Conservatoire National des Arts et Métiers
Pascal AUREGAN

Rapport de projet

NFE211

Test de chaine décisionnelle sur cas simple.
Base opérationnelle en fichier et PostgresSQL
ETL avec TALEND Data Integration,
Reporting avec SAS

le cnam

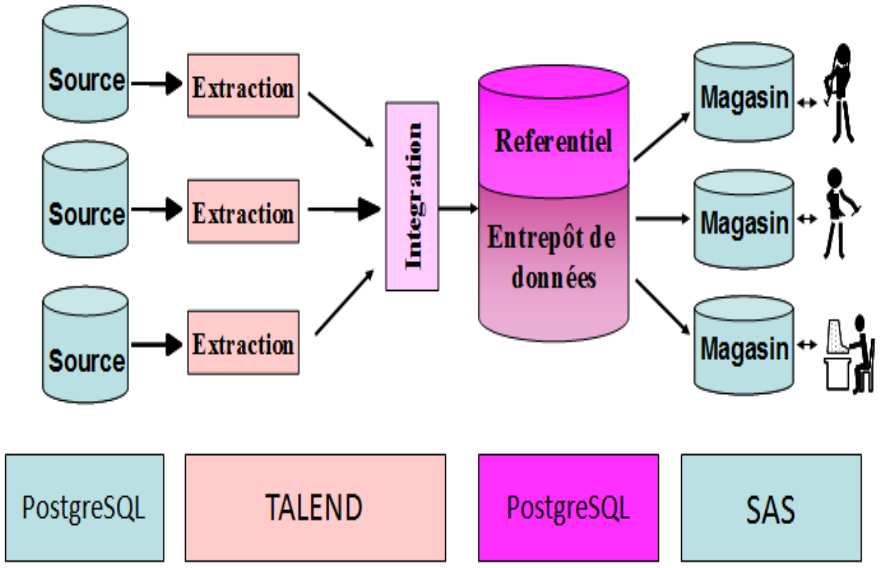
Table des matières

1 Introduction

Le but de se projet est de mettre en place une chaine décisionnelle. L'objectif de ce projet n'étant pas de pousser dans ses retranchements les outils de la chaine décisionnelle, il a été décidé de créer de toute pièce un jeu de données. Le thème de ce jeu de données est l'achat de BD comme dans le cours. La base opérationnelle décrit les factures d'un ensemble librairies en France dont les clients sont connus. La chaine décisionnelle doit être en mesure de répondre in fine à la question : analyser les ventes de BD en France en fonction des clients, du lieu et date d'achat. La couche opérationnelle est assurée par une base de données de fichier csv et une base de données PostgresSQL.

Pour la couche ETL, il a été choisi TALEND Data Integration.

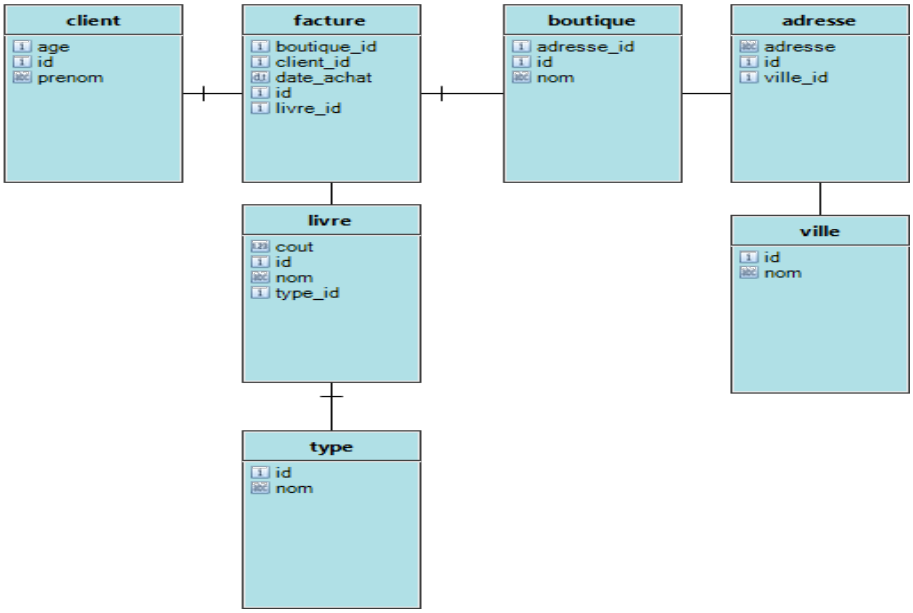
La couche reporting étudiée lors de ce projet est assurée par SAS Studio Version Etudiante.



