Équipe informatique
PAU

Documentation

CyrusStar

CYRUS CONSEIL

Remerciements :

Nous tenons à remercier les fournisseurs sans lesquels ce flux n'aurait jamais vu le jour.

Table des matières

1 Introduction

CyrusStar est un ensemble d'applications permettant de créer un flux de valeurs liquidatives de fonds, puis de l'observer via une application Web.
On peut considérer que le flux global appellé flux_vls ($\approx$ 20 000 fonds) est ensuite restreint à l'ensemble des fonds de CYRUS ($\approx$ 1 000 fonds). L'application web porte uniquement sur la partie restreinte du flux. La présente documentation se divisera donc en deux parties. La première partie sera consacrée au flux de données en lui même, obtention des données, intégration des données et agrégation des données afin ne fournir à l'utilisateur que les données susceptibles de le concerner. La deuxième partie sera quant à elle consacrée à l'application web CyrusStar.
Chacune des parties sera divisée en trois sous parties :

- Spécifications.
- Rôle et identifiaction des différents fichiers.
- Maintenance de l'application.

2 Spécifications

2.1 Spécification fonctionnelle

2.1.1 Présentation

Nous nommons flux_vls l'ensemble des applications gérant les différents processus allant de l'obtention des données jusqu'à l'agrégation des données stockées. Les données sont les valeurs liquidatives datées des fonds trouvés. Nous appelons fonds un portefeuille de titres géré par un professionnel et proposé par un établissement financier ou une banque. Flux_vls est donc composé d'un processus d'obtention des données, un processus d'intégration des données et un processus d'agrégation et de préparation des données. Il existe aussi un processus chargé de la maintenance de la base ou du moins ce qui peut être traité de manière automatique.

2.1.2 Processus d'obtention des données

Nous appelons donnée le tuple $\{ISIN$[1]$, devise, VL$[2]$, date\ de\ la\ valorisation\}$. L'isin est un code à 12 caractères permettant d'identifier les valeurs de manière internationale.
Les données sont obtenues par robot web.
Chaque soir le processus lance un robot web par fournisseur agrégé *(voir la liste des fournisseurs agrégés A.1 page 22)*.
Le robot web se connecte au site web du fournisseur.
Le robot web met les données au format « $ISIN; DEV; VL(ex : 10.00); jj/mm/aaaa$ » dans un fichier textué.
Le robot web crée donc un fichier par fournisseur.
Un robot web supplémentaire est destiné à obtenir les taux de conversion DEV→EUR avec le format « $DEV; jj/mm/aaaa; taux(ex : 10.00)$ » de manière à ce qu'on ait $1EUR = taux \times DEV$.

[1] International Securities Identification Number
[2] Valeur Liquidative

2.1.3 Processus d'intégration des données et base de données associée

Les fichiers créés par le processus d'obtention des données sont ouverts un par un et leur contenu est stocké dans une base de données appellée flux_vls. Le processus vérifie que les données ne sont pas trop absurdes. On entend par absurde le fait qu'un tuple candidat à l'insertion ne vérifie pas le bon format ou que la date de la valeur soit supérieure à la date du jour.

2.1.4 Processus d'agrégation et de préparation des données, base de données associée et calcul des taux d'évolution

L'ensemble des données stockées dans flux_vls est par ce processus restreint à l'ensemble des données qui concernent CYRUS CONSEIL (soit $\approx$ 1 000 fonds). Les données qui concernent CYRUS CONSEIL sont les valeurs des supports dont le montant souscrit est supérieur à 0 et possédant un ISIN. La discrimination se fait à partir de la base du système d'informations ajourd'hui appelée BDD8DEFSQL.
La base de données contenant les données restreintes sera appelée flux_vls_eur. flux_vls_eur est différente de la base contenant la totalité des données. Ainsi, si BDD8DEFSQL comporte des informations erronnées, la contamination ne se propage que dans flux_vls_eur. flux_ est ainsi protégée de toute contamination externe.
Les calculs des taux d'évolution s'effectuent à partir de flux_vls_eur. De cette manière on ne fait des calculs que pour les fonds dont on a besoin. Cela permet de diminuer le temps de calcul qui est déjà très long (5 heures). Le résultat des calculs est stocké dans une table prête à servir facilement.

