Table des matières

[1. Description générale du « Replicator » 2](#_Toc29237925)

[2. Architecture de l’outil 2](#_Toc29237926)

[A) Schéma 2](#_Toc29237927)

[B) Briques logicielles 2](#_Toc29237928)

[3. Architecture fonctionnelle 3](#_Toc29237929)

[A) Client 3](#_Toc29237930)

[B) Back 3](#_Toc29237931)

[C) Serveurs de bases de données 3](#_Toc29237932)

[4. Principes de fonctionnement 3](#_Toc29237933)

[5. Fonctionnalités 4](#_Toc29237934)

[6. Prérequis 5](#_Toc29237935)

[7. Cas d’usage 6](#_Toc29237936)

[8. Roadmap 6](#_Toc29237937)

# Description générale du « Replicator »

Le replicator est un outil qui permet de répliquer des données d’une base de données à une autre.

Cet outil permet de choisir dans une base de données source des tables et colonnes afin de pouvoir exporter les données et les réinsérer dans une autre base de données qui peut être située sur un autre serveur.

Cet outil est léger, portable et compatible avec différentes plateformes.

Il fonctionne de manière autonome.

# Architecture de l’outil

## Schéma

## Briques logicielles

* Socle docker
* RabbitMQ
* Service de publication de Web Services en mode REST
* Cellery, Flower : service d’ordonnancement de tâches
* MySQL, Oracle, Vertica, PostgreSQL
* Compilateur C : gcc, cc
* Framework
  + Xcode
  + Clion
* Bibliothèques, langages et outils utilisés
  + Python 3
  + Gtk3
  + json-c
  + libmysql
  + libxml
  + libcsv
  + mysql-client et bibliothèque de développement C
  + vertica-client et bibliothèque de développement C
  + postgre-client et bibliothèque de développement C
  + oracle-client et bibliothèque de développement C
* Outils utilisés
  + DBeaver
  + MySqlWorkbench
  + Atom
  + Transmit
  + gitKraken
  + Postman (test des Web Services)

# Architecture fonctionnelle

Architecture 3-Tiers (potentiellement une architecture à n niveaux)

1. Client (front) : ihm utilisateur (ihm : interface homme machine)
2. Serveur d’application (back) : replicator
3. Serveur base de données

## Client

Le client fait des requêtes en Web Services au serveur d’application pour avoir la liste des tables d’un schéma donné.

L’utilisateur, graphiquement, va pouvoir, dans un premier temps, sélectionner un schéma, des tables et des colonnes à répliquer, puis définir la base cible et le schéma de destination.

## Back

1. Un service de Web Services qui permet de faire des interrogations aux bases de données
2. Un service d’orchestration de tâches (mise en file d’attente des exports et imports de bases de données)
3. Un service de suivi des tâches
4. Un service de réplication de données « le Replicator »

## Serveurs de bases de données

Des moteurs de bases de données de type :

1. MySQL
2. PostgreSQL
3. Vertica
4. Oracle
5. …

# Principes de fonctionnement

Ce service de réplication permet un fonctionnement en mode asynchrone : le client donne des tâches à effectuer à un serveur qui effectue ces traitements de façon autonome. Le client interroge à intervalles réguliers le serveur de traitement pour avoir l’état des tâches demandées.

**Les étapes :**

1. Les utilisateurs définissent les données à répliquer
2. Le client envoie en Web Service un fichier d’ordres qui va permettre au serveur de reconnaitre les informations nécessaires à la connexion aux bases de données, ainsi que les tâches à effectuer
3. Le client envoie en Web Service une tâche d’export à l’ordonnanceur
4. L’ordonnanceur déclenche le programme de réplication avec comme paramètre l’identifiant de la tâche à exécuter (permet d’identifier le fichier d’ordres envoyé préalablement)
5. Le réplicateur génère un fichier qui contient l’extraction des données
6. Le client interroge en Web Service l’ordonnanceur pour connaître le statut de fin de tâche de l’export des données
7. En cas de réussite de la tâche d’export, le client envoie en Web Service une tâche d’import à l’ordonnanceur
8. Le client interroge en Web Service l’ordonnanceur pour connaître le statut de fin de tâche de l’import des données

Nb :

Le client peut définir plusieurs tâches de réplication sans avoir besoin de rester connecter avec le serveur.

