YouCode / Youssoufia

Développeur Data

2023-2024

Projet : **Evaluation (Sprint Talend)**

**YONLI Fidèle**

Table des matières

[**Introduction** 2](#_Toc147397076)

[**Gestion de projet** 2](#_Toc147397077)

[**Processus ETL** 2](#_Toc147397078)

[**Extraction** 2](#_Toc147397079)

[**Transformations** 2](#_Toc147397080)

[**Modélisation du datawarehouse** 2](#_Toc147397081)

[**Load (Chargement)** 2](#_Toc147397082)

[**Chargement dans les tables de dimension** 2](#_Toc147397083)

[**Chargement dans les tables de fait** 3](#_Toc147397084)

[**Tests Unitaires** 3](#_Toc147397085)

[**Création des datamarts** 3](#_Toc147397086)

# **Introduction**

Ce projet vise à établir un entrepôt de données en utilisant SQL Server pour un agent de e-commerce. Il implique également l'utilisation de Talend pour l'ETL, en extrayant, transformant et chargeant les données dans l'entrepôt. Une constellation rapide de schémas est mise en place pour les ventes et l'inventaire, avec des tables de dimensions et de faits détaillées. Des DataMarts physiques pour les ventes et l'inventaire sont créés, suivis d'une analyse approfondie à l'aide de Power BI, comprenant la tendance des ventes, l'analyse des produits, la segmentation des clients, et plus encore. Le projet comprend également l'optimisation de DataMarts par l'indexation, la partition, l'agrégation, et d'autres techniques. Des politiques de rétention de données sont définies, la logique de transformation est validée, et des autorisations appropriées sont mises en place pour assurer la sécurité des données.

# **Gestion de projet**

Pour mener à bien le projet, une organisation s’impose. Nous avons opté pour **Trello** comme plateforme de planification et d’organisation des différentes taches à réaliser. Ci-dessous une image de l’ensemble des taches réalisées et organisées.

*trello*

# **Processus ETL**

Tout part d’un processus ETL (Extraction-Transformations-Load). Nous l’avons implémenté sous Talend Data Integration. Chaque étape a un ensemble de taches et d’objectifs précis.

## **Extraction**

Cette étape constitue un job (**jExtraction**) à elle seule. Elle consiste à consolider (rassembler) les données issues des différentes sources en une vue unique. Pour ce faire, nous avons utilisé le composant **tUnite**. Puis, nous avons stocké les données consolidées dans une base de données (staging area) nommée **SprintTalend. C’est** dans cette zone de stockage que nous allons réaliser toutes les transformations nécessaires (de nettoyage principalement) pour construire le datawarehouse.

*Image jExtraction*

## **Transformations**

Cette étape consiste à faire subir à nos données le processus de nettoyage avant de les stocker dans notre datawarehouse. Plusieurs opérations ont été réalisées à cet effet sur plusieurs colonnes à l’aide du composant **tMap**:

* Uniformiser le format de Date *dd-MM-yyyy* dans la colonne **Date**
* Remplacer *InvalidPrice* par *TotalAmount/QuantitySold* dans la colonne **ProductPrice**
* Remplacer les valeurs manquantes de **CustomerEmail** et **SupplierContact**
* Crypter les colonnes **CustomerEmail**, **CustomerAddress**, **CustomerPhone**, **SupplierContact**

Puis, les données transformées sont transférées au job suivant via le composant **tBufferOutput**.

*Image de jTransformations*

## **Modélisation du datawarehouse**

Avant de passer au chargement dans notre entrepôt de données, nous avons besoin de préparer le datawarehouse. Nous allons donc modéliser notre entrepôt de données qui servira de lieu de chargement de nos données. Nous avons opté pour un schéma **fact constellation** pour plusieurs raisons :

* Plusieurs types d’évènements ou faits (inventaire, vente notamment) doivent être analysés ensemble
* Flexibilité d’analyse multidimensionnelle en croisant les données de différentes tables de faits
* Evolutif dans le sens où l’on peut ajouter d’autres évènements à notre entrepôt de données sans affecter le schéma déjà existant

*Image de la modélisation*

## **Load (Chargement)**

Tout est enfin pour que nous puissions peupler le datawarehouse des données transformées. Le chargement va s’effectuer en deux étapes :

* Chargement dans les tables de dimension
* Chargement dans les tables de fait

### **Chargement dans les tables de dimension**

La création des dimensions se fait progressivement en s’appuyant sur notre modèle fourni. Nous avons principalement utilisé le composant **tMap** pour la sélection des colonnes par dimension. Ensuite, avec le composant **tUniqRow**, nous avons extrait les valeurs uniques. Puis, de nouveau à l’aide du composant **tMap**, nous générons et associons des identifiants à nos valeurs uniques pour enfin les stocker dans une table de notre datawarehouse avec le composant **tDBOutputBulkExec**.

*Image du composant jDimensions*

Les tables de dimensions sont créées. Nous allons pouvoir extraire leurs clés primaires pour les tables de faits.

### **Chargement dans les tables de fait**

Les tables de fait sont les éléments de notre modèle qui contiennent les mesures et les identifiants des tables de dimension. Techniquement, pour leur construction, nous avons d’abord effectué des jointures entre les tables de dimension, puis avons remplacé les valeurs provenant des tables de dimension par leurs IDs pour constituer les tables de fait. De plus, nous conservons les données quantitatives nécessaires à notre analyse comme des colonnes de mesure. Tout ceci à l’aide du composant **tMap**.

Finalement, à l’aide du composant **tDBRow**, nous avons créé les clés étrangères pour établir les relations qui existent entre les tables selon notre modèle établi.

*Image jChargement*

# **Tests Unitaires**

# **Création des datamarts**