2.1.5 Processus de maintenance de la base

Il existe un processus lancé quotidiennement qui sauvegarde et archive flux_vls et flux_vls_eur. Il est normalement fait un vacuum. Il faudrait ajouter à cela un réindexage total lancé de manière bimensuelle. Il existe un outil pour créer une table de performances arrêtées à une date.

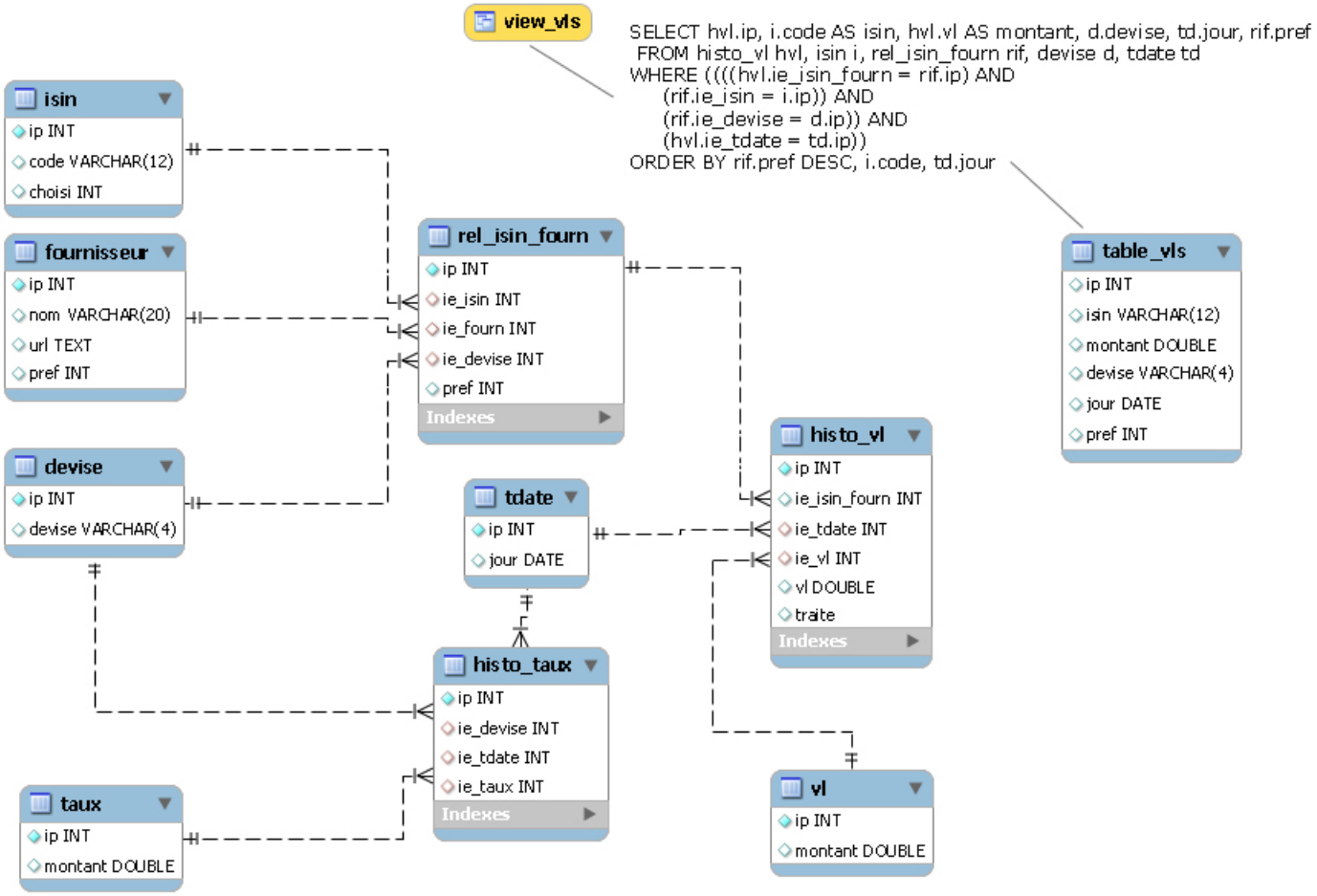
2.2 Rôle et identification des différents fichiers