2 Jeu de données de la base OLTP

2.1 Modèle relationnel

2.1.1 Présentation



Le modèle relationnel est un modèle normalisé.

— Une facture a été établie pour un client donné dans une boutique donnée à une date donnée et en rapport avec un livre donné.

— Le livre appartient à un type donné (Roman, BD, etc.).

— La boutique a une adresse située dans une ville

Voici combien de lignes ont chaque table :

— adresse : 26

— boutique : 25

— client : 8819

— facture : 8819

— livre : 8

— type : 3

En annexe, le code R qui a servi à générer les données dont il a été estimé que la pertinence n'avait que peu d'importance pour l'exercice. De plus, nous possédons un fichier de type csv contenant un dictionnaire de villes avec les départements associés ainsi que les régions. Voici en exemple les quatre premières lignes de ce fichier villes_def.csv

```
"ville","departement","region_name"
"Beauvais","Oise","Picardie"
"Compiegne","Oise","Picardie"
"Clermont","Oise","Picardie"
```

3 TALEND Open Studio for Data Integration 5.6

3.1 Création du modèle OLAP

3.1.1 Question à répondre

Analyse des ventes de bande dessinées d'une enseigne de librairie possédant plusieurs boutiques dans plusieurs villes.

3.1.2 Modèle ROLAP

L'implémentation ROLAP a été choisie par rapport au modèle MOLAP car même si la quantité de données était petite, il m'a semblé important d'étudier ce modèle qui paraît être très utilisé. Ne voyant rien justifier un modèle en flocon permettant de gagner de l'espace de stockage, j'ai utilisé un modèle en étoile. La table de faits est la table vente. Les dimentsions sont le temps, la boutique(lieu d'achat), le client, et le livre. L'étoile est de type transaction.

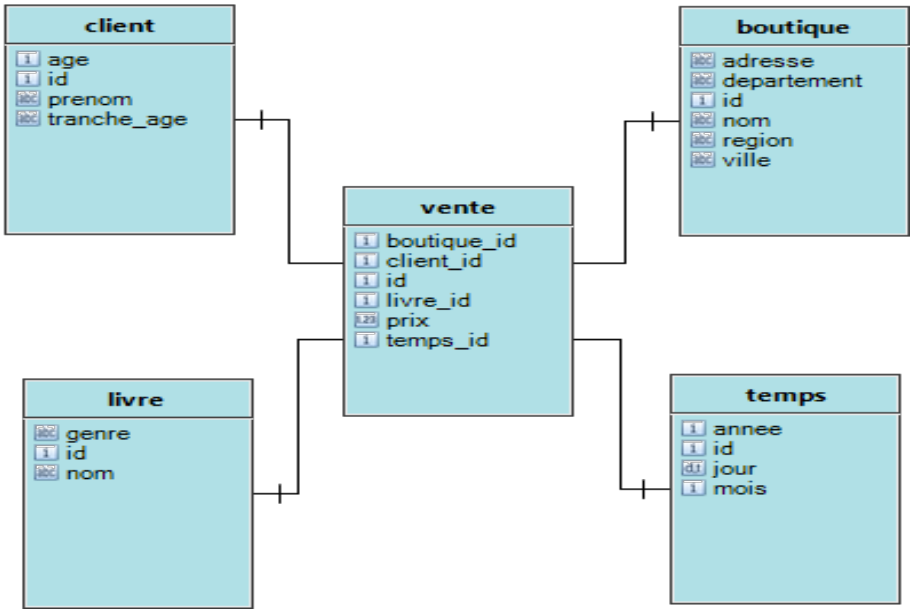


Table de faits vente

Indicateurs prix de la vente ; ce que le vente a rapporté

fonctions d'agrégat

— addition sur le prix

— moyenne sur le prix

Dimension client

— id

— prenom

— age

— tranche_age ("-12", "12-25", "35-65", "25-34", "+65")

hiérarchie age $<$ tranche_age

Dimension boutique

— id

— nom

— adresse

— departement

— ville

— region

hiérarchie adresse < ville < department < region

Dimension livre

- id
- nom
- genre

hiérarchie nom < genre

Dimension Temps

- id
- jour
- mois
- annee

hiérarchie jour < mois < annee

3.2 Intégration avec TALEND

3.2.1 Téléchargement Installation

Le téléchargement se fait sur http://fr.talend.com/download/data-integration. La documentation de l'outil est disponible au même endroit dans la partie Manuels d'utilisateurs et est facilement accessible. L'outil est en fait basé sur Eclipse ce qui le rend facilement appréhendable pour ceux qui connaissent le célèbre IDE mais vient aussi avec ses défauts.

