEST_FST
            UTIL_FST
            PRO_FST
            AR_ETAT
      On met alors à jour les ST avec ces données.
```

[2.3.13] MajAnnexesNoeud

Traite les mise à jour de liaison entre les tables annexes à Noeud (LT,SE, Chambre).
Les noeuds en cours d'intégration sont lus.
Trois dictionnaires ltNoeud, chNoeud et seNoeud sont remplis.

```
Pour chaque noeud :
      On lit les LT et on récupère :
            NO_CLIM
            NO_ALIM
      On met alors à jour les LT avec ces valeurs.

      On lit les SE et on récupère :
            NO_HAUT
      On met alors à jour les SE avec ces valeurs.
```

[2.3.14] majLiaisonEP

Traite les mise à jour de liaison entre les tables EP et les tables annexes à Noeud (LT,SE, Chambre).
On récupère les noeuds en cours d'intégration.

```
Pour chaque noeud :
      On lit les EP associés et pour chaque EP trouvé :
            On met à jour la liaison LT/EP
            On met à jour la liaison Chambre/EP
            On met à jour la liaison SE/EP
```

3 Mise à jour des jobs

Si les fichiers GEO viennent à changer de format, il sera nécessaire d'intervenir à plusieurs endroits dans le job.
En préambule, ceci requiert de bonnes compétences Talend.

Tout d'abord il faut modifier les metadonnées (dans la marge gauche, référentiel) (schémas des tables,schémas génériques). Il faut propager le changement à tous les jobs (opération automatiquement lancées).
Ensuite, il faut pour chaque composant faisant référence à un fichier GEO ou une table destination, vérifier que les modifications ont bien été prise en compte. Si ce n'est pas le cas, faire les ajouts des colonnes manquantes, ou en trop, refaire les liens. Si un composant est oublié, le job provoquera une erreur ou l'information de sera pas remontée. Il sera alors facile de remonter les processus et de corriger.

Exemple : ajout d'une colonne dans ARTERE_GEO.

1. Modification des metadonnées
2. Job import_shp, vérifier schéma de artere_shp_imp
3. Job tmp_to_imp, vérifier schéma de artere_shp_imp et artere_imp
4. Job acteur, vérifier schéma de artere_imp
5. Job imp_to_db, vérifier schéma de artere_imp, artere_interm, artere
6. Création des câbles / fourreaux : vérifier schéma de artère
7. Si la nouvelle colonne est une nouvelle information pour câble ou fourreau, alors bien sur, il faut vérifier les schémas de ces deux tables.
8. Etc, jusqu'à la fin du job, et la même chose pour le job main_check.

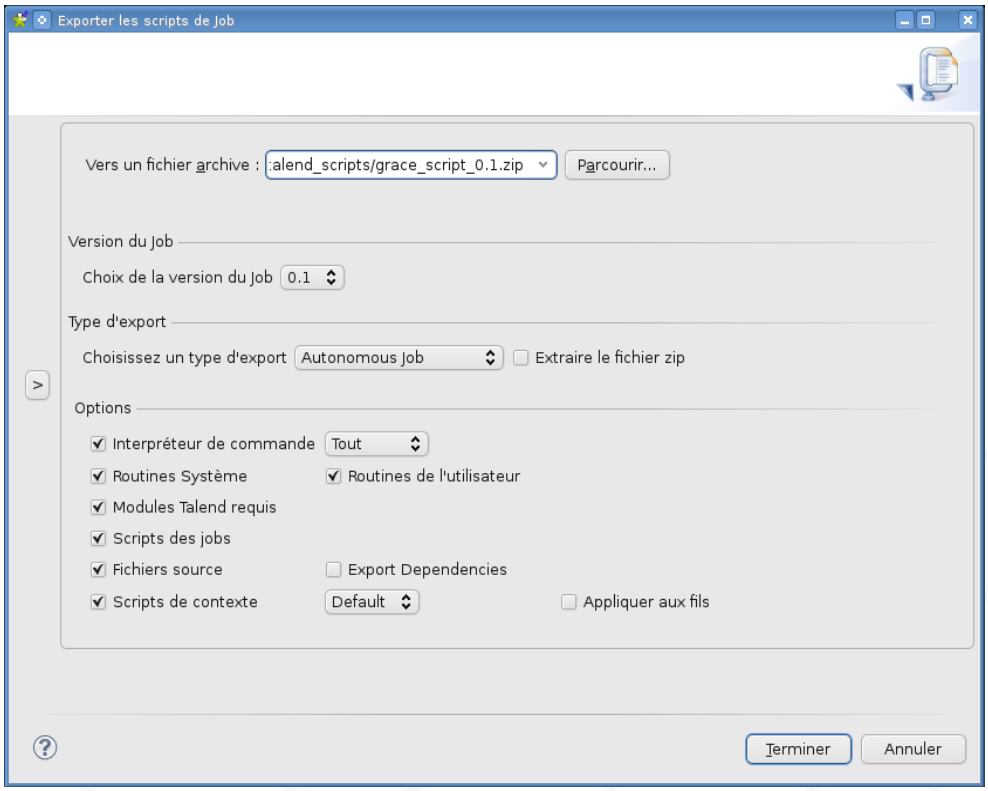
C'est beaucoup de choses à vérifier, mais c'est néanmoins nécessaire. Pour ne pas oublier de composant, le mieux est de suivre le déroulement des jobs et de faire les vérifications / modifications au fur et à mesure, et de vérifier (sur une base locale bien entendu) que les nouvelles informations se propagent bien.

Si s'agit de rajouter une information nécessitant des traitements plus complexe (par exemple, un calcul), alors il faudra connaître parfaitement l'intégralité des jobs afin de faire les modifications requises.

4 Génération de scripts exécutables

Les traitements peuvent être lancés depuis l'environnement Talend, ou bien depuis des scripts exécutables. Ceux-ci peuvent être générés de la manière suivante :

- Dans la marge à gauche, faire un clic droit sur le job Main et choisir l'option « Exporter les scripts de jobs »



- Préciser le chemin et le nom désiré pour l'export
- Cliquer sur terminer

Un fichier ZIP est créé. Il contient de manière autonome tout le code nécessaire à l'exécution des processus. Pour l'utiliser, il suffit d'extraire le ZIP sur le serveur (où sur la machine de traitement). Dans le répertoire « Main », le fichier Main_run.bat (ou Main_run.sh sous Linux) est à lancer pour débuter l'intégration des données. Dans le répertoire « Grace » se trouvent les sous-répertoires des sous-jobs utilisés, et dans chacun la définition des contextes (fichiers « Default.properties »). Ces fichiers doivent être modifiés et adaptés (si nécessaire) pour le bon fonctionnement du traitement.