2.2.1 Présentation

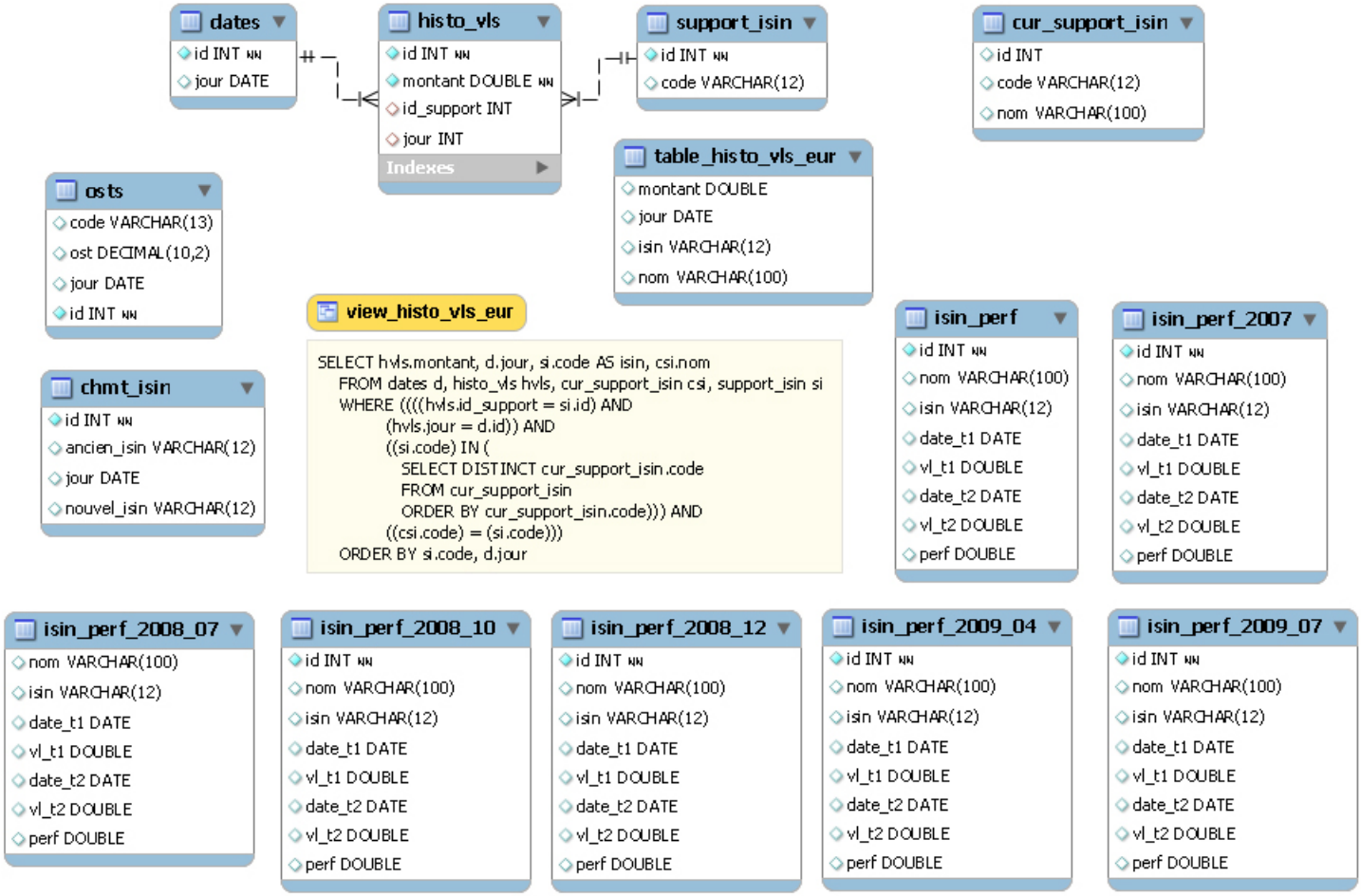
Nous allons dans cette section décire les différents fichiers intervenant dans le flux de valeurs liquidatives. Pour les bases de données on donnera simplement le diagramme.