3.2.2 Méthode utilisée

Pour chaque dimension, j'ai essayé d'avoir un cas d'usage différent :

- La dimension livre se construit par une jointure classique entre deux tables.
- La dimension client est construite en utilisant la table client suivie d'une transformation sur l'un de ses champs afin de déduire la tranche d'age à partir de l'age.
- La dimension boutique utilise des sources de données de type différents (CSV et postgres).
- La dimension temps construite à partir de la table temps subit un action sur ses données afin d'enlever les doublons.

3.2.3 Configuration des sources de données

Le csv

4 Reporting

4.1 SAS

4.1.1 Présentation

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

4.1.3 Processus de maintenance de la base

5 Conclusion

Bonne chance

A Annexes

A.1 Code R ayant généré les données

```
library(dplyr)
library(tidyr)


prenoms <- read.csv('liste_des_prenoms_par_annee.csv', sep = ';')

head(prenoms)
hist(prenoms$nombre)
summary(prenoms$nombre)
hist(x = prenoms$annee)
nb_clients <- length(prenoms$prenoms)



X <- 10 + rnorm(nb_clients, mean=5, sd = 1)
Y <- 5 + rnorm(nb_clients, mean=20, sd = 2)

U <- rbinom(n=nb_clients, size=1, prob=0.6)
summary(U)



hist(X)



hist(Y)
Z <- U * X + ((U-1)*-1 + Y)
hist(Z)
```

```
age <- floor(Z)
clients <- cbind(prenoms, age)
head(clients)
hist(clients$age)


type<-floor(1+rnorm(nb_clients, mean=1.4, sd=0.6)%%3)
hist(type)
clients <- cbind(clients, type)

livres <- c(6, 7, 1, 2, 4, 8, 3, 5)

# distribution sur livres
livres_achetes <- floor(rnorm(nb_clients, mean = 3.5, sd=1.5))
livres_achetes <- sapply(livres_achetes, FUN = function(x) max(0,x))
livres_achetes <- sapply(livres_achetes, FUN = function(x) min(7,x))
hist(livres_achetes)



clients <- cbind(clients, livres_achetes)
head(clients)

# lien vers la table livre
livre <- sapply(livres_achetes, FUN=function(x)livres[x+1])
clients <- cbind(clients, livre)



hist(clients$livre)

boutiques <- floor(runif(nb_clients, min=1, max=25))
clients <- cbind(clients, boutiques)
hist(boutiques)

client_id <- 1:length(clients$prenoms)

clients <- cbind(clients, client_id)
```

```
#last date in the data
end_date <- as.Date('2015-01-01')

date_achat <- end_date - runif(n=nb_clients, max=365*5, min=1)



clients <- cbind(clients, date_achat)
clients <- mutate(clients, prenom = prenoms)

clients_for_csv <- select(clients, id=client_id, age, prenom)
write.csv(x=clients_for_csv, file='clients.csv', row.names=FALSE)
nb_clients + 1 - client_id

clients <- mutate(clients, facture_id = (nb_clients + 1 - client_id))

facture_for_csv <- select(clients, id=facture_id, date_achat, boutique_id=b
write.csv(x=facture_for_csv, file='factures.csv', row.names=FALSE)

villes <- read.csv(file='ville.csv')
departements <- read.csv('departement.csv')
regions <- read.csv('region.csv')
```

A.2 Routine java toTrancheDage.java

Listing A.1 – toTrancheDage.java

```
public class to_tranche_age {
        private enum TRANCHE_AGE{
                LT_12("-12"),
                B12_25("12-25"),
                B25_34("25-34"),
                B35_65("35-65"),
                MT_65("+65");

                TRANCHE_AGE(String label){
                        this.label = label;
                }
```

```
            private final String label;

        }

    public static String perform(int age){
        String tranche_age = "-12";
        if(age < 0){
                throw new IllegalArgumentException(
                        "age should not be less than 0: " + age);}
        if(age < 12){return TRANCHE_AGE.LT_12.label;}
        if(age >= 12 && age <= 25){return TRANCHE_AGE.B12_25.label;}
        if(age > 25 && age <= 34){return TRANCHE_AGE.B25_34.label;}
        if(age > 34 && age <= 65){return TRANCHE_AGE.B35_65.label;}
        return TRANCHE_AGE.MT_65.label;
    }
}
```