Fonctionnement en mode asynchrone.

**Avantages de cette architecture :**

* Le client n’a pas besoin d’avoir des connexions aux bases de données
* Le back a un accès privilégié et performant pour faire des imports/exports de données volumineuses
* Le fonctionnement des imports/exports est asynchrone

# Fonctionnalités

Actuellement, le replicator écrit entièrement en langage C, permet :

* Export mySQL vers mySQL / vertica vers vertica
  + D’exporter des données d’une base mySQL vers une autre base mySQL en utilisant les commandes d’import et d’export fournies par le SGBD (mysqldump et mySQL)
    - Avantages :
      * Pouvoir exporter l’intégralité des tables choisies avec leurs spécifications (indexes, contraintes d’intégrité, …)
      * Solution éprouvée et performante
      * Évolution des outils en fonction des montées de versions SGBD
    - Inconvénient :
      * Impossible de changer de type de base de données
* Export mySQL vers vertica
  + D’exporter des données d’une base mySQL avec une sélection préalable des tables et des colonnes vers une base vertica
    - Avantages :
      * Sélectionner finement les données à répliquer
      * Pouvoir les importer dans un type de base différent
    - Inconvénients :
      * Moins performant que les outils fournis par les SGBD
      * Complexe à développer pour retranscrire les spécifications d’une base à une autre (régénération des indexes, des différents types de données (format date, …), les contraintes d’intégrité, …)
      * Compliqué de suivre les évolutions des SGBD
* Extraction de données ciblées (tables et colonnes spécifiques) en différents formats pour pouvoir les archiver ou bien faire des rapports (csv, sql, json, xml).

# Prérequis

* Un fichier de configuration utilisé par le client, qui permet de connaître :
  + Les serveurs
  + Les connexions (ports, utilisateurs, schémas, mots de passe)
* Un OS de type linux/MacOS. 2Go de mémoire minimum pour le back et 10Go d’espace disque
* Installation d’un socle docker
* Une connexion internet
* Python version 3
* Installation des clients (outils et librairies de développement)
  + MySQL
  + Vertica
  + Oracle
  + PostgreSQL
* Installation du service d’ordonnancement des tâches et du suivi « TOM »
  + Cellery et Flower (Ordonnancement des tâches)
  + RabbitMQ (stockage des événements de suivi des tâches)

Nb : Outils utilisés lors de mon stage

* Installation du service de publication de Web Services « Api-generator »
  + Connexion aux différentes bases de données (récupération des structures des tables)
  + Service d’upload de fichier (fichier de commande uploadé sur le serveur de traitement par le client)

Nb : Outils utilisés lors de mon stage

* Installation des images docker
  + 2 conteneurs mySQL (3306, 3307)
  + 1 conteneurs Vertica (3308)

# Cas d’usage

* Cet outil peut être utilisé comme système de sauvegarde, comme système de fail-over
* Recopier des données de production pour les mettre sur des environnements de recette pour permettre de reproduire des anomalies de production ou réaliser/tester des évolutions applicatives
* Permet de créer un environnement pour du reporting sur des données de production récentes (les éditions prennent beaucoup de ressources systèmes, dégradent les temps de réponses et surchargent généralement le SGBD)

# Roadmap

* Revoir l’interface graphique pour être plus « user-friendly » (ex : voir le modèle de données de façon graphique)
* Compatibilité avec plus de bases de données (Oracle, Postgre)
* Prendre en compte les spécifications de chaque SGBD pour pouvoir recréer les tables avec la même structure que la base source
* Utiliser un parser csv pour l’export des données en format fichier
* Portage du client et du back sous Windows
* Transfert des exports inter-serveurs (facile à intégrer)
* Faire un écran qui permet de visualiser l’état (le statut) des tâches effectuées (volumétrie, temps d’exécution, …)
* Compression des fichiers d’export