2.2.2 Bases de données

Base de données flux_vls



Base de données flux_vls_eur



Programmes

/home/postgres/db/script/do_back_up.sh Lancé par le cron de postgres quotidiennement. Ici, $TODAY représente la date du jour. $TODAY est déterminé avec la commande bash :

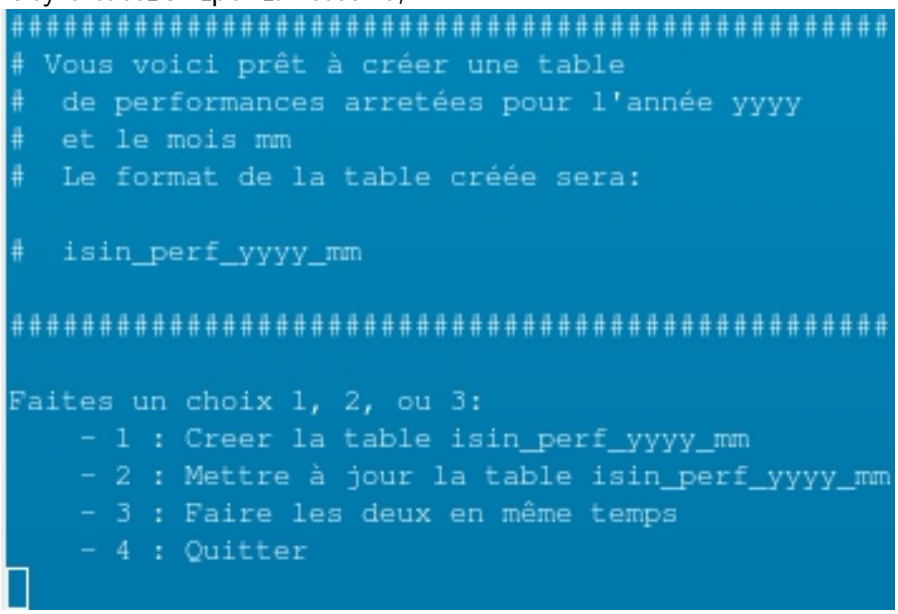
```
`date +"%Y%m%d-%H"`
```

Ce que fait le fichier :

1. Crée un répertoire /home/postgres/db/backup/$TODAY,
2. Sauvegarde la base de données flux_vls dans le fichier flux_vls$TODAY.sql via pg_dump,
3. Sauvegarde la base de données flux_vls_eur dans le fichier flux_vls_eur$TODAY.sql via pg_dump,
4. Le répertoire $TODAY est « taré » et « gzipé » via tar -cvzf,
5. Le serveur de bases de données est redémarré.

/home/malko/flux_vls/outils/create_isin_perf_arretee.rb Lancé manuellement avant chaque génération des états de compte.
Ce programme crée une table de la forme : isin_perf_yyyy_mm avec yyyy_mm l'année (2009 par exemple) et le mois (07 par exemple).
Ce programme s'utilise comme suit :
La table de performances arrêtées se crée le 02 ou le 03 du mois pour avoir les données du 1er du mois.

1. Se connecter sur la machine du flux (actuellement prodserver : 192.168.0.78),
2. Aller dans `/home/malko/flux_vls/outils`,
3. ruby create_isin_perf_arretee.rb,



4. Choisir 3 dans la plupart des cas,
5. Suivre ce qui est indiqué.

2.2.3 Processus d'obtention des données

Dans la suite de ce document nous appelons # `/home/malko/flux_vls/`.

Programmes

#/imports/import_vl.sh Lancé par le cron.

- Création du fichier `#/imports/a_import.py` par le script `#/imports/creation_fichier_import.py`.
  Du code python est créé pour gérer chaque robot.
  Exemple du code créé dans le fichier a_import.py (ici gestion du robot cardif) :

```
import os
import datetime
import fournisseurs.import_cardif
global repertoire_fichiers
repertoire_fichiers = '/home/malko/flux_vls/fichiers/'
def file_exists(fichier):
    try:
        file(fichier)
        return True
    except:
        return False
if file_exists('log/imports.log'):
    os.remove('log/imports.log')
file_log = open('log/imports.log', 'a')
try:
    print "debut " + "cardif"
    fournisseurs.import_cardif.import_cardif()
except Exception, e:
    print "!!!!!!!!!!!!!! cardif a plante"
    print e
    fichier = '/home/malko/flux_vls/imports/' +
        'fournisseurs/import_cardif.py\n'
    file_log.write(fichier)
file_log.close()
```

– Lancement du script #/imports/a_import.py.
  L'historique des évènements s'enregistre dans le fichier #/imports/log/imports.log.
  Ce script lance les robots.
– Lancement de #/import_ds_bdd.rb

robots Les robots sont dans le répertoire #/imports/fournisseurs/. Un robot a comme nom de fichier import_fournisseur.py où fournisseur est le nom du fournisseur. Un robot peut avoir aussi un fichier import_fournisseur.rb mais il faut impérativement qu'il y ait un fichier import_fournisseur.py. Un fichier python contient obligatoirement une méthode comme :

```
def import_fournisseur():
        #code du robot ou
        #appel du robot ruby import_fournisseur.rb
```

Les robots utilisent les fichiers #/imports/fournisseurs/support.py pour les robots entièrement en python et #/imports/fournisseurs/support.rb pour les robots mi–python/mi–ruby. Ces fichiers contiennent les classes $Support(isin, devise, vl, date)$.
Le robot morningstar est particulier : il se trouve dans le répertoire #/imports/fournisseurs/robot_morningstar et est lancé séparément par son propre cron.

#/imports/manage_robot.sh Ce que fait ce script :

– Efface les fichiers python et ruby du fournisseur. Pour créer les fichiers python et ruby il faut avoir enlevé du répertoire #/imports/fournisseurs/ les fichiers import_fournisseur.py et import_fournisseur.rb car on veut être sûr de ne pas supprimer un fichier par inadvertance.
– Crée les fichiers python et ruby nécessaires au robot fournisseur à partir des modèles #/imports/fournisseurs/modele/import_modele.[py|rb].
– Ouvre les fichiers python et ruby du robot fournisseur.

```
NAME
        manage_robot.sh - manage robots for creating, opening or deleting
SYNOPSIS
        manage_robot.sh [open|clean] fournisseur;
DESCRIPTION
         manage the robot named fournisseur
        open opens both of scripts named import_fournisseur.[py|rb]
                *.py files are opened with emacs
                *.rb files are opened with scribes
        clean deletes with rm both of scripts named import_fournisseur.[py|rb]
AUTHOR
        Pascal AUREGAN aka KisSCoOL
```

#/fichiers/import_fournisseur.txt | taux_ecb.txt Ce sont les fichiers produits par les robots. Ces fichiers sont importés puis archivés.
Exemple de fichier import_fournisseur.txt

```
FR0010129114;63.07;EUR;28/07/2009
```

Exemple de fichier taux_ecb.txt :

```
LTL;30/07/2009;3.4528
```

Fichiers logs

#/imports/log/imports.log La partie de ce fichier qui nous intéresse ici est la partie liée aux téléchargements des robots. Lorsque un robot plante, une première indication est écrite.
Exemple d'une ligne de log pour ecofi :

```
debut ecofi
"ECOFI:::FR0010706549::: non trouve"
```

#/imports/log/log_sp.log On trouve dans ce fichier les liens morningstar visités.

2.2.4 Processus d'intégration des données

Programmes

#/import_ds_bdd.rb Ce programme importe les fichiers présents dans `#/fichiers/`. Les taux taux_ecb.txt sont importés en premier. Les fichiers import_fournisseur.txt sont importés ensuite.

#/imports/backup_import_files_bdd.rb Ce programme archive dans `#/fichiers/backup` les fichiers présents dans `#/fichiers/`. Les archives créés à partir de « tar -cvzf » se nomment sous la forme yyyymmdd-hh.tgz où yyyy, mm, dd, hh représentent respectivement l'année (ex : 2009), le mois (ex : 05 pour le mois de Mai), le jour (ex : 31) et l'heure (ex : 02 pour 2 heures du matin).

Fichiers logs

#/imports/log/imports.log

#/import_ds_bdd.rb Exemple pour l'import d'ecofi :

```
::2009-08-01 08:59 => updating database...
"00:48 => debut ecofi"
"00:50 => fin ecofi"
::2009-08-01 08:59 => database updated
```

#/imports/backup_import_files_bdd.rb Exemple de sauvegarde :

```
::2009-08-01 02:43 => changing directory to
        /home/malko/flux_vls/imports/ successed
::2009-08-01 02:43 => creating 20090801-02 directory ...
::2009-08-01 02:43 => moving files to 20090801-02 directory...
20090801-02/
20090801-02/import_ecofi.txt
::2009-08-01 02:43 => directory saved
```

2.2.5 Processus d'agrégation, de préparation des données et calcul des taux d'évolution

Programmes

#/imports/import_mssql/import_mssql.sh (Lancé par le cron) Lance successivement `#/imports/import_mssql/import_mssql.rb`, `#/imports/import_mssql/comp_perf.rb`, `#/mail/mail.py`

Sur ServeurBDD :

C :\workspace\python\supports_utilises\script\do_maj_supports.bat

Lancé quotidiennement à 22h30 par le gestionnaire de tâches planifiées de $Windows^{®}$. Lance le script C :\workspace\python\supports_utilises\script\supports_utilises.py

C :\workspace\python\supports_utilises\script\supports_utilises.py

Crée le fichier http://192.168.0.1:81/supports/supports.xml qui se trouve dans C :\workspace\python\supports_utilises\rendu à partir de la base de données « BDD8DEFSQL » . Le script se connecte à la base « BDD8DEFSQL » et serialise en xml les données de la vue « used_supports » . La vue « used_supports » est le résultat de la requête suivante :

```
SELECT DISTINCT TOP (100) PERCENT dbo.supports.ISIN,
dbo.supports.id_support, dbo.supports.nom, l.libelle
FROM          dbo.libelles AS l INNER JOIN
                      dbo.supports LEFT OUTER JOIN
                      dbo.contrat_support ON dbo.supports.id_support =
dbo.contrat_support.id_support ON l.id = dbo.supports.flux_label
WHERE      (dbo.supports.ISIN IS NOT NULL)
GROUP BY dbo.contrat_support.id_support, dbo.supports.ISIN,
dbo.supports.id_support, dbo.supports.nom, l.libelle
HAVING       (SUM(dbo.contrat_support.montant) > 0) OR
                      (l.libelle NOT LIKE 'BASIQUE')
ORDER BY dbo.supports.id_support
```

http ://192.168.0.1 :81/supports/supports.xml

Voici un exemple de ce fichier :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<supports>
        <support
                isin="FR0010551424"
                id_support="5711"
                nom="DARWIN DIVERSIFIE 60-80 1/1.000"
                flux_label="BASIQUE"
        />
</supports>
```

flux_label prend ses valeurs dans $\{"BASIQUE","PERF","PERF\_ET\_MORN"\}$

- $"BASIQUE"$ désigne un fonds dont le montant total agrégé est strictement positif.
- $"PERF"$ oblige le flux à récupérer la vl du fonds. Il faut utiliser ce label lorsque le fonds est déjà dans flux_vls mais que le montant total agrégé est nul (et si on a besoin de la performance du fonds).
- $"PERF\_ET\_MORN"$ oblige le flux à récupérer la vl du fonds par le robot morningstar. Il faut utiliser ce label lorsque le fonds n'est pas dans flux_vls mais que le montant total agrégé est nul (et si on a besoin de la performance du fonds).

Il faut bien évidemment penser à mettre à jour les labels des supports pour ne pas faire de requêtes inutiles.

#/outils/chmt_isin/chmt_isin.sh Lancé par le cron. Met à jour les données relatives aux changements d'isin et aux divisions de valeurs liquidatives des fonds.
Ce que fait ce fichier :

- Se connecte à la base « outil_produit » sur le serveur 192.168.0.74,
- Sérialize dans `#/outils/chmt_isin/chmt_isin.txt` la table « changer_isin » ,
- Remplit la table « chmt_isin » de la base les données « flux_vls_eur » à partir des données de `#/outils/chmt_isin/chmt_isin.txt`,
- Sérialize dans `#/outils/chmt_isin/osts.txt` la table « splits »,
- Remplit la table « osts » de la base de données « flux_vls_eur » à partir des données de `#/outils/chmt_isin/osts.txt`.

#/imports/import_mssql/import_mssql.rb Se sert du fichier `http://192.168.0.1:81/supports/supports.xml` pour connaître la liste des fonds dont la performance doit être calculée.
Ce que fait ce fichier :

- vide puis remplit la table « table_vls » de « flux_vls » à partir de la vue « view_vls »,
- remplit la table « cur_support_isin » à partir de `http://192.168.0.1:81/supports/supports.xml`,
- fait passer toutes les vls connues de ces fonds dans la base de données « flux_vls_eur »,
- vide puis remplit la table « table_histo_vls_eur » de la base « flux_vls_eur » à partir de la vue « view_histo_vls_eur ».

#/imports/import_mssql/comp_perf.rb Calcule les performances pour les fonds de la table « cur_support_isin » qui sont présents dans la base de données flux_vls. Se sert de la table « osts » et « chmt_isin ». La performance des fonds au jour j de l'année n est calculée entre le 31 Décembre de l'année $n-1$ — ou la date de création du fonds si le fonds est créé l'année durant — et la dernière date de valorisation connue.

#/mail/mail.py Récupère tous les logs de `#/imports/log` et de `#/imports/import_mssql/log` et les envoie par mail aux destinataires choisis.

Fichiers logs

#/imports/import_mssql/log/log_cron.log Fichier log du cron pour le `#/imports/import_mssql/import_mssql.sh`.
Généralement rien ne se passe dans ce fichier.

#/imports/import_mssql/log/log_mssql.log Fichier log de ce qui est fait dans #/imports/import_mssql/import_mssql.rb.
Il y a très rarement des problèmes.

#/imports/import_mssql/log/log_comp.log Fichier log de ce qui est fait dans #/imports/import_mssql/comp_perf.rb. Il y a très rarement des problèmes.

#/imports/import_mssql/log/log_mail.log Fichier log de l'envoi des mails quotidiens de compte–rendu. Il y a très rarement des problèmes.

3 Maintenance de l'application

Cette partie sera présentée sous forme de F.A.Q (Frequently Asked Questions).

3.1 Comment arrêter la situation à une date choisie

voir 2.2.2

3.2 Comment ajouter une situation

voir 2.2.3 Créer le fichier comme indiqué et importer le fichier avec `#/imports/backup_import_files_bdd.rb`. Si on a besoin de la performance avant le lendemain on lance `#/imports/import_mssql/import_mssql.sh` sinon on laisse faire le flux et le lendemain les performances seront à jour.

3.3 Comment ajouter ou réparer un robot ?

Pour ajouter un robot :

- Vérifier que l'on a bien le tuple $\{ISIN^1, devise, VL^2, date\ de\ la\ valorisation\}$ sur le site. Si on n'a pas exactement ce tuple, il ne faut pas faire de robot. La date de valorisation doit être inscrite sur le site : elle ne doit pas être déduite à partir de la date du jour.
- Penser à injecter les valeurs des fonds au 31/12 pour que la performance calculée soit juste.
- Implémenter le robot (voir 2.2.3)

Pour tester puis réparer le cas échéant le robot :

[1] International Securities Identification Number
[2] Valeur Liquidative

```
cd "#/imports";
#fournisseur = nom du founisseur
#fournisseurs = repertoire fournisseurs
#et non pas nom du fournisseur avec un s
python test_fourn.py fournisseurs/import_fournisseur.py;
```

Le flux lancera automatiquement le robot le lendemain.

Suggestions : il faudrait finir le robot acropole.

3.4 Comment vérifier les résultats ?

Regarder le mail envoyé chaque jour et réparer ce qui ne va pas.
Si lcf ne fonctionne pas, c'est certainement qu'il faut relancer quelques firefox -jssh : en lancer un dans un terminal puis quelques autres à partir d'un autre terminal.
Le site des caisses d'épargnes ne fonctionne plus (ce) : il manque des données. Lodh ne fonctionne plus non plus ainsi que Crédit suisse et suravenir. Il faut aller régulièrement sur leur site respectif pour voir si on ne peut pas réintégrer un robot.
Un fonds de Cardif n'est pas trouvé. Pour euro tunnel dans la partie calcul de performance, il s'agit d'un problème de l'amf. Il faut savoir aussi que l'amf et afer (et pourquoi pas d'autres ?) nous donne parfois des valeurs liquidatives valides dans plusieurs années.
Si la machine plante au milieu de la nuit, regarder dans `#/fichiers/` si les fichiers d'imports sont présents. Si les fichiers sont présents, taper la commande suivante en se plaçant dans # :

```
ruby import_ds_bdd.rb
```

Le calcul des performances peut attendre le lendemain.

3.5 Comment traiter les OST/Changement ISIN ?

voir 2.2.5

3.6 Comment s'occuper du flux ?

Il faut être convaincu qu'on est sûr de rien. Par conséquent, il faut tout vérifier avant de faire quoique ce soit même si on pense savoir. Si il nous est rapporté qu'il y a une erreur, on vérifie toujours et on ne rentre pas des valeurs pour faire plaisir à un tel ou à un autre. Si on pense qu'il n'y a pas d'erreur, on vérifie quand même : on a parfois des surprises.

3.7 A quelle heure peut on vérifier les données ?

Le mail de logs arrive généralement vers 12h50. Si l'heure d'arrivée du mail s'écarte trop de cette heure, c'est qu'il doit y avoir un problème. Il faut effectuer les modifications jusqu'à 18h environ. Il ne faut pas redémarrer la machine avant 17h30 car le robot Morningstar tourne. toto

4 Conclusion

Bonne chance

A Annexes

A.1 Fournisseurs agrégés

A.1.1 Classés par ordre alphabétique

- ADEQUITY
- AFER
- AGF asset management
- AMF
- AVIVA asset management
- AXA
- BNP asset management
- BOURSORAMA
- BANQUE POPULAIRE asset management
- CRÉDIT AGRICOLE asset management
- CARDIF
- CARMIGNAC
- CAISSE D'ÉPARGNE
- CIC
- CPR asset management
- CRÉDIT SUISSE asset management
- ECB *(pour les devises)*
- ECHIQUIER
- ECOFI
- EURONEXT
- FIDELITY
- FORTIS
- TEMPLETON
- GAN
- GENERALI
- HSBC
- INVESCO
- La Tribune
- LCF *(La Compagnie Financière Edmond Rothschild)*
- Le Crédit Lyonnais asset management
- LODH
- LYXOR

- MMA
- Morningstar
- ODDO
- OFFIVALMO
- Banque d'Orsay
- RCB *(Rothschild et compagnie)*
- RICHELIEU
- SOCIETE GÉNÉRALE asset management
- S&P *(Standards and Poors)*
- SURAVENIR
- SYCOMORE
- UBS asset management
- UFG asset management

A.1.2 Classés par priorité décroissante

(source : base de production)

id	nom du fournisseur	priorité	id	nom du fournisseur	priorité
46	adequity	44	22	cpr_am	21
43	rcb	43	21	euronext	20
42	orsay	42	20	gan	19
41	cic_sicav	41	19	ce	18
40	morningstar	40	18	bp_am	17
39	lodh	38	17	sycomore	16
38	boursorama	37	16	mma	15
37	sp	36	15	suravenir	14
36	latribune	35	14	generali	13
35	afer	34	13	fidelity	12
34	richelieu	33	12	oddo	11
33	ubs_am	32	11	aviva_am	10
32	lyxor	31	10	cs_am	9
31	lcl_am	30	9	agf_am	8
30	ofivalmo	29	8	fortis	7
3	axa_im	28	7	lcf	6
28	hsbc	27	6	invesco	5
27	bnp_am	26	5	echiquier	4
26	ecofi	25	4	ufg_am	3
25	ca_am	24	29	cardif	2
24	fr_templeton	23	2	sgam	1
23	carmignac	22	1	amf	0